Improvements in Polynomial-Time Feasibility Testing for EDF

Alejandro Masrur   Sebastian Drössler   Georg Färber
Institute for Real-Time Computer Systems
Technische Universität München, Germany
{Alejandro.Masrur, Sebastian.Droessler, Georg.Faerber}@rcs.ei.tum.de

Abstract

This paper presents two fully polynomial-time sufficient feasibility tests for EDF when considering periodic tasks with arbitrary deadlines and preemptive scheduling on uniprocessors. Both proposed methods are proven, analytically and by means of an extensive experimental comparison, to be more accurate than known polynomial-time feasibility tests. Additionally, we show for a wide interval of practical processor utilization that one of these methods presents almost the same efficiency, in terms of accepted task sets, as the more complex pseudo-polynomial-time exact feasibility tests.

1. Introduction

For the case where periodic tasks with arbitrary deadlines are preemptively scheduled under EDF (Earliest Deadline First) on a uniprocessor, exact feasibility tests with pseudo-polynomial complexity are already known, e.g. [2] and [8]. However, faster polynomial-time sufficient tests are still preferable for some applications where, for example, schedulability has to be tested online. Furthermore, the more predictable running time of polynomial-time algorithms is an additional advantage in the latter context.

In this paper, we show how it is possible to achieve more accuracy in polynomial-time feasibility testing for EDF under the given conditions. We propose two feasibility tests basing actually on same idea as the density condition [10], [7] and as Devi's condition [4], i.e. to reduce algorithms' complexity by sacrificing exactness.

The next two sections provide a survey of related work and a description of the used task model. Posteriorly, the tight relation between Devi's condition and the best known feasibility bound is pointed out in section 4. In section 5, we prove analytically that a slight, but for the purpose of this paper profitable, improvement of the best known feasibility bound is possible. Section 6 presents two better fully polynomial-time feasibility tests for EDF; one of them exhibits complexity $O(n^2)$ while the other has same complexity as Devi's condition, namely $O(n \log n)$. Additionally, section 7 outlines results from thorough experiments comparing our proposed methods against the best known feasibility tests for EDF. Finally, some concluding remarks are presented in section 8.

2. Related work

In [6], Liu and Layland showed that a feasibility test for synchronous tasks, scheduled preemptively under EDF and on a uniprocessor, can be performed as follows in polynomial time, when deadline (d_i) is equal to period (p_i) for every task: $U = \sum_{i=1}^{n} \frac{e_i}{p_i} \leq 1$. In this inequality, e_i is the worst-case execution time of the i-th task, n is the number of tasks and U is called processor utilization. Afterwards in [2], Baruah et al. proved that applying the utilization test of Liu and Layland is also valid when $d_i \geq p_i$ holds for all i.

When deadlines are allowed to be less than periods, the complexity grows considerably. However, assuming the processor utilization to be less than 100% for a synchronous scheduling, Baruah et al. also proved that if a deadline is missed, this happens before a maximum time upper limit known as *feasibility bound*. This result allowed Baruah et al. to design a pseudo-polynomial time algorithm for the case where deadlines are not forced to be equal to periods.

Another pseudo-polynomial time algorithm for $d_i \leq p_i$ was presented by Ripoll et al. in [8]. Ripoll et al. introduced two better feasibility bounds, which they combined in an efficient algorithm. On the other hand, George et al. considered in [5] also the case $d_i > p_i$ and got a feasibility bound, referred as George's bound in this paper, with the expression: $I_n = \frac{\sum_{i=1}^{n}(p_i - min(d_i, p_i)) \cdot u_i}{1-U}$, where $u_i = \frac{e_i}{p_i}$. George's bound reduces to Ripoll's bound if $d_i \leq p_i$ holds for all possible i and it is the best known feasibility bound for EDF in case that no restrictions are imposed to d_i.

The other feasibility bound presented by Ripoll et al. is based on the busy period analysis, whose calculation itself presents pseudo-polynomial complexity. This latter pseudo-polynomial feasibility bound was also independently obtained by Spuri [9].

All exact algorithms, including the one of Albers and Slomka [1], present pseudo-polynomial complexity. In order to reach polynomial complexity in feasibility testing for EDF, when d_i can be less than p_i, exactness must be sacrificed. Based on this idea, Liu in [7] and Stankovic et al. in [10] propose independently the density test, which results by replacing p_i by $min(d_i, p_i)$ in the utilization test of Liu and Layland, that is: $\sum_{i=1}^{n} \frac{e_i}{min(p_i,d_i)} \leq 1$.

Assuming that tasks are sorted by non-decreasing deadlines d_i, Devi presented in [4] a better approach in terms of accepted task sets than the density condition. However, Devi's condition describes a higher complexity $O(n \log n)$ because of requiring sorted task sets.

3. Task model and notation

In this section, we specify some more details about the task model and notation already used for related work.

We consider a set τ of periodic real-time tasks, which are fully preemptable and independent. As already mentioned, we assume that τ is scheduled on a uniprocessor. Each task T_i in τ is characterized by its period of repetition p_i, its relative deadline d_i and its worst-case execution time e_i. Another parameter of tasks is the initial release time or phase ϕ_i. For this paper, it is assumed that $\phi_i = 0$ holds for all tasks, i.e. τ is a synchronous task set. Relative deadlines d_i are not restricted and they can be less as well as greater than the respective periods p_i for $1 \leq i \leq n$, where n is the number of tasks in τ.

For remaining sections, we assume that the total processor utilization given by τ is less than 100%, i.e. $U < 1$ holds. In addition, the time is represented by the real-valued $t \geq 0$, whereas George's bound for τ is denoted by I_n.

4. Devi's condition and George's bound

The relation between Devi's algorithm and the best known feasibility bound was indirectly shown in [1]. However, the following lemma formulates this relation in a more explicit way.

LEMMA 1 *Assuming that tasks in τ are sorted according to non-decreasing relative deadlines, Devi's condition is equivalent to an iterative calculation of George's feasibility bound. The task set τ is feasible, if George's bound for the first k tasks is less than or equal to d_k for every possible k, where $1 \leq k \leq n$.*

Proof: It is assumed that tasks are sorted according to non-decreasing relative deadlines, so if $i < j$ holds, $d_i \leq d_j$ will also hold. Devi's condition states that τ is feasible if the following inequality holds for every possible k, where $1 \leq k \leq n$ [4]:
$\sum_{i=1}^{k} \frac{e_i}{p_i} + \frac{1}{d_k} \sum_{i=1}^{k} \left(\frac{p_i - min(p_i,d_i)}{p_i} \right) \cdot e_i \leq 1$.

Reordering terms, we get $\sum_{i=1}^{k} \left(\frac{p_i - min(p_i,d_i)}{p_i} \right) \cdot e_i \leq \left(1 - \sum_{i=1}^{k} \frac{e_i}{p_i} \right) \cdot d_k$, and finally:
$\frac{\sum_{i=1}^{k} (p_i - min(p_i,d_i)) \cdot u_i}{1 - U_k} \leq d_k$.

Where $U_k = \sum_{i=1}^{k} \frac{e_i}{p_i}$ and $u_i = \frac{e_i}{p_i}$, i.e. the left side of this latter inequality is George's bound I_k for the first k tasks. The thesis follows. □

Lemma 1 suggests that improving the best known feasibility bound also results in better polynomial-time feasibility tests. This aspect is analyzed in the next sections.

5. Improving George's bound

In this section, we prove that it is possible to reach a slight but anyway useful improvement of the best known feasibility bound. For the sake of clarity, the following lemma restates one of the theorems presented by George et al. in [5].

LEMMA 2 *If the schedule of a given synchronous task set τ is not feasible, i.e. a deadline is missed for the first time at t_{miss}, then $t_{miss} < I_n$ holds, where I_n is George's bound.*

Proof: This lemma simply excludes the possibility that $t_{miss} = I_n$ holds. Because a deadline is missed at t_{miss}, the total execution demand of τ at t_{miss} is greater than the available time t_{miss}, i.e. $t_{miss} < h(t_{miss})$. Where $h(t)$ is the demand bound function defined by Baruah et al. in [2] and has the expression: $h(t) = \sum_{i=1}^{n} max \left(0, \lfloor \frac{t - d_i}{p_i} \rfloor + 1 \right) \cdot e_i$. Considering $min(p_i, d_i)$ instead of d_i, we obtain: $t_{miss} < \sum_{i=1}^{n} \left(\lfloor \frac{t_{miss} - min(p_i,d_i)}{p_i} \rfloor + 1 \right) \cdot e_i$. And removing the floor function, we reach:
$t_{miss} < \sum_{i=1}^{n} \left(\frac{t_{miss} - min(p_i,d_i)}{p_i} + 1 \right) \cdot e_i$.

Reshaping this inequality to obtain t_{miss}, we get the expression $t_{miss} < I_n$. In words, George's bound is always greater than the time instant at which a deadline is missed for the first time. □

LEMMA 3 *If a deadline is missed for the first time at t_{miss}, then $t_{miss} < I_n^{(1)} = \frac{c_x^{(0)} \cdot e_x + \sum_{i=1; i \neq x}^{n} (p_i - min(p_i,d_i)) \cdot u_i}{1 - U_{-x}}$, where $u_i = \frac{e_i}{p_i}$, $U_{-x} = \sum_{i=1; i \neq x}^{n} u_i$ and $c_x^{(0)}$ is given by $max \left(0, \lceil \frac{I_n - d_x}{p_x} \rceil \right)$.*

Proof: We know from lemma 2 that t_{miss} is always less than I_n. Consequently, $c_x^{(0)} = max \left(0, \lceil \frac{I_n - d_x}{p_x} \rceil \right)$

is equal to or greater than the amount of jobs of task T_x in the interval $[0, t_{miss}]$. Additionally, as a deadline is missed at time t_{miss}, the total execution demand in $[0, t_{miss}]$ is greater than t_{miss}, i.e. $t_{miss} < \sum_{i=1;i\neq x}^{n} max\left(0, \lfloor\frac{t_{miss}-d_i}{p_i}\rfloor + 1\right) \cdot e_i + c_x^{(0)} \cdot e_x \leq \sum_{i=1;i\neq x}^{n}\left(\lfloor\frac{t_{miss}-min(p_i,d_i)}{p_i}\rfloor + 1\right) \cdot e_i + c_x^{(0)} \cdot e_x$.

By removing the floor function and reshaping to get t_{miss}, we obtain:
$t_{miss} < \frac{c_x^{(0)} \cdot e_x + \sum_{i=1;i\neq x}^{n}(p_i - min(p_i,d_i)) \cdot u_i}{1-U_{-x}}$. □

LEMMA 4 *$I_n > I_n^{(1)} = \frac{c_x^{(0)} \cdot e_x + \sum_{i=1;i\neq x}^{n}(p_i - min(p_i,d_i)) \cdot u_i}{1-U_{-x}}$ holds, where $u_i = \frac{e_i}{p_i}$, $U_{-x} = \sum_{i=1;i\neq x}^{n} u_i$ and $c_x^{(0)}$ is given by $max\left(0, \lceil\frac{I_n - d_x}{p_x}\rceil\right)$.*

Proof: From the derivation of I_n, we know that the approximated total execution demand, which results by removing the floor function and considering $min(p_i, d_i)$ instead of d_i, is equal to I_n. This can be mathematically expressed as follows:
$I_n = \sum_{i=1;i\neq x}^{n}\left(\frac{I_n - min(p_i,d_i)}{p_i} + 1\right) \cdot e_i + c'_x \cdot e_x$.

In the previous inequality, we have substituted the term $\frac{I_n - min(p_x,d_x)}{p_x} + 1$ by c'_x. Reshaping to get I_n, we have: $I_n = \frac{c'_x \cdot e_x + \sum_{i=1;i\neq x}^{n}(p_i - min(p_i,d_i)) \cdot u_i}{1-U_{-x}}$.

Finally, it is easy to see that $c'_x > c_x^{(0)}$ holds, where $c_x^{(0)}$ is equal to $max\left(0, \lceil\frac{I_n - d_x}{p_x}\rceil\right)$. The thesis follows. □

Lemma 4 shows that it is possible to achieve an improvement, based on lemma 3, of the best known feasibility bound. Following theorem demonstrates that a recursive application of lemma 3 is also valid.

THEOREM 1 *If a deadline is missed for the first time at t_{miss}, then $t_{miss} < I_n^{(k)} = \frac{\sum_{i=1}^{k} c_i^{(i-1)} \cdot e_i + \sum_{i=k+1}^{n}(p_i - min(p_i,d_i)) \cdot u_i}{1-U_{-1;\ldots;-k}}$, where $u_i = \frac{e_i}{p_i}$, $U_{-1;\ldots;-k} = \sum_{i=k+1}^{n} u_i$, $c_i^{(i-1)} = max\left(0, \lceil\frac{I_n^{(i-1)} - d_i}{p_i}\rceil\right)$, $I_n^{(0)} = I_n$ is George's bound and $1 \leq k \leq n$.*

Proof: We know from lemma 3 and lemma 4 that $t_{miss} < I_n^{(1)} < I_n$ holds. Consequently, considering $x = 1$ in lemma 3, we have that $c_1^{(0)} = max\left(0, \lceil\frac{I_n - d_1}{p_1}\rceil\right)$ is equal to or greater than the amount of jobs of task T_1 in the interval $[0, t_{miss}]$. In the same way, $c_2^{(1)} = max\left(0, \lceil\frac{I_n^{(1)} - d_2}{p_2}\rceil\right)$ is equal to or greater than the amount of jobs of task T_2 in $[0, t_{miss}]$. We proceed analogously to lemma 3 to get: $t_{miss} < I_n^{(2)} = \frac{c_1^{(0)} \cdot e_1 + c_2^{(1)} \cdot e_2 + \sum_{i=3}^{n}(p_i - min(p_i,d_i)) \cdot u_i}{1-U_{-1;-2}}$, where $U_{-1;-2} = \sum_{i=3}^{n} u_i$. Exactly as in lemma 4, it can be easily proved that $t_{miss} < I_n^{(2)} < I_n^{(1)} < I_n$ holds. Proceeding recursively for the first k tasks and $1 \leq k \leq n$, we get:

$$t_{miss} < \frac{\sum_{i=1}^{k} c_i^{(i-1)} \cdot e_i}{1-U_{-1;\ldots;-k}} + \frac{\sum_{i=k+1}^{n}(p_i - min(p_i,d_i)) \cdot u_i}{1-U_{-1;\ldots;-k}}. \quad (1)$$

The right member of this inequality is $I_n^{(k)}$, where $c_i^{(i-1)}$ is given by $max\left(0, \lceil\frac{I_n^{(i-1)} - d_i}{p_i}\rceil\right)$. Note that for $k = n$, inequality 1 reduces to $t_{miss} < I_n^{(n)} = \sum_{i=1}^{n} c_i^{(i-1)} \cdot e_i$. □

6. Polynomial-time feasibility tests for EDF

In this section, we prove that it is possible to build polynomial-time sufficient feasibility tests for EDF basing on theorem 1.

LEMMA 5 *Assuming that tasks in τ are sorted according to non-decreasing relative deadlines, τ is feasible under EDF if $I_k^{(k)} \leq d_k$ holds for every k, where $1 \leq k \leq n$ and $I_k^{(k)}$ is the feasibility bound of theorem 1 for the first k tasks.*

Proof: Let us analyze the case $k = 1$—only for T_1; if $I_1^{(1)} \leq d_1$ holds, there are no deadlines in $[0, I_1^{(1)})$, because tasks are sorted according to non-decreasing relative deadlines—d_1 is the shortest one, and theorem 1 excludes the possibility that a deadline miss happens at $t = I_1^{(1)}$. As a consequence, the task set with only T_1 is feasible. For $k = 2$, we compute $I_2^{(2)}$—only for T_1 and T_2; if $I_2^{(2)} \leq d_2$ holds, the task set with only T_1 and T_2 is also feasible because T_1 alone was proven to be feasible in the previous step and there are no deadlines of T_2 in $[0, I_2^{(2)})$. Moreover, a deadline miss cannot happen at $t = I_2^{(2)}$ according to theorem 1. For $k = 3$, we compute $I_3^{(3)}$—only for T_1, T_2 and T_3; if $I_3^{(3)} \leq d_3$ holds, the first three tasks are also feasible because T_1 and T_2 together were proven to be feasible in the previous step and there are no deadlines of T_3 in $[0, I_3^{(3)})$. A deadline miss cannot happen at $t = I_3^{(3)}$ because of theorem 1.

Now, let us assume that $I_k^{(k)} \leq d_k$ holds for $1 \leq k \leq n-1$. If $I_n^{(n)} \leq d_n$ also holds, τ is feasible because the first $n-1$ tasks were proven to be feasible in the previous steps and there are no deadlines of T_n in $[0, I_n^{(n)})$. In accordance to theorem 1, a deadline miss can neither happen exactly at $t = I_n^{(n)}$. The thesis follows. □

LEMMA 6 *A polynomial-time feasibility test based on theorem 1 is more accurate than Devi's condition.*

Proof: Immediate from lemma 1 and lemma 4. □

6.1. Feasibility test with $\mathbf{O(n^2)}$

Figure 1 presents an algorithm with $O(n^2)$ denominated *ptftn*^{2}, which bases on lemma 5. Theorem 1 assumes that George's bound $I_n = I_n^{(0)}$ is known. Thus, this latter must be calculated before $I_n^{(n)}$ can be obtained.

After sorting tasks according to non-decreasing relative deadlines, the algorithm of figure 1 computes iteratively the feasibility bound of theorem 1 for the first k tasks, where $1 \leq k \leq n$. For the reason that it generally delivers better results, we begin calculating $I_k^{(k)}$ from the task with the longest deadline T_k on. This proceeding affects in no way the validity of theorem 1, because this latter makes no assumption on tasks' order.

In every iteration of the nested for-loop, $I_k^{(k-i+1)}$ is calculated and compared to d_k. If for any step $I_k^{(k-i+1)} \leq d_k$ holds, the first k tasks will be feasible because of lemma 5 and of taking into account that theorem 1 guarantees that $I_k^{(k-i+1)} \geq I_k^{(k)}$ holds for $1 \leq i \leq k$. In this latter case, the algorithm continues running towards $k = n$. However, if $i = 1$ is reached, i.e. $I_k^{(k-i+1)} = I_k^{(k)}$ is reached, and $I_k^{(k)} > d_k$ holds, we would not be able to make any assertion about the feasibility of the first k tasks, so that we will have to consider them as not feasible.

6.2. Feasibility test with $\mathbf{O(n \log n)}$

A complexity $O(n \log n)$ is possible by limiting the nested for-loop in figure 1 to certain number of iterations. We propose to fix these iterations to 100, what results in an algorithm we call *ptftnlogn-100*. However, it might be better to choose this value according to particular cases. We denominate these similar algorithms by *ptftnlogn-x*. Figure 2 shows necessary changes on *ptftn*^{2} in order to obtain the less complex *ptftnlogn-100*.

7. Experimental results

In this section, we present and evaluate some experiments comparing the proposed algorithms with already known polynomial-time feasibility tests. Furthermore, we extended this comparison to include the fastest exact feasibility test, called *AllApprox*, presented in [1]. Mentioned algorithms are contrasted with respect to accuracy and their running time. In order to achieve a sensible comparison, random task sets were uniformly generated for different processor utilizations as recommended in [3]—*UUniFast* was applied.

```
$U_k = r_k = 0$;
sort according to non-decreasing $d_i$;
for $k = 1$ to $n$ /*n=number of Tasks in $\tau$*/
    $U_k = U_k + \frac{e_k}{p_k}$;
    $r_k = r_k + (p_k - min(p_k, d_k)) \cdot \frac{e_k}{p_k}$;
    if $U_k \geq 1$
        return("not feasible");
    end
    /*If $U_k < 1$, we can find George's bound*/
    $I_k^{(0)} = \frac{r_k}{1-U_k}$;
    $r_i = r_k$;
    $U_i = U_k$;
    for $i = k$ to 1 /*Bound of theorem 1*/
        $c_i^{(k-i)} = max\left(0, \lceil \frac{I_k^{(k-i)} - d_i}{p_i} \rceil\right)$;
        $U_i = U_i - \frac{e_i}{p_i}$;
        $r_i = r_i - (p_i - min(p_i, d_i)) \cdot \frac{e_i}{p_i} + c_i^{(k-i)} \cdot e_i$;
        $I_k^{(k-i+1)} = \frac{r_i}{1-U_i}$;
        if $I_k^{(k-i+1)} \leq d_k$
            break;
        elseif $i == 1$ /*last iteration?*/
            return("not feasible");
        end
    end
end
return("feasible");
```

Figure 1. Algorithm *ptftn*^{2}

7.1. Performance comparison

Figure 3 illustrates the performance, in terms of accepted task sets, versus processor utilization. To obtain task parameters, we proceeded as follows: Once generated u_i as mentioned above, we created periods p_i also in a random way with uniform distribution; consequently, we got $e_i = u_i \cdot p_i$. Relative deadlines d_i were uniformly chosen from the range $[e_i, p_i]$. Additionally, we sampled the utilization axis in 2.5% hops, for which the huge amount of 10000 different task sets were generated each time. In this manner, different curves were created for 5, 10, 100, 500 and 1000 tasks per task set. These curves were averaged together in figure 3 in order to show how feasibility tests behave independently of the number of tasks n.

As depicted in figure 3, *ptftn*^{2} performs almost as well as exact algorithms in the utilization range $(0, 85\%]$. This algorithm outperforms all known polynomial-time tests for utilizations in $(60\%, 90\%)$, reaching a peak at $U = 0.8$ of about 40% more accepted task sets over Devi's test.

On the other side, *ptftnlogn-100* accepts around 10% more task sets than Devi's test in the utilization range $(70\%, 90\%)$. As expected, the performance of this latter algorithm degrades to that of Devi's test as n increases.

Figure 4 shows a performance comparison versus gap,

```
$U_k = r_k = 0$;
$iters = 100$;
sort according to non-decreasing $d_i$;
for $k = 1$ $to$ $n$ /*n=number of Tasks in $\tau$*/
⋮
    $i = k$;
    /*Bound of theorem 1, only $iters$ tasks*/
    while $i > 0$
        ⋮
        if $I_k^{(k-i+1)} \leq d_k$
            break;
        elseif $i == 1$ || $i == k - iters$
            return("not feasible");
        end
        $i = i - 1$;
    end
end
return("feasible");
```

Figure 2. Algorithm *ptftnlogn-100*

where gap is defined as the difference between p_i and d_i. The gap axis was also sampled in 2.5% hops, for which again 10000 different task sets were generated each time. Figure 4 presents the average behavior for 5, 10, 100, 500 and 1000 tasks per task set too. For this plot, processor utilization was fixed to 80%. Periods p_i and worst-case execution times e_i were selected in the same way as before. This time, deadlines were chosen by $d_i = p_i - gap$, where gap ranges between 0 for $d_i = p_i$ and 100% for $d_i = e_i$. The proposed *ptftn*^{2} performs almost like exact algorithms and outperforms all polynomial-time tests in the gap range $(30\%, 55\%)$. On its part, proposed test *ptftnlogn-100* performs better than all polynomial-time tests for gaps in $(35\%, 55\%)$. Again, the performance of this latter test worsens as the number of tasks increases.

Figure 5 shows the schedulability versus number of tasks for 80% processor utilization. The number of tasks was incremented in 50-task steps, while for each step 10000 task sets were generated. Test *ptftn*^{2} performs almost like exact tests as the number of tasks grows. As expected, the performance of *ptftnlogn-100* falls rapidly for $n > 100$.

7.2. Comparison of running time

We implemented algorithms on Matlab—besides using Matlab's *sortrows* for sorting task sets where necessary, routine *testlist.add*, for adding elements to a sorted list, in *AllApprox* was implemented with $O(\log n)$. Algorithms were timed on a 2GHz Intel Core 2 Duo machine running Windows XP. For creating curves, no distinction was made between feasible and unfeasible task sets.

Figure 6 compares the average running time versus utilization. Here again, we proceeded as described above in order to generate curves independently of the number of tasks. As it can be observed, proposed algorithms are around two orders of magnitude faster than AllApprox in the utilization interval $(0, 70\%]$, but they are almost two orders of magnitude slower than the density test. In $(70\%, 90\%)$, the running time of *ptftn*^{2} gets drastically closer to Allapprox, but there is still approximately an order of magnitude between them. This drastic change experienced by *ptftn*^{2} responds to the fact that the feasibility bound $I_k^{(k)}$ of theorem 1 is calculated as it gets necessary—if $I_k^{(k-i+1)} \leq d_k$ holds for any i, the calculation of $I_k^{(k)}$ will be broken off, see figure 1—and as expected, we need to calculate the whole $I_k^{(k)}$ for higher

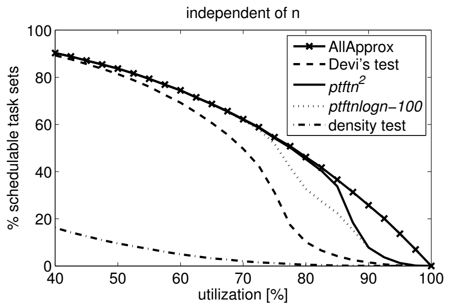

Figure 3. Schedulability vs. utilization

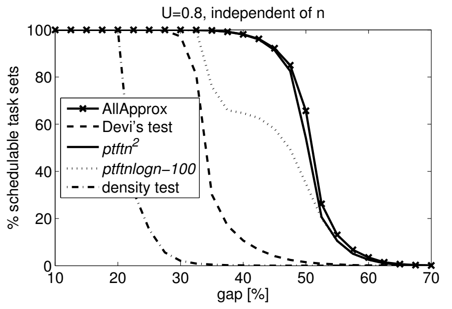

Figure 4. Schedulability vs. gap

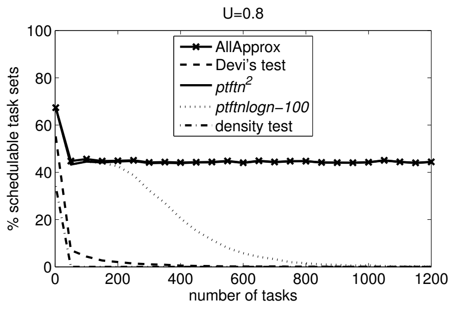

Figure 5. Schedulability vs. number of tasks

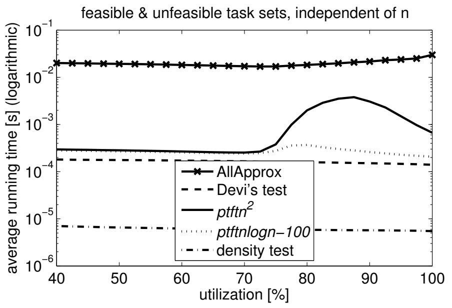

Figure 6. Running time vs. utilization

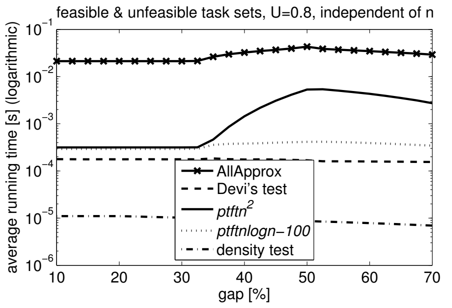

Figure 7. Running time vs. gap

utilizations. Around 85% utilization, the running time of *ptftn*^{2} falls gradually because more task sets are found unfeasible with less iterations.

In figure 7, the average running time for $U = 0.8$ versus gap is plotted. Task sets were generated as previously in order to reach curves that are relatively independent of n. In this case, algorithms behave similarly to figure 6.

On its side, figure 8 depicts the running time of algorithms for $U = 0.8$ as the number of tasks grows. The number of tasks was incremented in 50-task steps proceeding exactly as already described. The running time of *ptftn*^{2} remains an order of magnitude faster than AllApprox as n grows, but it gets worse in relation to the other polynomial-time algorithms.

8. Conclusions

In this paper, two better polynomial-time sufficient feasibility tests were presented for EDF considering arbitrary relative deadlines and a preemptive uniprocessor scheduling. For complexity $O(n^2)$, we proved that our proposed test *ptftn*^{2} outperforms Devi's condition in the sense of accuracy. Moreover, we showed that *ptftn*^{2} behaves almost like exact feasibility tests in terms of accepted task sets and for the utilization range $(0, 85\%]$. A complexity $O(n \log n)$ was also shown to be possible, however, the performance

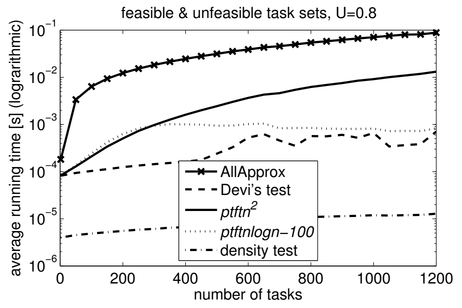

Figure 8. Running time vs. number of tasks

of our suggested test *ptftnlogn-100* deteriorates towards the one of Devi's test as the number of tasks n grows. As stated in section 6.2, it is additionally possible to design other tests with complexity $O(n \log n)$ that might be more convenient for particular applications.

Finally, considering that *ptftn*^{2} performs well when processor utilization is less than 90%, we believe it to be the best alternative to more time-consuming exact algorithms in many practical situations. In addition, as the running time is kept moderate for most practical application, e.g. $n \approx 600$, we believe *ptftn*^{2} also to be more recommendable than the other polynomial-time feasibility tests.

References

[1] K. Albers and F. Slomka. Efficient feasibility analysis for real-time systems with edf scheduling. *Proceedings of the DATE 05 Conference*, March 2005.

[2] S. Baruah, A. Mok, and L. Rosier. Preemptively scheduling hard-real-time sporadic tasks on one processor. *Proceedings of the Real-Time Systems Symposium*, December 1990.

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

[4] M. Devi. An improved schedulability test for uniprocessor periodic task systems. *Proceedings of the 15th Euromicro Conference on Real-Time Systems*, July 2003.

[5] L. George, N. Rivierre, and M. Spuri. Preemptive and non-preemptive real-time uniprocessor scheduling. *Rapport de Recherche RR-2966, INRIA*, 1996.

[6] C. Liu and J. Layland. Scheduling algorithms for multiprogramming in hard real-time environments. *Journal of the Association for Computing Machinery*, 20(1):40–61, 1973.

[7] J. Liu. *Real-Time Systems*. Prentice Hall, 2000.

[8] A. Ripoll, I. Crespo and A. Mok. Improvement in feasibility testing for real-time tasks. *Real-Time Systems*, 11(1):19–39, 1996.

[9] M. Spuri. *Earliest Deadline Scheduling in Real-Time Systems*. PhD Thesis at Scuola Superiore S. Anna, Italy, 1995.

[10] J. Stankovic, M. Spuri, K. Ramamritham, and G. Buttazzo. *Deadline Scheduling for Real-Time Systems: EDF and Related Algorithms*. Kluwer, 1998.