Improved approximate response time bounds for static-priority tasks

Thi Huyen Chau Nguyen, Pascal Richard
Lisi/Ensma
Poitiers, France
{nguyenc,richardp}@ensma.fr

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

Abstract

We consider sporadic tasks with static priorities and constrained deadlines to be executed upon a uniprocessor platform. Pseudo-polynomial time algorithms are known for computing worst-case response times for this task model. Some applications require to evaluate efficiently upper bounds of response times. For this purpose, we propose parametric algorithms that allow to make a tradeoff between quality of results and computational effort according to an input accuracy parameter. In this paper, we present a parametric polynomial-time algorithm for computing upper bounds of worst-case response times, that is based on an improved FPTAS *(Fully Polynomial Time Approximation Scheme). Then, we show that our bound does not achieve constant error bound in comparison with the exact worst-case response time. However, using the resource augmentation technique, we obtain a performance guarantee that allows to define a compromise between our response-time bound and processor capacity requirements. The algorithm average behavior is then analyzed through numerical experimentations.*

1. Introduction

We consider sporadic tasks with static priorities and constrained deadlines to be executed upon a uniprocessor platform. The Deadline Monotonic algorithm (DM) is optimal for scheduling this task model. A real-time system is said *feasible* if no deadline miss can occur at run-time. Basically, two main categories of algorithms have been proposed for verifying necessary and sufficient feasibility conditions of DM-scheduled systems: Response Time Analysis (RTA) [3, 12] and Processor Demand Analysis (PDA) [14, 15, 6]. Both approaches are known to have pseudo-polynomial time complexity, and it is currently unknown whether the task set feasibility can be computed in time polynomial in the representation of the task system.

RTA computes, for each task, the *worst-case response time* — the maximum interval of time between a release of a task and its completion. If, for all tasks, the response time is shorter than the deadline, then the task set is feasible. Instead, PDA searches, for each task, any instant earlier than the deadline, large enough to accommodate the computational requirement of the task itself and all the higher priority tasks. If such an instant exists for all tasks then the task set is feasible.

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 [8, 1, 2] and for static-priority scheduling [9, 16, 11, 17] which allow to verify the feasibility and to derive the response-time bound of a task at the same time:

- If the approximate test returns "feasible", then the considered task is guaranteed to be feasible on a unit-speed processor and a response-time bound can be deducted using the method presented in [16].
- If the test returns "infeasible", the task set is guaranteed to be infeasible on a *slower processor*, of computing capacity $(1-\epsilon)$. But, no conclusion can be taken if a unit-speed processor is considered, and no response-time bound can be obtained.

Such efficiently computed bounds (if exist) can introduce a loss of accuracy. It is desirable that this loss of accuracy be *quantified* in some manner in order to define a compromise between response-time bound guarantees and resource requirements.

This research. The objective of this paper is to define upper bounds on worst-case response time that are *efficiently computable*; and have *quantifiable deviations* from the exact bounds. We provide an alternative definition of the *Request Bound Function* (a characterization of work for static-priority tasks) which leads to a new FPTAS for

analysing system feasibility. Based on this approximate test, we define a new method for deducting response-time upper bounds. Our FPTAS and method are improvements of FPTAS's and methods presented in [16, 11, 17]. We then analyse the performance guarantees of our bound using the classical approximation ratio technique and the resource augmentation technique. Lastly, we give numerical experiments to compare our upper bound to other known ones and to capture its average performance guarantee.

Organization. Section 2 presents known results for validating static-priority tasks executed upon a uniprocessor platform. Section 3 presents an improvement of the approximate test presented in [11] and a new method for computing upper bounds of task worst-case response times. Section 4 presents results on worst-case error bounds of these approximate values of worst-case response times. Section 5 describes the numerical experimentations. Lastly, we conclude in Section 6.

2. Definitions

2.1. Task model

A sporadic task τ_i, $1 \leq i \leq n$, is defined by a worst-case execution time (WCET) C_i, a relative deadline D_i and a period T_i which is the minimal interval of time 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 [13]. The release jitter J_i of a task τ_i is the *largest* delay between its release time and (first) ready time. In this paper, we study systems with no release jitters to simplify notations, but jitters can be easily introduced, as shown in [11].

We assume that all tasks to be run upon a same processor are independent and do not suspend themselves. 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. Known results on worst-case response time analysis

2.2.1 Exact analysis

To the best of our knowledge, no polynomial-time algorithm is known for computing the exact worst-case response times ($wcrt$) for the considered task model. Pseudo-polynomial time algorithms are known and are based on 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 [14] for details):

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

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

Also in [14], a level-i busy period is defined as the interval of time where only tasks with a priority higher or equal to i are running.

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:*

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

Exact algorithms for calculating the worst-case response times of sporadic tasks are known. Using successive approximations starting from a lower bound of R_i, we can compute the $wcrt$ as the smallest fixed point of $W_i(t) = t$.

Another approach to compute $wcrt$ has been introduced in [16]. We provide further details since these principles will be reused in the remainder. We have shown that the worst-case response time of a task can be computed using Time Demand Analysis (see [14] for detail), for every feasible task set (and only for them). For a feasible task τ_i, it is sufficient to check the following testing set [14]:

$$S_i \stackrel{\text{def}}{=} \{aT_j \mid j = 1 \ldots i, a = 1 \ldots \lfloor \frac{D_i}{T_j} \rfloor\} \cup \{D_i\} \quad (3)$$

In [14], it is shown that τ_i is feasible if and only if $\exists t \in S_i, W_i(t) \leq t$. There exist some methodes for reducing the number of these scheduling points in [15] and in [6], but both of these methodes produce an exponential complexity of $\mathcal{O}(2^n)$.

Based on the scheduling set defined in Eq. (3), in [16], we first defined the notion of the critical point (under the assumption that the task τ_i will meet its deadline at execution time).

Definition 2 *The critical point for a feasible task τ_i is:*

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

Then, we have shown that the cumulative request bound function at the critical point of a given task τ_i leads to its worst-case response time.

Theorem 1 ([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 $R_i = W_i(t^*)$.*

2.2.2 Approximate $wcrt$ Analysis

Two main approaches have been designed for computing upper bounds of $wcrt$ in polynomial-time that are both based on linear approximation of the request bound function.

Linear-Time Response-Time Bound. Bini and Baruah [5] have shown that the worst-case workload, that is the maximum amount of time that the processor executes task τ_i in any interval of length t, can be bounded by a linear function (i.e. see [5] for details):

$$\text{LA}(\tau_i, t) \stackrel{\text{def}}{=} U_i\, t + C_i(1 - U_i) \tag{4}$$

Using such a linear function, [5] presents an upper bound of the worst-case response time of a task τ_i:

$$R_i \leq \frac{C_i + \sum_{j=1}^{i-1} C_j(1 - U_j)}{1 - \sum_{j=1}^{i-1} U_j} = ub_i^{BB} \tag{5}$$

We have shown in [4] that this well-known upper bound does not have a constant error bound (i.e., there exist 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. But, we have shown using a resource augmentation technique that this linear bound is an upper bound on a unit-speed processor and a lower bound on a half-speed processor. Thus, a processor speedup of two is an upper bound on the price being paid for using an efficiently computable upper bound on response time.

Approximation Scheme Technique. Approximation techniques have been recently used to define approximate feasibility tests. These tests are run in polynomial time according to the task set size and an "accuracy" parameter $1/\epsilon$ (i.e, they are FPTASs). Using the framework of the approximate feasibility analysis presented by Fisher and Baruah ([10]), we have shown how to compute approximate (upper bounds) worst-case response times in [16, 11].

A new approximation scheme will be presented in the next section, that improves best known results (e.g., [10, 11]) for feasibility analysis. Then, we present a new method for computing $wrct$ upper bounds that improves the results presented in [16, 17, 11].

3. Worst-Case Response-Time Bound

In this section, we introduce an approximate feasibility test which will lead to the deduction of a worst-case response-time bound:

1. We define a new FPTAS incorporating the linear function of Bini and Baruah (Eq. (4)) for analyzing feasibility of task sets. If this FPTAS returns a critical point (Definition 2), we can conclude that the given task τ_i is feasible. If no such point is obtained, then we can conclude that τ_i is infeasible on a processor of $(1 - \epsilon)$ speed (ϵ is the input accuracy parameter of the scheme).
2. We propose a "deduction method" which derives an upper bound of the worst-case response time and has as input the critical point computed in (1) (under the assumption that the scheme returns such a point, otherwise Eq. (5) can be used to obtain a worst-case response-time bound.)

3.1. Approximation Scheme

The request bound 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 [9]: we consider the first $(k - 1)$ steps of $\text{RBF}(\tau_i, t)$, where k is defined as $k \stackrel{\text{def}}{=} \lceil 1/\epsilon \rceil - 1$ and a linear approximation, thereafter. From this definition, we verify that $(k + 1) \geq 1/\epsilon$.

In [11], under the assumption that all the task parameters are integers, the following approximate request bound function was defined:

$$\delta(\tau_i, t) \stackrel{\text{def}}{=} \begin{cases} \text{RBF}(\tau_i, t) & \text{for } t \leq (k-1)T_i, \\ (t + T_i - 1)\frac{C_i}{T_i} & \text{otherwise.} \end{cases} \tag{6}$$

Thus, up to $(k - 1)T_i$, 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 the linear approximation based on Eq. (4) that will lead to an improved approximate feasibility algorithm in comparison with the known results.

For this purpose, we try to find out some properties of the critical points which allow to restrict the search for these

points to a set where the linear approximation in Eq. (4) is larger than (or equal to) the request bound function.

First of all, we prove the following property of all busy periods:

Lemma 1 *A level-i busy period cannot be completed in any interval $(mT_j, mT_j + C_j), 1 \leq j \leq i$, where m is an arbitrary integer.*

Proof: We prove the result by contradiction. Assume that a busy period is completed by time $t \in (mT_j, mT_j + C_j)$, then at time mT_j, this busy period is not completed yet. Moreover, at that time, a job of τ_j is released. Hence, the busy period must include the execution of this job. As this job cannot be completed before time $mT_j + C_j$, even if executed without any preemption, the busy period cannot be completed before time $mT_j + C_j$, which contradicts our assumption. ■

The following corollary follows directly from Lemma 1:

Corollary 1 *If t is the length of the synchronous level-i busy period then $\forall j, 1 \leq j \leq i, \exists m \in N, t \in [mT_j + C_j, (m+1)T_j]$.*

From the corollary above, we obtain this property of all critical points:

Corollary 2 *If t^* is the critical point of the task τ_i then $\forall j, 1 \leq j \leq i, \exists m \in N, t^* \in [mT_j + C_j, (m+1)T_j]$.*

Proof: We prove the result by contradiction. If $\exists j, 1 \leq j \leq i, \nexists m \in N, t^* \in [mT_j + C_j, (m+1)T_j] \Rightarrow \exists h \in N, t^* \in (hT_j, hT_j + C_j)$. Let t^p be the scheduling point which is right before t^* in S_i and t^{bp} the instant where the synchronous level-i busy period is completed. In [16] is shown that $t^{bp} \in (t^p, t^*]$. Since t^p and t^* are two adjacent points in S_i, we necessarily have $t^p \geq hT_j$. Consequently, $t^{bp} \in (hT_j, hT_j + C_j)$, which contradicts Corollary 1. ■

Now we can reduce the scheduling set defined in Eq. (3) (set of scheduling points where we search for the critical point) as follows:

Corollary 3 *For a feasible task τ_i, it is sufficient to check the following testing set:*

$$\begin{aligned} S_i^{(0)} &\stackrel{\text{def}}{=} \{aT_j \mid j = 1 \ldots i, a = 1 \ldots \lfloor \frac{D_i}{T_j} \rfloor\} \\ &\quad \cup \{D_i\}, \\ S_i &\stackrel{\text{def}}{=} S_i^{(0)} \backslash \{t \in S_i^{(0)} \mid t \in (aT_j, aT_j + C_j), \\ &\quad j = 1, \ldots, i,\ a \geq 0\}. \end{aligned} \tag{7}$$

τ_i is feasible if and only if $\exists t \in S_i, W_i(t) \leq t$.

Now we consider the linear function of Bini and Baruah in Eq. (4). This function can be stated as follows:

$$\text{LA}(\tau_i, t) \stackrel{\text{def}}{=} (t + T_i - C_i)\frac{C_i}{T_i}.$$

We prove that this function is also an upper bound of the request bound function under a certain condition of t.

Lemma 2 *$\forall \tau_j, \forall t \in [mT_j + C_j; (m+1)T_j]$, where m is an arbitrary integer, we have:*

$$\text{RBF}(\tau_j, t) = \left\lceil \frac{t}{T_j} \right\rceil C_j \leq \text{LA}(\tau_j, t).$$

Proof: For $t \in [mT_j + C_j; (m+1)T_j]$, we have:

$$\begin{aligned} mT_j + C_j \leq t &\qquad \leq (m+1)T_j \\ \left\lceil \frac{mT_j + C_j}{T_j} \right\rceil \leq \left\lceil \frac{t}{T_j} \right\rceil &\qquad \leq \left\lceil \frac{(m+1)T_j}{T_j} \right\rceil \end{aligned}$$

As $C_j \leq T_j$, we have $m + 1 \leq \left\lceil \frac{t}{T_j} \right\rceil \leq m + 1 \Rightarrow \left\lceil \frac{t}{T_j} \right\rceil = m + 1 \Rightarrow \text{RBF}(\tau_j, t) = (m+1)C_j$. Moreover,

$$\begin{aligned} t &\geq mT_j + C_j \\ \frac{t + T_j - C_j}{T_j} C_j &\geq \frac{mT_j + C_j + T_j - C_j}{T_j} C_j \\ \text{LA}(\tau_j, t) &\geq (m+1)C_j = \text{RBF}(\tau_j, t). \end{aligned}$$

The lemma is proved. ■

From this lemma and Corollary 3, we obtain that $\text{LA}(\tau_i, t)$ may be less than $\text{RBF}(\tau_i, t)$ at some time instants but for all instants which count, that is to say, instants t which are potential critical point of the task τ_i, this function is always an upper bound of the request bound function.

Consequently, it allows us to define an improved approximate request bound function:

$$\gamma(\tau_i, t) \stackrel{\text{def}}{=} \begin{cases} \text{RBF}(\tau_i, t) & \text{for } t \leq (k-1)T_i, \\ (t + T_i - C_i)\frac{C_i}{T_i} & \text{otherwise.} \end{cases} \tag{8}$$

From the definitions of $\gamma(\tau_i, t)$ and $\delta(\tau_i, t)$, 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 [11].

Theorem 2 *Under the assumption that all the task parameters are integers (hence, $\forall i, 1 \leq i \leq n, C_i \geq 1$), $\gamma(\tau_i, t)$ can be a tighter upper bound of $\text{RBF}(\tau_i, t)$ in comparison with $\delta(\tau_i, t)$:*

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

We shall see that an FPTAS can be based on $\gamma(\tau_i, t)$. Note also that while $\delta(\tau_i, t)$ can be used only if all the system parameters are integers, $\gamma(\tau_i, t)$ can be applied for task systems in which parameters are real numbers.

To define an approximate feasibility test based on the principle of PDA (Processor Demand Analysis [14]), we

Tasks	C_i	D_i	T_i
τ_1	2	4	4
τ_2	3	8	8

Table 1. Static-priority task set

define an approximate cumulative request bound function as:

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

By virtue of Corollary 3, according to the error bound ϵ leading to $k = \lceil 1/\epsilon \rceil - 1$, and applying the approximation technique of [10], we can define the following testing set $\widehat{S}_i \subseteq S_i$:

$$\begin{aligned} \widehat{S_i^{(0)}} &\stackrel{\text{def}}{=} \{bT_a \mid a = 1, \ldots, i-1, b = 1, \ldots, k-1\} \\ &\quad \cup \{D_i\}, \\ \widehat{S}_i &\stackrel{\text{def}}{=} \widehat{S_i^{(0)}} \setminus \{t \in \widehat{S_i^{(0)}} \mid t \in (aT_j, aT_j + C_j), \\ &\quad j = 1, \ldots, i,\ a \geq 0\}. \end{aligned} \tag{9}$$

We now establish the principle of the algorithm:

- 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 to the input size n and the input parameter $1/\epsilon$.

We present in Table 1 a task set in which the task τ_2 is not proved feasible using the test presented in [11] (which uses $\delta(\tau_i, t)$ to approximate $\text{RBF}(\tau_i, t)$) with $k = 2$. Using the new approximation of the Request-Bound Function ($\gamma(\tau_2, t)$), τ_2 is now proved feasible since $\widehat{W}_2(8) = 8$ (i.e., see Figure 3.1 that presents both approximate cumulative request bound functions for task τ_2).

3.2 Correctness of Approximation

We now prove the correctness of this approximate feasibility test. The key point for the correctness of the approximation scheme is $1 \leq \gamma(\tau_i, t)/\text{RBF}(\tau_i, t) \leq (1 + \epsilon)$, which shows that the deviation of our approximation from the exact RBF is bounded. 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.

The first theorem states that our approximation always exceeds or equals RBF at all scheduling points.

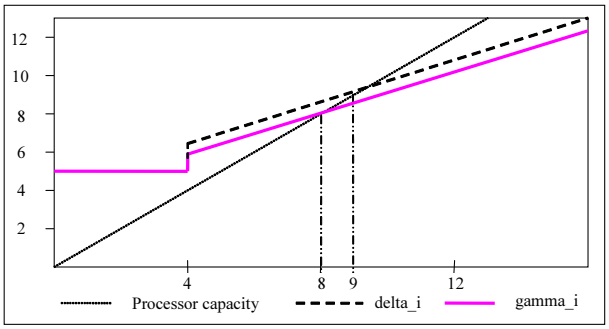

Figure 1. Approximate cumulative request bound functions on task set of Table 1

Theorem 3 $\forall j, 1 \leq j \leq i-1, \forall t \in S_i,\ \gamma(\tau_j, t) \geq \text{RBF}(\tau_j, t)$.

Proof: Straightforward from Lemma 2 and the new definition of S_i (Eq. (7)). ■

The second theorem gives the approximation ratio:

Theorem 4 $\forall j, 1 \leq j \leq i-1, \forall t \in (0, D_i],\ \gamma(\tau_j, t) \leq (1 + \frac{1}{k})\text{RBF}(\tau_j, t)$.

Proof: Let us recall the linear function in Eq. (4):

$$\text{LA}(\tau_j, t) \stackrel{\text{def}}{=} (t + T_j - C_j)\frac{C_j}{T_j}$$

For $t \leq (k-1)T_j$, it is evident that the inequality holds as $\gamma(\tau_j, t) = \text{RBF}(\tau_j, t)$. We consider now the case $t > (k-1)T_j$, where $\gamma(\tau_j, t) = \text{LA}(\tau_j, t)$.

Since $\text{RBF}(\tau_j, t)$ is a step function and $\text{LA}(\tau_j, t)$ is a strictly increasing linear function, it is clear that the ratio $\text{LA}(\tau_j, t)$ / $\text{RBF}(\tau_j, t)$ is an increasing monotonic function in all intervals $(mT_j, (m+1)T_j], m \in N, m \geq (k-1)$, i.e., $\text{LA}(\tau_j, t)$ / $\text{RBF}(\tau_j, t)$ is strictly increasing except at instant t which is a multiple of the period T_j where this ratio reaches a local maximum, and this function is left-continuous. Since the local maxima of $\text{LA}(\tau_j, t)$ / $\text{RBF}(\tau_j, t)$ are attained at the instants $(m+1)T_j, m \geq (k-1)$, then for finding its global maximum with $t > (k-1)T_j$, we just have to consider the instants $t = hT_j$ with $h \in N, h \geq k$.

We have

$$\begin{aligned} \frac{\text{LA}(\tau_j, hT_j)}{\text{RBF}(\tau_j, hT_j)} &= \frac{(h+1)T_j - C_j}{hT_j} \\ &= 1 + \frac{1}{h} - \frac{C_j}{hT_j} \\ &\leq 1 + \frac{1}{h} \leq 1 + \frac{1}{k}. \end{aligned}$$

The theorem is proved. ■

Using the same approach presented in [10], now we can establish the correctness of approximation.

First of all, we prove that if a task τ_i is concluded infeasible by the approximate test, then it is infeasible with certainty upon a processor of $(1-\epsilon)$ capacity.

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

Proof: We will prove that if $\forall t \in (0, D_i]$, $\widehat{W}_i(t) > t$, then $\forall t \in (0, D_i]$, $W_i(t) > (1-\epsilon)t$:

$$\begin{aligned} \widehat{W}_i(t) &> t \\ C_i + \sum_{j=1}^{i-1} \gamma(\tau_j, t) &> t. \end{aligned}$$

From Theorem 4, for any instant $t \in (0, D_i]$, we have:

$$\begin{aligned} C_i + \sum_{j=1}^{i-1} \frac{k+1}{k} \text{RBF}(\tau_j, t) &> t \\ \frac{k+1}{k}\left(C_i + \sum_{j=1}^{i-1} \text{RBF}(\tau_j, t)\right) &> t \\ \frac{k+1}{k} W_i(t) &> t \\ W_i(t) &> \frac{k}{k+1} t. \end{aligned}$$

We have

$$\frac{k}{k+1} = 1 - \frac{1}{k+1}.$$

Note that $k \stackrel{\text{def}}{=} \lceil \frac{1}{\epsilon} \rceil - 1$ and $\lceil \frac{1}{\epsilon} \rceil \geq \frac{1}{\epsilon}$, we obtain

$$\frac{k}{k+1} \geq 1 - \epsilon.$$

Thus for any $t \in (0, D_i]$,

$$W_i(t) > (1-\epsilon)\, t.$$

The theorem follows. ■

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 $t \in \widehat{S}_i$ such that $\widehat{W}_i(t) \leq t$, then $W_i(t) \leq t$.*

Proof: Since $t \in \widehat{S}_i$, then $t \in S_i$. Moreover, Theorem 3 allows to conclude that $\forall t \in S_i, W_i(t) \leq \widehat{W}_i(t)$. Hence, $W_i(t) \leq t$ and τ_i is feasible. ■

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

Theorem 7 *If $\forall t \in \widehat{S}_i, \widehat{W}_i(t) > t$, then we also verify that $\forall t \in (0, D_i], \widehat{W}_i(t) > t$.*

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 $\widehat{W}_i(t)$ is an non-decreasing step left-continuous function, we conclude that $\forall t \in (t_1, t_2), \widehat{W}_i(t) > t$. Then the theorem follows. ■

3.3. Approximate Worst-Case Response Times (Deduction Method)

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 its worst-case response-time upper bound. 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 [5] for instance).

In [11] is presented a deduction method which leads to the following upper bound:

Definition 3 ([11]) *Consider a task τ_i such that there exists a time $t \in \widehat{S}_i$ satisfying $\widehat{W}_i(t) \leq t$, then an approximate upper bound of its worst-case response time is defined by:*

$$\begin{aligned} \hat{t}^* &\stackrel{\text{def}}{=} \min\left\{ t \in \widehat{S}_i \middle| \widehat{W}_i(t) \leq t \right\}, \\ \widetilde{R_i} &\stackrel{\text{def}}{=} \widehat{W}_i(\hat{t}^*). \end{aligned} \tag{10}$$

We propose next a new method for deducting a response-time upper bound of a task τ_i in the assumption that τ_i is concluded feasible by the approximate test.

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

$$\begin{aligned} \hat{t}^* &\stackrel{\text{def}}{=} \min\left\{ t \in \widehat{S}_i \middle| \widehat{W}_i(t) \leq t \right\}, \\ \widehat{R_i} &\stackrel{\text{def}}{=} W_i(\hat{t}^*). \end{aligned} \tag{11}$$

Now we prove that such a method defines a tighter worst-case response-time upper bound of task τ_i in comparison with the upper bound $\widetilde{R_i}$ obtained by the existing deduction method.

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

$$R_i \leq \widehat{R_i} \leq \widetilde{R_i}.$$

Proof: Let t^* be the critical point corresponding to the worst-case response time of τ_i (i.e., the first time instant in S_i such that $W_i(t) \leq t$). Let $\hat{t}^*$ be the first time instant in $\widehat{S}_i$

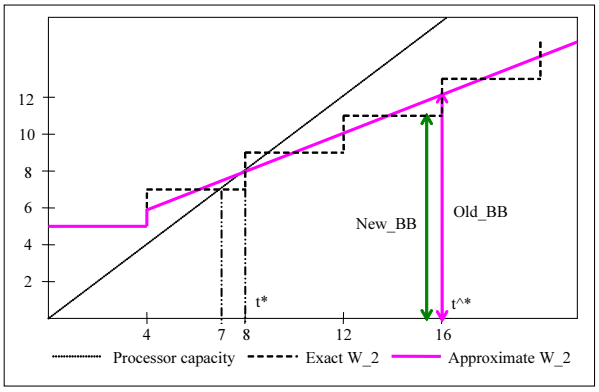

Figure 2. Exact response time and upper bounds using old and new methods of task set of Table 2

Tasks	C_i	D_i	T_i	t^*	$\hat{t}^*$	R_i	$\widehat{R_i}$	$\widetilde{R_i}$
τ_1	2	4	4					
τ_2	3	16	16	8	16	7	11	12

Table 2. Static-priority task set and its feasibility analysis

such that $\widehat{W_i}(t) \leq t$ (i.e., $\hat{t}^*$ is critical). Since $\widehat{S_i} \subseteq S_i$, we have $\hat{t}^*$ is also the first time instant in S_i satisfying $\widehat{W_i}(t) \leq t$.

Since Theorem 6 shows that $\forall t \in S_i, \widehat{W_i}(t) \geq W_i(t)$, we necessarily have $t^* \leq \hat{t}^*$. Since $W_i(t)$ is a non-decreasing function, $W_i(t^*) \leq W_i(\hat{t}^*)$. Equivalently, $R_i \leq \widehat{R_i}$.

Since $\forall t \in S_i, \widehat{W_i}(t) \geq W_i(t)$, then $W_i(\hat{t}^*) \leq \widehat{W_i}(\hat{t}^*)$, hence we obtain the right side of the inequality. ■

We illustrate Theorem 8 by considering the task set presented in Table 2. We just analyse the feasibility of the task τ_2. Using Deadline Monotonic scheduling, we obtain $t^* = 8$ and $R_2 = W_2(t^*) = 7$. We choose $\epsilon = 0.4$ thus $k = 2$ and $\widehat{S_2} = \{4, 16\}$. $\widehat{W_2}(4) > 4$ and $\widehat{W_2}(16) < 16 \Rightarrow \hat{t}^* = 16 \Rightarrow \widehat{R_2} = W_2(\hat{t}^*)$=11 and $\widetilde{R_2} = \widehat{W_2}(\hat{t}^*) = 12$. As expected, we have $R_2 \leq \widehat{R_2} \leq \widetilde{R_2}$.

4 Worst-case analysis of error bound

4.1 Approximation Ratio analysis

The performance guarantee of our upper bound can be analysed through its *approximation ratio*. Let a be the value obtained by an algorithm A, and opt be the exact (i.e., optimal) value; algorithm A has a approximation ratio of c, where $c \geq 1$, if and only if $opt \leq a \leq c \times opt$ for all inputs to the algorithm A (if such a c does not exist, then algorithm A is said to have no approximation ratio).

The next result states that the approximate response-time bound $\widehat{R_i}$ does not, in fact, have an approximation ratio.

Theorem 9 *For any accuracy parameter ϵ, there exist some task systems for which $cR_i \leq \widehat{R_i}$ for any integer c.*

Proof: Let k be defined by $k = \lceil 1/\epsilon \rceil - 1$.

We prove this theorem by demonstrating a task system and a task τ_i for which $\widehat{R_i}/R_i$ tends to ∞. All tasks in our system will have $D_i = T_i$; hence, let us represent the parameters of a task τ_i by an ordered pair (C_i, T_i). Consider the following task set: $\tau_1 = (K, 2K+\lambda)$, $\tau_2 = (K, 2K+\lambda)$ and $\tau_3 = (\lambda K, K(2K+\lambda) + \frac{2K(K+\lambda)}{\lambda})$, where λ is an arbitrarily small positive number such that $\frac{1}{\lambda}$ is an integer and K is an arbitrary integer that is strictly greater than $k-1$. Note that by construction, $D_3 = T_3 > (k-1)(2K+\lambda)$. Also, from Eq. (9), $\widehat{S_3}$ contains the point $t = D_3$.

Using the Deadline Monotonic Scheduling policy, the task τ_3 can only be executed λ units of time within every subsequent interval of time of length $2K + \lambda$. As a consequence, the exact worst-case response time of τ_3 is: $R_3 = K(2K + \lambda)$. The approximation switches to a linear approximation at $(k-1)(2K+\lambda)$ which is strictly less than $K(2K + \lambda)$. Thus we consider that $t > (k-1)(2K + \lambda)$. Hence, in $\widehat{S_3}$, we consider only the instant $t = D_3$.

The approximate response-time analysis leads to:

$$\widehat{W_3}(t) \quad = \quad \lambda K + 2(t + K + \lambda)\frac{K}{2K + \lambda}$$

Solving the condition of the critical point $\widehat{W_3}(t) \leq t$, we have:

$$t \geq K(2K + \lambda) + \frac{2K(K + \lambda)}{\lambda}$$

Since $t = D_3$ satisfies the inequality above, it is the critical point of τ_3.

Applying Definition 4 and replacing $\hat{t}^* = D_3$, we obtain the approximate worst-case response time of the task t_3:

$$\widehat{R_3} = W_3(\hat{t}^*) = K(2K + 2 + \lambda) + \frac{2K^2}{\lambda}$$

Thus,

$$\lim_{\lambda \to 0} \frac{\widehat{R_3}}{R_3} = \lim_{\lambda \to 0} \frac{K(2K + 2 + \lambda) + \frac{2K^2}{\lambda}}{K(2K + \lambda)} = \infty$$

and the theorem is proved. ■

4.2 Resource Augmentation Analysis

Theorem 9 above reveals that the approximate response-time bound $\widehat{R_i}$ does not offer any quantifiable performance guarantee, according to the conventional approximation ratio measure that is used in optimization theory. However, an alternative approach towards approximate analysis – the technique of *resource augmentation* – is becoming increasingly popular in real-time scheduling theory. In this technique, the performance of the algorithm under analysis is compared with that of an optimal algorithm *that runs on a slower processor.* In this section, we apply this resource augmentation technique to quantify the deviation of $\widehat{R_i}$ from optimality.

Firstly, we assume that the processor slowdown factor is applicable to all tasks in the system (there exist such systems in which the worst-case execution times (WCET) of some tasks are always the same regardless the capacity of the processor). As a consequence of this assumption, we can consider a task running on an s-capacity processor like a task with the WCET augmented $1/s$ times running on a unit-speed processor (but all the other parameters are exactly the same as when it runs on the s-capacity processor). Hence, letting us denote C_i^s, D_i^s, T_i^s respectively the worst-case execution time, the relative deadline and the period of a task τ_i on a s-capacity processor, then we necessarily have:

$C_i^s = \frac{C_i}{s}$
$D_i^s = D_i$
$T_i^s = T_i$

Now we introduce some further notations of the request bound and workload function of a task τ_i on a s-speed processor:

$$\text{RBF}^s(\tau_i, t) \stackrel{\text{def}}{=} \left\lceil \frac{t}{T_i^s} \right\rceil C_i^s \quad (12)$$

$$W_i^s(t) \stackrel{\text{def}}{=} C_i^s + \sum_{j=1}^{i-1} \text{RBF}^s(\tau_j, t) \quad (13)$$

Clearly we have the following lemma:

Lemma 3 $\forall i, 1 \leq i \leq n, \forall t, \text{RBF}^s(\tau_i, t) = \frac{\text{RBF}(\tau_i,t)}{s}$, $W_i^s(t) = \frac{W_i(t)}{s}$.

Using this relationship between the request bound function, the workload function on an s-capacity processor and those on a unit speed processor, along with exploiting the properties of our approximate request bound function, we can prove the worst-case slowdown factor of the upper bound $\widehat{R_i}$.

Theorem 10 *The bound $\widehat{R_i}$ (Eq. (11)) is*

1. *an upper bound on the worst-case response time of τ_i; and*
2. *a lower bound on the worst-case response time of τ_i* if the system is implemented upon a processor of speed $k/(k+1)$.

Proof: The first statement can be directly obtained from Theorem 8. We prove the second statement.

Let us denote $s = \frac{k}{k+1} < 1$

By Theorem 4, $\forall j, 1 \leq j \leq i-1, \forall t \in (0, D_i]$, we have:

$$\begin{aligned}
\gamma(\tau_j, t) &\leq \frac{k+1}{k}\text{RBF}(\tau_j, t) \\
\sum_{j=1}^{i-1}\gamma(\tau_j, t) &\leq \sum_{j=1}^{i-1}\frac{k+1}{k}\text{RBF}(\tau_j, t) \\
C_i + \sum_{j=1}^{i-1}\gamma(\tau_j, t) &\leq \frac{k+1}{k}C_i + \sum_{j=1}^{i-1}\frac{k+1}{k}\text{RBF}(\tau_j, t) \\
C_i + \sum_{j=1}^{i-1}\gamma(\tau_j, t) &\leq C_i^s + \sum_{j=1}^{i-1}\text{RBF}^s(\tau_j, t) \\
\widehat{W_i}(t) &\leq W_i^s(t). \quad (14)
\end{aligned}$$

Let t^* be the critical point corresponding to the worst-case response time of τ_i on the processor of speed-s (i.e., the first time instant in S_i such that $W_i^s(t) \leq t$). Let $\hat{t}^*$ be the first time instant in $\widehat{S}_i$ such that $\widehat{W_i}(t) \leq t$ (i.e., $\hat{t}^*$ is critical). Since $\widehat{S}_i \subseteq S_i$, we have $\hat{t}^*$ is also the first time instant in S_i satisfying $\widehat{W_i}(t) \leq t$

From (14), we necessarily have $\hat{t}^* \leq t^*$. Since $W_i^s(t) = \frac{W_i(t)}{s} > W_i(t)$ and the fact that $W_i(t)$ and $W_i^s(t)$ are non-decreasing functions, we have $W_i(\hat{t}^*) \leq W_i^s(t^*)$. Equivalently, $\widehat{R_i}$ is a lower bound of the exact response time $W_i^s(t^*)$. ■

How is the systems designer to interpret Theorem 10 above? First, it is guaranteed that $\widehat{R_i}$ is indeed an upper bound on R_i; hence, it is a safe estimate of the exact worst response time. And while Theorem 10 is unable to bound the amount by which $\widehat{R_i}$ exceeds the actual value of R_i, it does assure the designer that [s]he could have obtained a worst-case response time no better than $\widehat{R_i}$ if the system had instead been implemented upon a processor $k/(k+1)$ times as fast. Stated differently, *a processor speedup of $(k+1)/k$ is an upper bound on the price being paid for using an upper bound on response time that is computed in polynomial time according to task parameters and the constant $1/\epsilon$.*

5 Experiments

In this section, we describe numerical experimentations that we performed in order to compare our upper bound to other known ones and to capture its average performance guarantee.

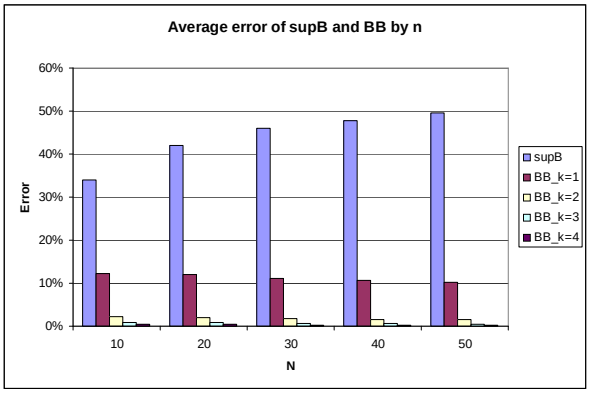

Figure 3. Average error of $supB$ and BB

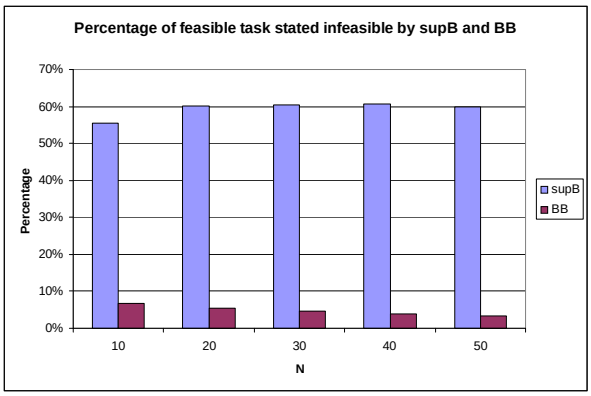

Figure 4. Feasible tasks stated infeasible by $supB$ and BB

Stochastic Model. We randomly generated task sets with constrained deadlines. Unbiased utilizations were generated using the UUniFast algorithm [7]. 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 (which leads to k, the number of steps to be considered before the linear approximation). For fixed parameters, every experiment is replicated 400 times in order to achieve unbiased statistics.

Comparison with linear-time bounds. We compared our bound (denoted BB) with the best known upper bound computed in linear time for every task τ_i: $supB_i$, the upper bound of Bini and Baruah computed using the rounding principle presented in [5]:

$$supB_i = \frac{C_i + \sum_{j=1}^{i-1} C_j(1 - U_j)}{1 - \sum_{j=1}^{i-1} U_j}.$$

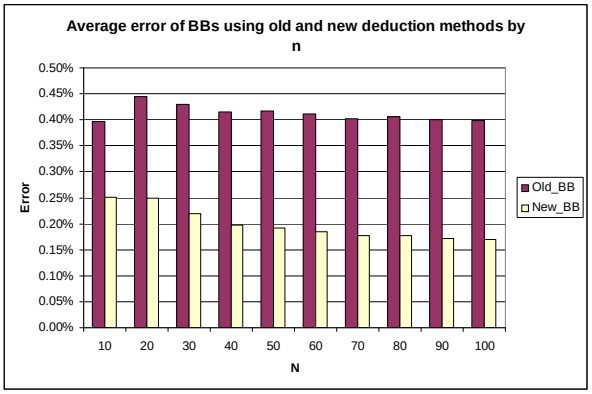

Figure 5. Average errors of old_BB and new_BB

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., $(ub_i - R_i)/R_i$, where ub_i is an upper bound and R_i is the exact worst-case response time) and the rate of tasks stated "infeasible" using the upper bound (i.e., $ub_i > D_i$) for feasible tasks (i.e., $R_i \leq D_i$). Numerical results are presented in Figure 3 and Figure 4. 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 bound, even if $k = 1$ (i.e., the smallest possible value since $k = \lceil 1/\epsilon \rceil - 1$ and $0 < \epsilon < 1$). The average error is less than 1% when $k = 3$ (i.e., $0.25 \leq \epsilon < 0.33$). Concerning the second graph, we see that our approach is less pessimistic since only few feasible tasks are not accepted by our method. What is more interesting is that as the task number increases, the average error of $supB$ increases while the average error of our bound decreases.

Comparison with the best known approximation bound

In Figure 6, we computed and compared the average errors by k of two upper bounds: the bound Old_Algo obtained from [11] and the bound New_Algo resulted from the algorithm presented in this paper. Our bound shows a slight improvement of about 1 % in comparison with the best known approximation bound Old_Algo.

Comparison with bounds obtained from existing deduction method. Based on the same approximation scheme

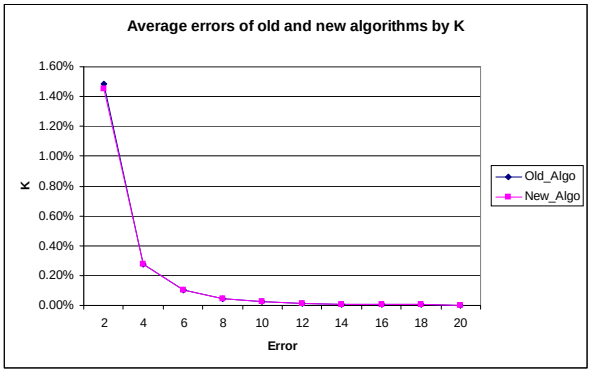

Figure 6. Average error of old and new algorithms by k

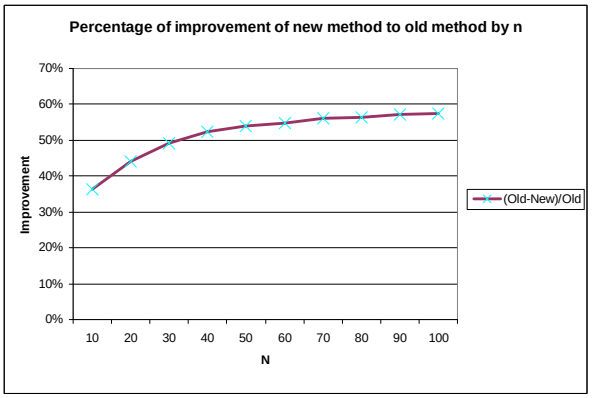

Figure 7. Average error improvement of new_BB to old_BB

presented in this paper, for every task stated "feasible", we computed and compared two upper bounds: the bound Old_BB obtained from the existing method (Definition 3) and the bound New_BB derived using our deduction method (Definition 4):

$$\begin{aligned} \hat{t}^* &\stackrel{\text{def}}{=} \min\left\{t \in \widehat{S}_i \middle| \widehat{W}_i(t) \leq t\right\}, \\ old\_BB &\stackrel{\text{def}}{=} \widehat{W}_i(\hat{t}^*), \\ new\_BB &\stackrel{\text{def}}{=} W_i(\hat{t}^*). \end{aligned}$$

In Figure 5, we report the average error of these two bounds in comparison with the exact worst-case response times, when the task number varies from 10 to 100 (step 10). We can see that with our method, the average error is decreased to approximately one half. In Figure 7, we plot this average error improvement with respect to the task number. It can be noticed that this improvement increases as the task number increases.

Resource augmentation analysis. For every task set, we computed (using a binary search) the exact slowdown factor s so that our bound is the actual response time upon an s-speed processor.

In Figure 8, we monitored the average and the minimum slowdown factor according to the number of steps to be considered before the linear approximation k. It can be seen that the average slowdown factor is always between $k/(k+1)$ and 1 and very close to 1, which means that the processor capacity that one might waste when using our approach is very small. As expected, the minimum value is exactly equal to our theoretical bound ($k/(k+1)$), i.e., the worst-case slowdown factor has been reached for every simulation run.

In Figure 9, we compared the average slowdown factor (SDF) of $supB$ and that of our bound. It can be noticed that concerning the average SDF indicator, our bound shows

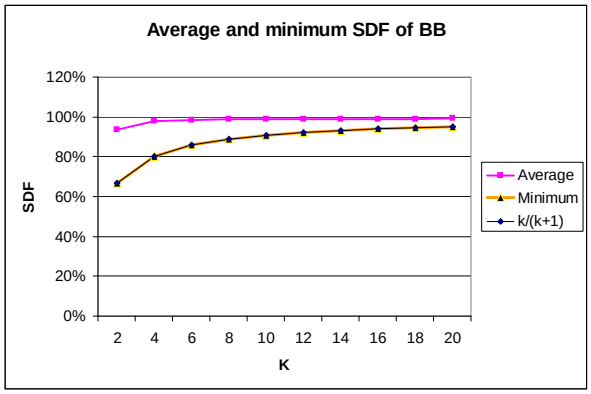

Figure 8. Average and minimum Slowdown Factor of BB

an approximately 25% improvement over the bound $supB$ even with $k = 2$. The average SDF with $k = 4$ is greater than 97%, that is to say, the average "processor capacity waste" is less than 3%.

6 Conclusions

We define a FPTAS and a deduction method to derive worst-case response-time upper bounds for static-priority tasks with constrained deadlines. If the accuracy parameter ϵ used to define the time spent before starting an approximate analysis 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 $1/\epsilon$.

We have to quantify the loss of accuracy. We have shown that our upper bound does not offer non-trivial performance guarantees according to the conventional approximation ra-

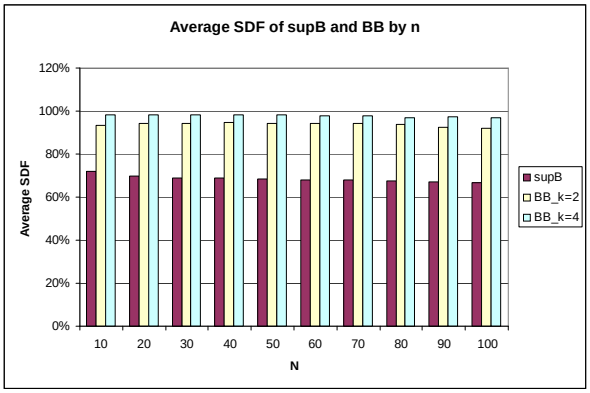

Figure 9. Average Slowdown Factor of $supB$ and BB

tio. However, using the concept of resource augmentation, we obtain the following quantitative guarantee — our bound is indeed an upper bound on the exact response time, and the exact response time would necessarily be at least as large as this bound if the system were instead implemented upon a processor that is at most only $k/(k+1)$ (where $k \stackrel{\text{def}}{=} \lceil 1/\epsilon \rceil - 1$) times as fast.

References

[1] K. Albers and F. Slomka. An event stream driven approximation for the analysis of real-time systems. *proc. Euromicro Int. Conf. on Real-Time Systems (ECRTS'04)*, pages 187–195, 2004.

[2] K. Albers and F. Slomka. Efficient feasibility analysis for real-time systems with edf scheduling. *proc. of Design, Automation and Test in Europe Conference (Date'05)*, 2005.

[3] 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.

[4] S. Baruah, E. Bini, T. Nguyen, and P. Richard. Continuity and approximability of response time bounds. *Euromicro Conf. on Real-Time Systems (ECRTS'07), Work-in Progress*, 2007.

[5] E. Bini and S. Baruah. Efficient computation of response time bounds under fixed-priority scheduling. *proc. Int. Real-Time and Network Systems (RTNS'07)*, 2007.

[6] E. Bini and G. Buttazzo. Schedulability analysis of periodic fixed priority systems. *IEEE Transactions on Computers*, 53, Nov 2004.

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

[8] S. Chakraborty, S. Kunzli, and L. Thiele. Approximate schedulability analysis. *proc. Int. Symposium on Real-Time Systems (RTSS'02)*, 2002.

[9] N. Fisher and S. Baruah. A fully polynomial-time approximation scheme for feasibility analysis in static-priority systems with arbitrary relative deadlines. *proc. Euromicro Int. Conf. on Real-Time Systems (ECRTS'05)*, pages 117–126, July 2005.

[10] N. Fisher and S. Baruah. A polynomial-time approximation scheme for feasibility analysis in static-priority systems with bounded relative deadlines. *Proc. Int. Conf. on Real-Time Systems (RTNS'05)*, pages 233–249, 2005.

[11] N. Fisher, T. Nguyen, J. Goossens, and P. Richard. Parametric polynomial-time algorithms for computing response-time bounds for static-priority tasks with release jitters. *13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'07)*, 2007.

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

[13] J. Lehoczky. Fixed priority scheduling of periodic tasks with arbitrary deadlines. *proc. IEEE Int. Real-Time System Symposium (RTSS'90)*, pages 201–209, 1990.

[14] J. Lehoczky, L. Sha, and Y. Ding. The rate monotonic scheduling algorithm: exact characterization and average case behavior. *proc. IEEE Int. Real-Time System Symposium (RTSS'89)*, pages 166–171, 1989.

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

[16] P. Richard and J. Goossens. Approximating response times for static-priority tasks with release jitters. *WIP, Euromicro Int. Conf. on Real-Time Systems (ECRTS'06),*, 2006.

[17] P. Richard, J. Goossens, and N. Fisher. Approximate feasibility analysis and response-time bounds of static-priority tasks with release jitters. *proc. Int. Real-Time and Network Systems (RTNS'07)*, 2007.