The Fully Polynomial-Time Approximation Scheme for Feasibility Analysis in Static-Priority Systems with Arbitrary Relative Deadlines Revisited

Thi Huyen Chau Nguyen, Pascal Richard, Nathan Fisher

The Fully Polynomial-Time Approximation Scheme for Feasibility Analysis in Static-Priority Systems with Arbitrary Relative Deadlines Revisited

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

Nathan Fisher
Department of Computer Science
Wayne State University
fishern@cs.wayne.edu

Abstract

We consider sporadic tasks with static priorities and arbitrary deadlines to be executed upon a uniprocessor platform. Pseudo-polynomial time algorithms are known for computing worst-case response times for this task model. Fisher and Baruah proposed in 2005 an approximate feasibility test running in polynomial time according to an accuracy parameter ϵ (i.e., approximation scheme or FPTAS*). We show that this algorithm is valid for the Rate Monotonic scheduling policy, but must be slightly modified to be applicable for arbitrary priority assignment. We also introduce some improvements to achieve a higher efficiency. We illustrate the improved algorithm on a detailed example.*

1 Introduction

A feasibility test is an algorithm used to check if a task set is feasible or not. It can be either used off-line if the system workload is predictable or on-line if new tasks can be admitted at run-time. Two main and linked approaches are used to check whether a task is feasible or not: Response Time Analysis (RTA) that returns the worst-case response time of a task (i.e., the maximum amount of time between a release and a completion of jobs generated by the task) or a Processor Demand Analysis (PDA) that returns a boolean value, feasible or infeasible. In both cases, exact feasibility tests are known for static priority tasks with arbitrary deadlines [6]. However, it is already known that worst-case response time computation is an NP-hard problem [3] and cannot be approximated within a constant approximation factor, even in the context of tasks with constrained-deadlines (i.e., deadlines are less than or equal to activation periods). Nevertheless, positive results are known using the resource augmentation techniques [4, 5, 1, 7]. Please notice that the feasibility problem of static priority tasks with constrained-deadlines is still an open issue from the computational point of view: pseudo-polynomial time algorithms are known but it is not known if this problem is NP-hard or not.

However, these exact tests have very efficient implementations in practice that render them quite suitable for feasibility analysis of FP systems. Nevertheless, there are certain scenarios in which schedulability tests cannot be used, where the complexity and execution time of exact tests may become huge drawbacks as in dynamic systems, optimization of large scale systems, etc. For these problems, a pseudo-polynomial algorithm for computing the task set feasibility may be unacceptably slow; instead, it may be acceptable to use a faster algorithm that provides an approximate, rather than exact, analysis [2, 5, 1, 7].

This research. In [5, 7] is studied the approximate feasibility analysis of static priority tasks with constrained-deadlines. In this paper, we revisit the approximate feasibility analysis presented in [4] that considers sporadic tasks with static priorities and arbitrary deadlines to be executed upon a uniprocessor platform. Pseudo-polynomial time algorithms are known for computing worst-case response times for this task model. Fisher and Baruah proposed in 2005 an approximate feasibility test running in polynomial time according to an accuracy parameter ϵ (i.e., approximation scheme or FPTAS). We will show that, while the algorithm remains correct for rate-monotonic (RM) priority assignments, the algorithm is not correct for non-RM priority assignments and requires some modifications. We revised the algorithm in order to correct the detected problem and we also provide an improvement. We illustrate the revised algorithm on a detailed example.

Organization. Section 2 presents static-priority tasks with arbitrary deadlines. Section 3 presents the task model and basic functions for schedulability analysis of fixed-priority task systems. We shall detail the main ideas of Fisher and Baruah FPTAS. Then, we give a counter example of Fisher and Baruah's FPTAS for arbitrary priority assignment and its correctness for RM. Section 4 presents the revised algorithm according to the detected problems in the former algorithm, presents the revised algorithms and proves its correctness. Lastly, we conclude in Section 5 and give details for some future works.

2 Task model and notations

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. We assume that deadlines are not related to periods (i.e., tasks with arbitrary deadlines). We suppose that $\forall i, C_i > 0$. The l^{th} job of τ_i is denoted as $\tau_{i,l}$. The utilization factor of task τ_i is the fraction of time that τ_i requires the processor: $U_i \overset{\text{def}}{=} C_i/T_i$. The utilization factor of the task set is: $U \overset{\text{def}}{=} \sum_{j=1}^{n} U_j$.

We assume that sporadic 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. A task can be preempted by any task of higher priority. 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.

The total execution time requested by task τ_i over time interval $[0, t)$ is defined by the request bound function:

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

Then, the cumulative processor demand function of tasks having a priority higher or equal to $\tau_{i,l}$ is defined by:

$$W_{i,l}(t) \overset{\text{def}}{=} lC_i + \sum_{j<i} \text{RBF}(\tau_j, t) \tag{2}$$

3 Analysis of Fisher and Baruah Approximation Scheme

Next, we first present the main ideas of the Fisher and Baruah approximation scheme for checking the feasibility of a sporadic task with an arbitrary deadline. Then, we provide a counter example of an important results used in the algorithm. We think that these problems cannot be solved without changing some formulas and proofs. Nevertheless, please notice that the algorithm structure will be preserved.

3.1 Fisher and Baruah Algorithm

In order to approximate the request bound function according to an error bound $1 + \epsilon$ (accuracy parameter, $0 < \epsilon < 1$), the algorithm presented in [4] uses the approximation principle introduced in [5] for sporadic tasks with constrained-deadlines (i.e., $D_i \leq T_i$ for all tasks): we consider the first $(k-1)$ steps of $\text{RBF}(\tau_i, t)$, where k is defined as $k \overset{\text{def}}{=} \lceil 1/\epsilon \rceil - 1$ and a linear approximation, thereafter. From this definition, we verify that $(k+1) \geq 1/\epsilon$.

Hence, an approximate resquest-bound function can be defined as follows:

$$\delta(\tau_i, t) \overset{\text{def}}{=} \begin{cases} \text{RBF}(\tau_i, t) & \text{for } t \leq (k-1)T_i, \\ (t + T_i)U_i & \text{otherwise.} \end{cases} \tag{3}$$

Thus, up to time $(k-1)T_i$, 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.

The approximate cumulative processor demand function of each job $\tau_{i,l}$ is defined as follows:

$$\widehat{W}_{i,l}(t) = lC_i + \sum_{j<i} \delta(\tau_j, t) \tag{4}$$

Fisher and Baruah's algorithm [4] (denoted FB05 below and presented in Algorithm 1) is based on analysing a polynomial number of scheduling points in the Lehoczky's testing set ([6]) using the approximate cumulative processor demand function. In a first stage of FB05 algorithm, the testing set is explored for checking the feasibility of jobs of τ_i (i.e., jobs with completion time prior or equal to $\max_{j \in 1,\ldots,i-1} \{(k-1)T_j\}$). Thus, the considered testing set is:

$$\tilde{S}_i \overset{\text{def}}{=} \{t = bT_a : a = 1 \ldots i, b = 1 \ldots k-1\} \cup \{0\} \tag{5}$$

We call two elements t_{a-1} and t_a $(t_{a-1} < t_a)$ in some set *adjacent* if there exists no t in that set satisfying $t_{a-1} < t < t_a$. Using this testing set, the first stage determines whether all jobs of task τ_i with deadlines less than $t = \max_{\{j \in \{1 \ldots i\}} \{(k-1)T_j\}$ are schedulable. If no deadline is missed up to time t, then the first stage algorithm returns the lowest indexed job (denoted h) whose request is not satisfied by time t.

Task τ_i may generate a pseudo-polynomial number of jobs in the level-i busy period. Nevertheless, Fisher and Baruah claim that in every interval $(t_{a-1}, t_a]$ defined by two adjacent scheduling points in the testing set (i.e., there is no scheduling point between t_{a-1} and t_a in the testing set), only one job of τ_i can have its deadline in $(t_{a-1}, t_a]$. They defined a simple way to compute the index of the first job of τ_i released in $(t_{a-1}, t_a]$. For that purpose, a "close" approximation of the workload function is defined to determine the jobs having the execution requests satisfied by time t (Equations 5 in [4]) are:

$$\widetilde{W}_i(t) = \delta(\tau_i, t) + \sum_{j<i} \delta(\tau_j, t) \tag{6}$$

Based on that function, Fisher and Baruah indicate that the index of the most recently released job of τ_i to have its execution request satisfied by time t is:

$$\left\lceil \frac{t}{T_i} \right\rceil - Z_i(t) \tag{7}$$

Algorithm 1. Fisher and Baruah ApproxFirstStage

```
input :
        $\tau = C[n], T[n], D[n]$ : array of integers              /* Task parameters */
        $i$ : integer                                              /* Index of the analysed task */
        $k$ : integer                                              /* The FPTAS accuracy parameter */
output: lowest_active : integer                                    /* Lowest active job */

Construct $\tilde{S}_i$ using Equation 5                           /* Ordered set of scheduling points */
lowest_active=1                                                    /* index of job to be analysed */
foreach $t_a \in \tilde{S}_i - \{0\}$ do
    if $t_a > (lowest\_active - 1)T_i + D_i$ then                  /* a job completes in $(t_{a-1}, t_a]$ */
        Let $t_{a-1}$ be the adjacent element prior to $t_a$ in $\tilde{S}_i$;
        Let $y$ be the total execution requirement of all jobs of priority greater than $\tau_i$ released at time $t_a$;
        $a = (t_{a-1}, \widehat{W}_{i,\text{lowest\_active}}(t_{a-1}) + y)$ ;
        $b = (t_a, \widehat{W}_{i,\text{lowest\_active}}(t_a))$;
        Determine if the line $(a, b)$ intersects with $f(t) = t$ ;
        Let this point of intersection be $t'$;
        if $t' > (lowest\_active - 1)T_i + D_i$ then return "not feasible" else lowest_active=lowest_active+1 ;
    end
    $x = \max\left(0, \left\lceil \frac{t_a}{T_i} \right\rceil - Z_i(t)\right)$;
    lowest_active $= \max(x + 1, \text{lowest\_active})$;
end
return lowest_active;
```

where $Z_i(t)$ determines the number of jobs $l \leq \left\lceil \frac{t}{T_i} \right\rceil$ such that $\widetilde{W_i}(t) > t$ is defined as follows:

$$Z_i(t) = \max\left(\left\lceil \frac{\widetilde{W_i}(t) - t}{C_i} \right\rceil, 0\right) \tag{8}$$

As a consequence, in every interval between two adjacent scheduling points in $\tilde{S}_i$, at most one job of τ_i is tested to be feasible or not. The complete algorithm of the ApproxFirstStage defined in [4] is presented in Algorithm 1.

Since the level-i busy period is not necessarily fully explored in the first stage (i.e., when the first stage returns lowest_active values), the feasibility status of the analysed tasks can be still open. To cope with that problem, a second stage is run to check in constant time the task using a linear approximation (i.e., a worst-case response time upper bound computed in linear time). The second stage takes as input the job index h returned by the first stage (i.e., the job request is not satisfied by time t).

The main algorithm first performs a call to the first stage. If it returns the index h of a job, then the second stage is called for checking in constant time if $\tau_{i,h}$ is feasible or not. Hereafter, we only detail the first stage since the original second stage remains unchanged in the revised algorithm.

3.2 Analysis of Fisher and Baruah's Algorithm

We first show that the Fisher and Baruah is not correct under arbitrary priority assignment. Then, we show that the algorithm is correct under the RM scheduling policy.

Problems in the general case. FB05 correctness is based on the fact that at most one job must be checked between two subsequent scheduling points of the testing set $\tilde{S}_i$. We shall see that is not true using a counter-example. In the same way, the function defined the number of jobs for completion at each scheduling point in the approximate test (i.e., Equation 6 in [4]) need also to be modified to correct the proofs (i.e., Lemma 4 in [4] cannot be proved correct using Equation 6).

We think that the mentioned problems cannot be easily corrected without modifying the formulas presented in the previous section and also the corresponding proofs must be revisited for proving the correctness of the revised algorithm. Nevertheless, we shall prove that through these formula modifications, all these problems can be fixed while preserving the main ideas of the algorithm. We modified the previous functions to enforce the Fisher and Baruah's Lemma 4 correctness. Furthermore, we improve the algorithm by introducing in the first stage a test for checking if the level-i busy period is "approximately" completed or not. Such a modification will decrease the computational effort for checking the feasibility of the analysed task.

We first prove that the observation which is the base for the first stage of the approximation scheme proposed in [4] is not correct for non-RM priority assignments: several jobs of the analysed task can complete between two subsequent scheduling points.

Theorem 1 *The observation "For any two adjacent elements $t_1, t_2 \in \widehat{S}_i, |t_2 - t_1| \leq T_i$. Therefore, at most one job of τ_i can have its deadline occur between any 2 adja-*

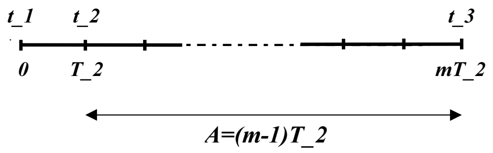

Figure 1. The interval between two adjacent elements in $\widehat{S}_i$ can be greater than T_i

cent elements of $\widehat{S}_i$" is wrong.

Proof: Let's consider the counter example below which is also illustrated in the Figure 1:

$\tau_i = \{\tau_1, \tau_2\}, Prio(\tau_1) > Prio(\tau_2)$

$T_1 = mT_2$

$k = 2$ where k is the number of steps before the approximation.

From Eq. (5) we have

$$\begin{aligned}\widehat{S_2} &\overset{\text{def}}{=} \{t = bT_a \mid a = 1, \ldots, 2, b = 1, \ldots, 1\} \bigcup \{t = 0\} \\ &= \{t_1 = 0, t_2 = T_1, t_3 = T_2\}\end{aligned}$$

Note that zero is only defined for some simplifying proof statements. Let's consider t_2, t_3 which are adjacent in the set $\widehat{S_2}$, we have:

$|t_2 - t_3| = |T_1 - T_2| = |(m-1)T_2| = A$

Increasing m, we obviously can make $A > T_2$ and by choosing appropriately the relative deadline $D_2 < T_2$, $m-1$ deadlines of τ_2 can elapse between t_2 and t_3 which contradicts the statement mentioned. ■

We now will prove that using the definition Eq. (8) is not correct for establishing Lemma 4 in [4].

Theorem 2 *The statement "'If $l > \left\lceil \frac{t}{T_i} \right\rceil - Z_i(t)$ and $t \geq (l-1)T_i$, then $\widehat{W}_{i,l}(t) > t$"' is not correct.*

Proof: We prove this by giving a counter-example in which 3 conditions below are satisfied:

1. $t \in ((l-1)T_i, (l-1)T_i + D_i]$
2. $l = \left\lceil \frac{t}{T_i} \right\rceil - Z_i(t) + 1$
3. $\widehat{W}_{i,l}(t) \leq t$

First, we choose an arbitrary integer k strictly greater than two to be the number of steps before approximation k.

Then, we choose the task system as follows (every task is presented under the format $\tau_i =< C_i, D_i, T_i >$):

$\tau_i = \{\tau_1, \tau_2\}, Prio(\tau_1) > Prio(\tau_2)$

$\tau_2 =< (k+1)\alpha T_2, kT_2, T_2 >$ with an arbitrary real number α such that $0 < \alpha < \frac{1}{k+2}$

$\tau_1 =< k(1-(k+2)\alpha)T_2, D_1, kT_2 >$

Then we choose the time instant $t = T_1 = kT_2$. Note that $t > (k-1)T_2$ ($\Rightarrow \delta(\tau_2, t) = (t+T_2)U_2$) and that since $k > 2 \Rightarrow t < (k-1)kT_2 = (k-1)T_1$ ($\Rightarrow \delta(\tau_1, t) = \text{RBF}(\tau_1, t) = \left\lceil \frac{t}{T_1} \right\rceil C_1$). Finally, we choose $i = 2$ and we now prove condition 1:

$$\begin{aligned}\widetilde{W_2}(t) &= \delta(\tau_2, t) + \sum_{j<2} \delta_j(t) \\ &= \delta(\tau_2, t) + \delta(\tau_1, t) \\ &= (t + T_2)U_2 + \left\lceil \frac{t}{T_1} \right\rceil C_1 \\ &= (kT_2 + T_2)(k+1)\alpha + k(1-(k+2)\alpha)T_2 \\ &= kT_2 + \alpha T_2 > kT_2 = t.\end{aligned}$$

So $\widetilde{W_2}(t) > t \Rightarrow max\left(\left\lceil \frac{\widetilde{W_2}(t)-t}{C_2} \right\rceil, 0\right) = Z_2(t) \geq 1 \Rightarrow$ If we choose $l = \left\lceil \frac{t}{T_2} \right\rceil - Z_2(t) + 1$ then $l \leq \left\lceil \frac{t}{T_2} \right\rceil - 1 + 1 = \left\lceil \frac{t}{T_2} \right\rceil = \left\lceil \frac{kT_2}{T_2} \right\rceil = k \Rightarrow (l-1) < k \Rightarrow$
$t = kT_2 > (l-1)T_2$ (a). Moreover,
$t = kT_2 \leq (l-1)T_2 + kT_2 = (l-1)T_2 + D_2$ (b)

From (a), (b), we have condition 1 is satisfied.

Condition 2 is obviously satisfied by the construction of l.

Now we consider the last condition:

$$\begin{aligned}\widehat{W}_{i,l}(t) &= lC_2 + \sum_{j<2} \delta_j(t) \\ &= \left(\left\lceil \frac{t}{T_2} \right\rceil - \left\lceil \frac{\widetilde{W_2}(t) - t}{C_2} \right\rceil + 1\right) C_2 + \delta(\tau_1, t) \\ &\leq \left(\left\lceil \frac{t}{T_2} \right\rceil - \frac{\widetilde{W_2}(t) - t}{C_2} + 1\right) C_2 + \delta(\tau_1, t) \\ &= \left\lceil \frac{t}{T_2} \right\rceil C_2 + C_2 - \widetilde{W_2}(t) + \delta(\tau_1, t) + t \\ &= \left\lceil \frac{t}{T_2} \right\rceil C_2 + C_2 - \delta(\tau_2, t) + t = RHS\end{aligned}$$

Replace $t = kT_2$ we have:

$$\begin{aligned}RHS &= (k+1)C_2 - (kT_2 + T_2)\frac{C_2}{T_2} + t \\ &= t\end{aligned}$$

Condition 3 is proved. hence we have the counter example of the Lemma 4 in [4].

■

Correctness under RM. We will prove that even Rate Monotonic priority assignment (i.e., $\forall i \in [1, n] :: (T_1 \leq T_2 \leq \ldots \leq T_i)$) the observation that only a single deadline elapse between adjacent testing set points is correct. However, Lemma 4 in [4] still requires some refinement for ensuring its correctness. We shall see later that a simple additional assumption leads to a correct version of Lemma 4 allowing to use the result into the algorithm ApproxFirstStage.

Theorem 3 *The statement "If $l > \left\lceil \frac{t}{T_i} \right\rceil - Z_i(t)$ and $t > (l-1)T_i$, then $\widehat{W}_{i,l}(t) > t$" is not correct.*

Proof: We prove this by giving a counter-example in which 3 conditions below are satisfied:

1. $t > (l-1)T_i$
2. $l = \left\lceil \frac{t}{T_i} \right\rceil - Z_i(t) + 1$
3. $\widehat{W}_{i,l}(t) \le t$

First, we choose an arbitrary positive integer k (to be the number of steps before starting linear approximations) and an arbitrary positive integer m.

Then, we choose the task system as follows:
$\tau_i = \{\tau_1, \tau_2\}, Prio(\tau_1) > Prio(\tau_2)$ hence $T_1 \le T_2$
$\tau_2 =< C_1, D_1, T >$
$\tau_1 =< C_2, D_2, T >$
where C_1, C_2, T are chosen such that:

$$\frac{k+m-1}{k+m} < \frac{C_1 + C_2}{T} = U_1 + U_2 = U < 1 \qquad (9)$$

Then, we choose the time instant $t = (k-1+m)T$. Note that $t > (k-1)T = (k-1)T_1 = (k-1)T_2$ ($\Rightarrow \delta(\tau_1, t) = (t+T_1)U_1$ and $\delta(\tau_2, t) = (t+T_2)U_2$).

Now we choose the task to be considered $i = 2$ and $l = \left\lceil \frac{t}{T_i} \right\rceil - Z_i(t) + 1 = \left\lceil \frac{t}{T_2} \right\rceil - Z_2(t) + 1$ (hence by construction, condition 2 is satisfied).

Now, we will prove condition 1:
We have:

$$\begin{aligned}
\widetilde{W_2}(t) &= \delta(\tau_2, t) + \sum_{j<2} \delta_j(t) \\
&= \delta(\tau_2, t) + \delta(\tau_1, t) \\
&= (t+T)U \\
&= ((k-1+m)T + T)U \\
&= ((k+m)T)U \\
&> ((k+m)T)\frac{k+m-1}{k+m} \text{ from Eq. (9)} \\
&> (k+m-1)T = t.
\end{aligned}$$

By definition, $Z_i(t) = \max\left\{\left\lceil \frac{\widetilde{W}_i(t)-t}{C_i} \right\rceil, 0\right\}$. As proved above $\widetilde{W_2}(t) > t \implies \left\lceil \frac{\widetilde{W_2}(t)-t}{C_2} \right\rceil \ge 1 \implies Z_2(t) = \left\lceil \frac{\widetilde{W_2}(t)-t}{C_2} \right\rceil \ge 1$. Since $l = \left\lceil \frac{t}{T_2} \right\rceil - Z_2(t) + 1$, we have:

$$\begin{aligned}
l &\le \left\lceil \frac{t}{T_2} \right\rceil \\
l &< \frac{t}{T_2} + 1 \\
(l-1)T_2 &< t.
\end{aligned}$$

Condition 1 is satisfied.
Now, we consider the last condition:

$$\begin{aligned}
\widehat{W}_{2,l}(t) &= lC_2 + \sum_{j<2} \delta_j(t) \\
&= \left(\left\lceil \frac{t}{T} \right\rceil - Z_2(t) + 1\right) C_2 + \delta(\tau_1, t) \\
&\le \left(\left\lceil \frac{t}{T} \right\rceil - \frac{\widetilde{W_2}(t) - t}{C_2} + 1\right) C_2 + \delta(\tau_1, t) \\
&= \left\lceil \frac{t}{T} \right\rceil C_2 + C_2 - \widetilde{W_2}(t) + \delta(\tau_1, t) + t \\
&= \left\lceil \frac{t}{T} \right\rceil C_2 + C_2 - \delta(\tau_2, t) + t = RHS
\end{aligned}$$

Replace $t = (k-1+m)T$, we have the RHS equals t which implies that $\widehat{W}_{2,l}(t) = t$ and Condition 3 is satisfied. This contradiction is then proved.

■

Thus, Lemma 4 is not correct for arbitrary t, but the statement holds for any $t \le \max_{j<i}\{(k-1)T_j\}$ and please notice that the algorithm ApproxFirstStage does not check for t beyond this value. Thus, we can show the algorithm is correct with a refinement of the supposition of the lemma for t in the range zero to $\max_{j<i}\{(k-1)T_j\}$.

Lemma 1 *If $l > \left\lceil \frac{t}{T_i} \right\rceil - Z_i(t)$, and $t \in ((l-1)T_i < t, \max_{j<i}\{(k-1)T_j\}]$ then $\widehat{W}_{i,l}(t) > t$.*

Proof: Let $a = \left\lceil \frac{t}{T_i} \right\rceil - Z_i(t) + 1 \implies l \ge a, \widehat{W}_{i,l}(t) \ge \widehat{W}_{i,a}(t)$

For any l, choose an arbitrary t such that $(l-1)T_i < t \le \max_{j<i}\{(k-1)T_j\} \implies (a-1)T_i < t \le \max_{j<i}\{(k-1)T_j\}$.

Since $t > (a-1)T_i \iff \frac{t}{T_i} > a - 1 \iff \left\lceil \frac{t}{T} \right\rceil \ge a$. From definition of a, we obtain $Z_i(t) \ge 1$. As a consequence, since $Z_i(t) = \max\left\{\left\lceil \frac{\widetilde{W}_i(t)-t}{C_i} \right\rceil, 0\right\}$, we obtain $Z_i(t) = \left\lceil \frac{\widetilde{W}_i(t)-t}{C_i} \right\rceil \ge 1$.

We have:

$$\begin{aligned}
\widehat{W}_{i,a}(t) &= aC_i + \sum_{j<i} \delta(\tau_j, t) \\
&= \left(\left\lceil \frac{t}{T_i} \right\rceil - Z_i(t) + 1\right) C_i + \sum_{j<i} \delta(\tau_j, t) \\
&> \left\lceil \frac{t}{T_i} \right\rceil C_i - (\widetilde{W}_i(t) - t) + \sum_{j<i} \delta(\tau_j, t) \\
&> \left\lceil \frac{t}{T_i} \right\rceil C_i - \widetilde{W}_i(t) + \sum_{j<i} \delta(\tau_j, t) + t \\
&> \left\lceil \frac{t}{T_i} \right\rceil C_i - \delta(\tau_i, t) + t = RHS
\end{aligned}$$

Under RM we have $\forall i \in [1,n] :: (T_1 \leq T_2 \leq \ldots \leq T_i)$. As a consequence, $t \leq \max_{j<i}\{(k-1)T_j\} \Longrightarrow t \leq (k-1)T_i \Longrightarrow \delta(\tau_i, t) = \text{RBF}(\tau_i, t) = \left\lceil \frac{t}{T_i} \right\rceil$. Replace this into RHS we have $\widehat{W}_{i,a}(t) > t$. The lemma is proved. ■

Thus, using this previous restated Lemma 4 and under the RM scheduling policy, the algorithm ApproxFirstStage is correct.

4 Revised version of FB05 algorithm

We show now that the algorithm can be modified in order to be applicable for non-RM priority assignment. These modifications are all in the first stage of the original algorithm to enforce the FPTAS correctness.

4.1 Preliminary results

Based on this approximate $\widehat{W}_{i,l}(t)$, we define, for each given job $\tau_{i,l}$, an approximate intersection instant $\widehat{R}_{i,l}$ as:

Definition 1 *Let $\widehat{R}_{i,l}$ denote the first intersection between $\widehat{W}_{i,l}(t)$ and $f(t) = t$, then $\widehat{R}_{i,l}$ can be defined by:*

$$\widehat{R}_{i,l} \overset{\text{def}}{=} min\left\{t \mid \widehat{W}_{i,l}(t) = t\right\} \tag{10}$$

Then we define $\hat{N}_i$ as the index of the last job in the first level-i busy period considered in our test:

Definition 2

$$\hat{N}_i \overset{\text{def}}{=} min\left\{l \in N \mid \widehat{R}_{i,l} \leq (l-1)T_i + T_i\right\} \tag{11}$$

Finally, we define, for a given task τ_i, the set of points constructing intervals to be tested in our approximation as:

$$\begin{aligned}\widetilde{S}_i \overset{\text{def}}{=} \quad & \{t = bT_a \mid a = 1, \ldots, i-1; \\ & b = 1, \ldots, k-1\} \bigcup \{0\} \bigcup \{\infty\}\end{aligned} \tag{12}$$

Zero and infinite points are only introduced to simplify some proof statements. Furthermore, the relative deadline D_i is not defined in the testing set but is directly exploited in the algorithm. We then define a *primitive interval* of some set as an interval $(t_{a-1}, t_a]$ bounded by two adjacent points t_{a-1} and t_a of that set.

From this section to the rest of this paper, in the context concerning the approximate algorithm (presented in the next subsections), we will use the term "a job $\tau_{i,l}$ is completed" to indicate "the approximate processor demand function $\widehat{W}_{i,l}(t)$ cuts the processor capacity function" (i.e., $f(t) = t$), the term "meet the deadline" to indicate "$\widehat{R}_{i,l} \in [(l-1)T_i, (l-1)T_i + D_i]$ ", the term "the intersection time" to indicate "the approximate intersection time", and so on.

Notice that $\widehat{R}_{i,l}$ is just the first instant when $\widehat{W}_{i,l}(t) = t$ and not necessarily the completion time of $\tau_{i,l}$ in the general case. But, these intersection points $\widehat{R}_{i,l}$ correspond to upper bounds on job completion times if they belong to the first level-i busy period, as stated in the following lemma:

Lemma 2

$$\forall i \leq n, l \leq \hat{N}_i, \widehat{R}_{i,l} > (l-1)T_i$$

First, we present the following properties of $\widehat{R}_{i,l}$

Lemma 3 $\forall i \leq n$, *let l, h two natural numbers such that $l < h$, then $\widehat{R}_{i,l} < \widehat{R}_{i,h}$*

Proof: By definition $\widehat{W}_{i,h}(\widehat{R}_{i,h}) = \widehat{R}_{i,h}$. Moreover, $l < h \Longrightarrow W_{i,l}(t) < W_{i,h}(t)$. Therefore:

$$\widehat{W}_{i,l}(\widehat{R}_{i,h}) < \widehat{R}_{i,h} \Longrightarrow \exists t^* < \widehat{R}_{i,h} :: (\widehat{W}_{i,l}(t^*) = t^*)$$

Since $\widehat{R}_{i,l} \overset{\text{def}}{=} \min\{t \mid W_{i,l}(t) = t\}$, then $\widehat{R}_{i,l} \leq t^*$ and as a consequence $\widehat{R}_{i,l} < \widehat{R}_{i,h}$. ■

Now we prove Lemma 2:

Proof: If $\hat{N}_i = 1$ then the lemma obviously holds as the first intersection is always after 0. If $\hat{N}_i > 1$ then by Definition 11, $\forall i \leq n, l \leq \hat{N}_i - 1, \widehat{R}_{i,l} > (l-1)T_i + T_i > (l-1)T_i$. From Lemma 3, we know that $\widehat{R}_{i,\hat{N}_i} > \widehat{R}_{i,\hat{N}_i - 1} = (\hat{N}_i - 1 - 1)T_i + T_i = (\hat{N}_i - 1)T_i$. The lemma is proved. ■

Hence, we have a necessary and sufficient condition for checking if a job of a task τ_i is approximately feasible as follows:

Theorem 4

$$l \leq \hat{N}_i :: \widehat{R}_{i,l} \leq (l-1)T_i + D_i$$

if, and only if,

$$\exists t \in ((l-1)T_i, (l-1)T_i + D_i] : \widehat{W}_{i,l}(t) \leq t$$

Proof: The "only" if direction statement is straightforward from Lemma 2 and the definition of $\widehat{R}_{i,l}$. For the "if" direction, let us choose an arbitrary $t^* \in ((l-1)T_i, (l-1)T_i + D_i] : \widehat{W}_{i,l}(t) \leq t$. As $\widehat{W}_{i,l}(t)$ is a non-decreasing left-continuous step function and $\widehat{W}_{i,l}(0) = lC_i > 0$, then $\exists t^{int} \leq t^* :: (\widehat{W}_{i,l}(t^{int}) = t^{int})$ From definition of $\widehat{R}_{i,l} \Longrightarrow \widehat{R}_{i,l} \leq t^{int} \leq t^* \Longrightarrow \widehat{R}_{i,l} \leq (l-1)T_i + D_i$. Using Lemma 2, we have $\widehat{R}_{i,l} > ((l-1)T_i$ and thus $\widehat{R}_{i,l} \in ((l-1)T_i, (l-1)T_i + D_i]$. The theorem is proved. ■

4.2 Properties of primitive interval

Now we can use $\widehat{R}_{i,l}$ to verify, in the context that the first level-i busy period has not been terminated, if $\tau_{i,l}$ meet its deadline or not.

Based on the results presented in the previous section, we will propose in the next sections an algorithm that verifies the feasibility of τ_i by analyzing all the intersection instant $\widehat{R}_{i,l}$ located in a given primitive intervals of $\widetilde{S}_i$. In order to reduce this complexity, now we are looking for some properties allowing for testing the feasibility of the first job in each primitive interval. Precisely, the first job of τ_i will have a longer upper bound on the response time than any other response time of τ_i that is computed from an $\hat{R}_{i,j}$ in the same primitive interval.

First of all, we prove the following property of all jobs of a task τ_i that completed in the same primitive interval of $\widetilde{S}_i$:

Lemma 4 *For any $i \leq n$, l and h two natural numbers such that $l < h$ and any two adjacent scheduling points $t_1 < t_2$ in $\widetilde{S}_i$ such that $\widehat{R}_{i,l}, \widehat{R}_{i,h} \in (t_1, t_2]$ then $\widehat{R}_{i,h} < \widehat{R}_{i,l} + (h-l)T_i$.*

Proof: Let $\mathbf{T}$ denote the set of all tasks τ_j such that $j < i$ and $(k-1)T_j \leq t_1$. In other words, $\mathbf{T}$ is the set of all tasks whose the approximate RBF becomes a linear function after t_1. Note that $\mathbf{T} \subseteq 1..i-1$.

We can rewrite $\widehat{W}_{i,l}(t)$ in $(t_1, t_2]$ as follows:

$$\begin{aligned}
\widehat{W}_{i,l}(t) &= lC_i + \sum_{j<i} \delta(\tau_j, t) \\
&= lC_i + \sum_{j<i, j \notin \mathbf{T}} \text{RBF}(\tau_j, t) + \sum_{j \in \mathbf{T}} (t + T_j)U_j \\
&= lC_i + \sum_{j<i, j \notin \mathbf{T}} \text{RBF}(\tau_j, t) + \sum_{j \in \mathbf{T}} T_j U_j + t \sum_{j \in \mathbf{T}} U_j
\end{aligned}$$

$$\begin{aligned}
\text{Let } A &= lC_i + \sum_{j<i, j \notin \mathbf{T}} \text{RBF}(\tau_j, t) + \sum_{j \in \mathbf{T}} T_j U_j, \\
B &= \sum_{j \in \mathbf{T}} U_j.
\end{aligned}$$

Then we have:

$$\widehat{W}_{i,l}(t) = A + Bt$$

Note that at all instant $t \in (t_1, t_2]$, A and B are constant.

Since from definition of $W_{i,l}(t)$, $\widehat{W}_{i,h}(t) = W_{i,l}(t) + (h-l)C_i \Longrightarrow$ we obtain

$$\widehat{W}_{i,h}(t) = (h-l)C_i + A + Bt$$

Now we compute $\widehat{R}_{i,l}, \widehat{R}_{i,h}$ according to these new formulas of $\widehat{W}_{i,l}(t)$ and $\widehat{W}_{i,h}(t)$.

As $\widehat{R}_{i,l}$ is the intersection of $\widehat{W}_{i,l}(t)$ with $y = t$ in $(t_1, t_2]$

$$\Longrightarrow \widehat{R}_{i,l} = \frac{A}{1-B}$$

.

Similarly,

$$\widehat{R}_{i,h} = \frac{A + (h-l)C_i}{1-B}$$

Hence we have:

$$\begin{aligned}
\widehat{R}_{i,h} - \widehat{R}_{i,l} &= \frac{A + (h-l)C_i}{1-B} - \frac{A}{1-B} \\
&= (h-l)\frac{C_i}{1-B} \\
&= (h-l)T_i \frac{U_i}{1-B} = RHS
\end{aligned}$$

Replacing $B = \sum_{j \in \mathbf{T}} u_j$, we have:

$$\begin{aligned}
1 - B &= 1 - \sum_{j \in \mathbf{T}} U_j \\
&\geq 1 - \sum_{j<i} U_j \\
&> \sum_{1 \leq j \leq n} U_j - \sum_{j<i} U_j \\
&> U_i
\end{aligned}$$

Hence we obtain:

$$\frac{U_i}{1-B} < 1$$

$\Longrightarrow RHS < (h-l)T_i$. The lemma is proved. ■

From the lemma above, we obtain the main property about response time of jobs in a primitive interval: given two adjacent points t_1, t_2 of $\widetilde{S}_i$, provided that the level-i busy period have not been terminated before t_1, then the next theorem implies:

1. To verify if any jobs completed in $(t_1, t_2]$ have met their deadlines, we just need to check the job having been released sooner in the interval $(t_1, t_2]$.

2. To verify if between those 2 jobs there exists a job that is completed before the activation of its subsequent instance, it is sufficient to check the job having been released later. This latter property allows to check if the level-i busy period is "approximately" completed or not.

Theorem 5 *For any $i \leq n, l, h \in N, l < h$ and any 2 adjacent points $t_1 < t_2$ in $\widetilde{S}_i$ such that $\widehat{R}_{i,l}, \widehat{R}_{i,h} \in (t_1, t_2]$ then we have:*

$$\widehat{R}_{i,l} \leq (l-1)T_i + D_i \Longrightarrow \widehat{R}_{i,h} \leq (h-1)T_i + D_i, \tag{13a}$$

$$\widehat{R}_{i,h} > (h-1)T_i + T_i \Longrightarrow \widehat{R}_{i,l} > (l-1)T_i + T_i \tag{13b}$$

Proof: First, we prove Eq. (13a):

$$\widehat{R}_{i,l} \leq (l-1)T_i + D_i$$

$$\widehat{R}_{i,h} - (h-l)T_i \leq (l-1)T_i + D_i$$

(for $\widehat{R}_{i,h} - (h-l)T_i < \widehat{R}_{i,l}$ from Lemma 4)

$$\widehat{R}_{i,h} \leq (h-1)T_i + D_i$$

Now we prove Eq. (13b):

$$\widehat{R}_{i,h} > (h-1)T_i + T_i$$
$$\widehat{R}_{i,l} + (h-l)T_i > (h-1)T_i + T_i$$
$$\text{(for } \widehat{R}_{i,h} < \widehat{R}_{i,l} + (h-l)T_i \text{ from Lemma 4)}$$
$$\widehat{R}_{i,l} > (l-1)T_i + T_i$$

The theorem is proved. ■

The following result follows directly from Theorem 5 for the necessary condition (the sufficient condition is obvious):

Corollary 1 *$\forall t_1, t_2 \in \widetilde{S}_i$, Let first and last respectively be defined as:*

$$first \overset{\text{def}}{=} \min\left\{l \in N \mid \widehat{R}_{i,l} \in (t_1, t_2]\right\}$$
$$last \overset{\text{def}}{=} \max\left\{l \in N \mid \widehat{R}_{i,l} \in (t_1, t_2]\right\}$$

then the following inequalities are always satisfied:

$$\widehat{R}_{i,first} \le (first-1)T_i + D_i \Longleftrightarrow \tag{14a}$$
$$\forall l \in [first, last] : \widehat{R}_{i,l} \le (l-1)T_i + D_i, \tag{14b}$$
$$\widehat{R}_{i,last} > (last-1)T_i + T_i \Longleftrightarrow \tag{14c}$$
$$\forall l \in [first, last] : \widehat{R}_{i,l} > (l-1)T_i + T_i \tag{14d}$$

From the corollary above, we obtain that in each primitive interval there might be a pseudo-polynomial number of jobs of τ_i having been completed, but provided that the first level-i busy period have not already been terminated in previous primitive intervals, then we just need to check only the first job against its deadline to conclude the feasibilities of all these jobs. Moreover, in order to verify if the level-i busy period has been terminated in the analysed primitive interval, it is sufficient to check only the last job against the activation of its subsequent instance. As a consequence, in every primitive interval, if the last job of τ_i is completed before the next release of τ_i, then it is unnecessary to test the subsequent primitive intervals (i.e., the level-i busy period has been completed in the current primitive interval).

Finally, we are looking for some method to identify the indexes of the first and the last job having been completed in a given primitive interval of $\widetilde{S}_i$.

For this purpose, we first introduce the following function that is used to determine the set of jobs that have their approximate execution requests satisfied by any time instant t,

$$\widehat{W}_i(t) \overset{\text{def}}{=} \text{RBF}(\tau_i, t) + \sum_{j<i} \delta(\tau_j, t) \tag{15}$$

$\widehat{W}_i(t)$ represents a "close" approximate to the cumulative requests of task τ_i and all higher priority tasks with respect to *any* given time t. In comparison , $\widehat{W}_{i,l}(t)$ is only a "close" approximation when t lies in the interval $((l-1)T_i, lT_i]$. Please note that the function corresponds to the modification of Equation 6 (i.e. Equation 13 in [4]).

Notice that the number of *active* jobs at time t would be $\mathcal{O}(D_i/T_i)$. The next function is used to identify the index of the most recently released job of τ_i to have its execution request approximately satisfied by time t.

$$I_i(t) \overset{\text{def}}{=} \left\lceil \frac{t}{T_i} \right\rceil - \left\lceil \frac{\widehat{W}_i(t) - t}{C_i} \right\rceil \tag{16}$$

$I_i(t)$ is based on a modification of the function $Z_i(t)$ defined in [4].Precisely, we removed the maximum function in order to prove the two next lemmas. If the value of $I_i(t)$ is not negative then it is used to determine the index of the most recently released job to have its execution request satisfied (according to the approximation scheme) by time t. By finding the index of the most recently released job of τ_i to complete execution with respect to time t, we can determine in constant time the set of active jobs that have their execution requests satisfied at or prior to t.

Lemma 5 *If $I_i(t) > 0$ then $\forall l \in \{1, \dots, I_i(t)\}, \widehat{W}_{i,l}(t) \le t$*

Proof:
Let $b = I_i(t) \Longrightarrow l \le b, \widehat{W}_{i,l}(t) \le \widehat{W}_{i,b}(t)$
We have:

$$\begin{aligned}
\widehat{W}_{i,b}(t) &= bC_i + \sum_{j<i} \delta(\tau_j, t) \\
&= \left(\left\lceil \frac{t}{T_i} \right\rceil - \left\lceil \frac{\widehat{W}_i(t) - t}{C_i} \right\rceil \right) C_i + \sum_{j<i} \delta(\tau_j, t) \\
&= \left\lceil \frac{t}{T_i} \right\rceil C_i - \left\lceil \frac{\widehat{W}_i(t) - t}{C_i} \right\rceil C_i + \sum_{j<i} \delta(\tau_j, t) \\
&\le \left\lceil \frac{t}{T_i} \right\rceil C_i - \frac{\widehat{W}_i(t) - t}{C_i} C_i + \sum_{j<i} \delta(\tau_j, t) \\
&\le \left\lceil \frac{t}{T_i} \right\rceil C_i + \sum_{j<i} \delta(\tau_j, t) - \widehat{W}_i(t) + t \\
&\le \widehat{W}_i(t) - \widehat{W}_i(t) + t = t
\end{aligned}$$

The lemma is proved. ■

Lemma 6 *If, $t \in ((l-1)T_i, (l-1)T_i + D_i]$ and $l > I_i(t)$ then $\widehat{W}_{i,l}(t) > t$*

Proof: Let $a = I_i(t) + 1 \Longrightarrow l \ge a, \widehat{W}_{i,l}(t) \ge \widehat{W}_{i,a}(t)$
We have:

$$
\begin{aligned}
\widehat{W}_{i,a}(t) &= aC_i + \sum_{j<i} \delta(\tau_j, t) \\
&= \left(\left\lceil \frac{t}{T_i} \right\rceil - \left\lceil \frac{\widehat{W}_i(t) - t}{C_i} \right\rceil + 1 \right) C_i + \sum_{j<i} \delta(\tau_j, t) \\
&> \left\lceil \frac{t}{T_i} \right\rceil C_i - (\widehat{W}_i(t) - t) + \sum_{j<i} \delta(\tau_j, t) \\
&> \left\lceil \frac{t}{T_i} \right\rceil C_i + \sum_{j<i} \delta(\tau_j, t) - \widehat{W}_i(t) + t \\
&> \widehat{W}_i(t) - \widehat{W}_i(t) + t = t
\end{aligned}
$$

The lemma is proved. ■

Note that this function $I_i(t)$ cannot be applicable for $t = \infty$. Besides, for the last primitive interval of $\widetilde{S}_i$, which has ∞ as its right bound, it is also unnecessary to verify if the first level-i busy period has been terminated within or not (since $U < 1$, the level-i busy period will complete before time ∞). Therefore, we divide all the jobs of τ_i into two sets: one for jobs completed prior or equal to $\max_{j \in 1,\ldots,i-1} \{(k-1)T_j\}$ (i.e., completed in some primitive intervals of $\widetilde{S}_i$ that is not the last one) and one for jobs completed in $(\max_{j \in 1,\ldots,i-1} \{(k-1)T_j\}, \infty)$ (i.e., completed in the last primitive interval of $\widetilde{S}_i$). In this latter case, all approximate request bound functions are linear functions. Due to these remarks, the algorithm is still divided into two stages as in [4].

4.3 First Stage: Job with completion time prior or equal to $\max_{j \in 1,\ldots,i-1} \{(k-1)T_j\}$

We construct an algorithm which determines, for any 2 adjacent points $t_1, t_2 \in \widetilde{S}_i \ \{\infty\}$, which jobs of τ_i have their execution requests satisfied in $(t_1, t_2]$. Also, we can check in constant time whether all these jobs respect their respective deadlines as well as whether the level-i busy period has been terminated in that interval. The complete algorithm is presented in Algorithm 2.

We summarized the main differences between the revised version and the original version:

- The following functions have been changed: $Z_i(t)$ and $\widetilde{W}_i(t)$,

- the FBApproxFirstStage checks if a job has its deadline in the current primitive interval. But, it does not check if the job completes or not within the interval. Note that in the revised version, the $I_i(t)$ function is computed at the beginning of each iteration and compared with the value of the previous iteration (i.e., last_active value), whereas it is called at the end of the FBApproxFirstStage algorithm. Thus, in the revised version, it is checked if there is at least one job completed in the current primitive interval.

4.4 Second Stage: Job with completion time after $\max_{j \in 1,\ldots,i-1} \{(k-1)T_j\}$

The second stage is not modified in comparison with the original second stage algorithm. Please consult [4] for details.

4.5 Detailed example

We illustrate the approximation scheme proposed in this paper by considering the task set presented in [6]: $\tau_1(C_1 = 26, D_1 = 40, T_1 = 70)$ and $\tau_2(C_2 = 62, D_2 = 140, T_2 = 100)$. Using Deadline Monotonic, we assign to τ_1 the higher priority. We now consider the feasibility of τ_2 using both exact and approximation methods. The exact response time analysis stops after considering 7 jobs in the level-i busy period: for the $\tau_{2,7}$ the response time is $R_{2,7} = 96 < 100 = T_2$ (see [6] for details). Since all jobs meet their deadline (i.e., $\widehat{R}_{i,l} \leq (l-1)T_2 + D_2$) then τ_2 is stated feasible.

Now, we analyze the feasibility of τ_2 using the approximation scheme. We choose $\epsilon = 0.25$ thus $k = 3$ and $\widetilde{S}_i = \{0, 70, 140, \infty\}$. The two stages of the Approximate analysis will be executed. In the first stage we consider jobs completed prior or at $t = 140$ by using ApproxFirstStage.

- Before the first iteration of the loop in ApproxFirstStage, $last\_active = 0$.

- In the first iteration we consider the first primitive interval $(0, 70], t_a = t_1 = 70$. Since $I_2(70) = \left\lceil \frac{70}{100} \right\rceil - \left\lceil \frac{\widehat{W}_2(70)-70}{62} \right\rceil = 0 = last\_active$ we jump to the next iteration. $last\_active$ remains 0.

- Now we consider $(70, 140], t_a = t_2 = 140$. Since $I_2(140) = \left\lceil \frac{140}{100} \right\rceil - \left\lceil \frac{\widehat{W}_2(140)-140}{62} \right\rceil = 1 > last\_active$ we set $first = last\_active + 1 = 0 + 1 = 1, next = I_2(140) = 1$. Computing the instant in the interval $(70, 140]$ where $f(t) = t$ intersects $\widehat{W}_{2,first}(t)$, we obtain $t' = \widehat{R}_{2,first} = \widehat{R}_{2,1} = 114 < (1-1).100 + 140 \Longrightarrow$ There exists no job completed in this interval that has violated its deadline. Computing the instant in the interval $(70, 140]$ where $f(t) = t$ intersects $\widehat{W}_{2,last}(t)$, we obtain $t'' = \widehat{R}_{2,last} = \widehat{R}_{2,1} = 114 > (1-1).100 + 100 \Longrightarrow$ The first level-i busy period have not been terminated yet in this interval. We assign $last\_active = last = 1$ then we continue testing.

- As there is no more t_a to test, we return $last\_active + 1 = 1 + 1 = 2$.

Now we turn to consider the jobs completed after $t = 140$ using ApproxSecondStage. h used in this stage is the return value of the ApproxFirstStage hence $h = 2 \Longrightarrow \widehat{R}_{i,h} = \widehat{R}_{2,2} = 238, (63) < (2-1).100 + 140$. Therefore, we conclude τ_2 *is feasible*.

Algorithm 2. RevisedApproxFirstStage

```
input :
        $\tau = C[n], T[n], D[n]$ : array of integers          /* Task parameters */
        $i$ : integer                                          /* Index of the analysed task */
        $k$ : integer                                          /* The FPTAS accuracy parameter */
output: lowest_active : integer                                /* Lowest active job */

Construct $\tilde{S}_i$ using Equation 12                      /* Ordered set of scheduling points */
last_active=0 ;
foreach $t_a \in \tilde{S}_i - \{0\} - \{\infty\}$ do
    last=$I_i(t_a)$;
    if last>last_active then                                   /* a job completes in $(t_{a-1}, t_a]$ */
        first=last_active+1;
        /* Check the deadline of the first job of the current primitive interval
        */
        Let $t_{a-1}$ be the adjacent element prior to $t_a$ in $\tilde{S}_i$;
        Let $y$ be the total execution requirement of all jobs of priority greater than $\tau_i$ released at time $t_{a-1}$;
        $a = (t_{a-1}, \widehat{W}_{i,\text{first}}(t_{a-1}) + y)$ ;
        $b = (t_a, \widehat{W}_{i,\text{first}}(t_a))$;
        Determine if the line $(a, b)$ intersects with $f(t) = t$ ;
        Let this point of intersection be $t'$;
        if $t' > (first - 1)T_i + D_i$ then return "not feasible";
        /* Check if the level-$i$ busy period is completed in the current
        primitive interval (last job in the interval)                          */
        $c = (t_{a-1}, \widehat{W}_{i,\text{last}}(t_{a-1}) + y)$ ;
        $d = (t_a, \widehat{W}_{i,\text{last}}(t_a))$;
        Determine if the line $(c, d)$ intersects with $f(t) = t$ ;
        Let this point of intersection be $t''$;
        if $t'' \leq (last - 1)T_i + T_i$ then return "feasible";
        $last\_active := last$;
    end
end
return last_active;
```

5 Conclusions

We revised the approximation scheme presented in [4] to achieve its correctness for arbitrary priority assignment. We prove that the original algorithm is valid for RM. We then propose several formula modifications, a complete revised version of the algorithms for improving its efficiency. We already improve that function in [7] for approximating worst-case response time of static priority tasks with constrained-deadlines. We want to extend the feasibility test presented here to compute approximate worst-case response time with guaranteed resource augmentation ratio.

References

[1] E. Bini, T. Nguyen, P. Richard, and S. Baruah. A response-time bounds in fixed-priority scheduling with arbitrary deadlines. *IEEE Transactions on Computers (TOC'09)*, 58(2):279–286, feb 2009.

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

[3] F. Eisenbrand and T. Rothvoss. Static-priority real-time scheduling: Response time computation is np-hard. *Real-Time Systems Symposium (RTSS'08)*, 2008.

[4] 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.

[5] 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 and Network Systems (RTNS'05)*, pages 233–249, 2005.

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

[7] T. H. C. NGuyen, P. Richard, and E. Bini. Approximation techniques for response-time analysis of static-priority tasks. *Real-Time Systems*, 43:147–176, 2009.