Efficient schedulability tests for real-time embedded systems with urgent routines

J. Augusto Santos Jr. · George Lima · Konstantinos Bletsas

Abstract Task scheduling is one of the key mechanisms to ensure timeliness in embedded real-time systems. Such systems have often the need to execute not only application tasks but also some urgent routines (*e.g.* error-detection actions, consistency checkers, interrupt handlers) with minimum latency. Although fixed-priority schedulers such as Rate-Monotonic (RM) are in line with this need, they usually make a low processor utilization available to the system. Moreover, this availability usually decreases with the number of considered tasks. If dynamic-priority schedulers such as Earliest Deadline First (EDF) are applied instead, high system utilization can be guaranteed but the minimum latency for executing urgent routines may not be ensured.

In this paper we describe a scheduling model according to which urgent routines are executed at the highest priority level and all other system tasks are scheduled by EDF. We show that the guaranteed processor utilization for the assumed scheduling model is at least as high as the one provided by RM for two tasks, namely $2(\sqrt{2}-1)$. Seven polynomial time tests for checking the system timeliness are derived and proved correct. The proposed tests are compared against each other and to an exact but exponential running time test.

Keywords Real-time embedded systems · Schedulability analysis · Earliest Deadline First · Rate-Monotonic

1 Introduction

Motivation Embedded real-time systems have become increasingly complex. For example, nowadays cars contain several embedded computers connected to each other via communication networks. There is often the need of integration between high-level application functionality, such as multimedia appliances and navigation systems, with low-level control systems, such as those used in functions like drive-by-wire, brake-by-wire or steer-by-wire. Similar trends can be mentioned in other application fields like autonomous robots, aircraft systems or automation industry. In any case, safety-critical issues must be properly addressed since any failures, in the time or in the value domains, could result in either undesired material losses or endanger human safety. Moreover, there is often the need to run in these systems urgent routines, namely small pieces of code responsible for carrying out consistency-checkers, error-detection, interrupt handlers, etc. Ideally such routines must run at high frequency and with minimum delays.

The functions of such embedded systems are often described by recurrent tasks, which are triggered either by predefined time events or by signals captured from sensors connected to the computing system. Tasks must be selected for execution so as to comply with their specified deadlines taking into consideration that one task cannot jeopardize the other's time correctness. Scheduling is thus a fundamental service for embedded real-time systems whose correctness all other services depend on. Among the classical scheduling policies are the Rate-Monotonic (RM) and Earliest Deadline First (EDF) [15]. According to the former, each task has a fixed priority, assigned off-line such that tasks that are activated more frequently receive higher priorities. EDF is a dynamic-priority policy according to which the task with the current earliest deadline receives the highest priority. Either RM or EDF selects at any scheduling time the task with the highest priority to execute. When urgent routines are considered in fixed-priority scheduled systems, they usually run at the highest priority level. As task priorities varies in dynamic-priority scheduled systems, there may not be possible to ensure that some task will delay or preempt the execution of urgent routines.

When designing an embedded real-time system, one must ensure whether all tasks meet their deadlines. That is, given a system composed of a set of tasks scheduled by a given policy, are all deadlines always met? This question is usually known as the schedulability analysis problem. There are well known results in this area concerning both RM and EDF. For example, if tasks are preemptively scheduled by EDF on a processor, all deadlines are met if and only if the task set does not require more than 100 % of processing resources. When RM is considered, no deadline is missed provided that no more than $n(2^{\frac{1}{n}} - 1)$ of the processor is used. These results are valid for independent sporadic tasks with implicit deadlines. That is, these tasks have a minimum known inter-arrival time; do not share any resource but the processor; and any released task must finish execution at any time between its release instant and its next arrival. This usual task model is also assumed in this paper.

We note that the requirements of urgent routines are in line with the fixed-priority scheduling model. Such routines can be encapsulated into a high priority task avoiding the interference due to the execution of all other application tasks. Doing so, however, reduces the achievable system utilization to about 69 % in the case of RM, which is the limit of function $n(2^{\frac{1}{n}} - 1)$ for large n. On the other hand, using a dynamic-priority scheduling algorithm such as EDF does not offer guarantees that urgent routines are always executed with highest priority making them subject to preemption and execution delays. Figure 1 exemplifies these observations, showing possible delays in an EDF schedule for three tasks one of which represents urgent routines, identified by symbol u. In this illustration, up-arrows are task releases and white boxes are task executions. The system must finish the execution of

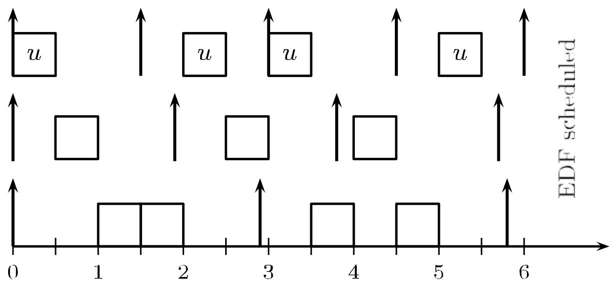

Fig. 1 EDF-scheduled tasks with urgent routines being delayed

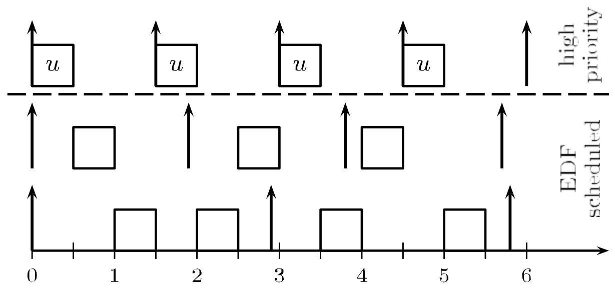

Fig. 2 EDF-scheduled tasks suffering interference due to the execution of a high priority task

a task before its next instance is released. As can be seen in the produced schedule, urgent routines are being subject to delays due to execution of other tasks during intervals $[1.5, 2)$ and $[4.5, 5)$. Fixed-priority scheduling for this example is not an option since the system cannot be feasibly scheduled by such a scheme.

Contribution We provide the means of analyzing EDF-scheduled systems that implement urgent routines as the highest-priority task. The assumed scheduling model has two fixed priorities. The highest priority is reserved to execute urgent routines whereas all other system tasks are scheduled by EDF within the lowest priority level. Figure 2 illustrates this scheme showing the schedule produced for the same example shown in Fig. 1. As can be noted, all tasks finish their execution before its next activation time, implicitly assumed to be their deadlines. It is interesting to observe that although this system has a high processor utilization it can be scheduled using such a mix of fixed- and dynamic-priority schemes. For identifying schedulable systems for this scheduling model, we have derived a set of efficient schedulability tests.

The main results of this paper can be summarized as follows:

- Seven new schedulability tests for the model described in Fig. 2 are derived. All these tests are proved correct. They provide *sufficient* schedulability conditions, which means that all non-schedulable systems are identified as such. System tasks are assumed to be sporadic and have implicit deadlines.
- We show that the assumed scheduling model provides the same schedulability bound as that obtained for RM when applied for 2 tasks. That is, the system can use at least $2(\sqrt{2} - 1) \approx 83\,\%$ of the processor independently of the number of tasks.
- We also show in this paper that checking schedulability for the model described in Fig. 2 can be done via well known schedulability tests for fixed-priority systems [2, 5, 15] applied to two tasks, one being the urgent routines and the other representing all EDF-scheduled tasks.

– We evaluate the proposed schedulability tests both theoretically and via simulation. In our theoretical evaluation we identify three schedulability tests (out of the seven proposed) that dominate the others. They then are jointly applied to verify system schedulability. The experimental evaluation shows that the average behavior of the proposed tests is well above the theoretical bound of $2(\sqrt{2}-1)$. Indeed, experiments carried out on synthetic generated systems indicate that the proposed tests perform very well for systems that use up to 95 % of processor.

Organization A brief overview on related work is given in Sect. 2. We then introduce the computation model and notation in Sect. 3. The proposed schedulability tests are described in Sect. 4. Some discussion on the derived tests is presented in Sect. 5. The proposed tests are evaluated by simulation in Sect. 6. Our final comments are given in Sect. 7.

2 Related work

The scheduling model assumed in this work conforms with what is usually known as hierarchical scheduling, an extensive research field for which we only mention some results. Most work on this field, though, aims at providing temporal isolation in the system so that possible overruns or overloads do not propagate in the system [16]. This is specially useful when not all task parameters are known beforehand. Temporal isolation is commonly implemented via servers, which are virtual tasks used to schedule the actual system tasks. Different hierarchical scheduling frameworks have been applied to several domains such as controlling execution interferences of device-drivers (e.g., [9]); providing composable schedulability analysis (e.g., [10, 19]) or serving as a means of scheduling aperiodic soft real-time tasks (e.g., [20]). This branch of research also focuses on both identifying suitable server parameters and schedulability analysis for a given system (e.g., [1, 8]). Here we are concerned with a much simpler hierarchical model, as described in Fig. 2, for which we show that efficient schedulability tests can be used.

To the best of our knowledge, Jeffay and Stone [12] were the first ones to address the schedulability analysis problem considering the model described in Fig. 2. Actually, they developed an *exact schedulability test* for a slightly more generic version of this model according to which there are different fixed-priority levels above the EDF-scheduled tasks. Their schedulability test is capable of precisely identifying both schedulable and non-schedulable systems. Later on, Gonzalez and Palencia [11] have also presented an exact test for a more general scheduling framework according to which the scheduler handles several priority levels and within each level tasks can be scheduled either in a fixed-priority fashion or by EDF. Zhang and Burns [21, 22] provided improvements on the running time of exact schedulability tests. All these tests run in pseudo-polynomial or exponential time, as it is common for exact tests in EDF-scheduled systems [3] since they are all based on processing time demand functions. Unlike these schemes, we are interested in schedulability tests with low computational complexity for the specific model depicted in Fig. 2. Although these exact schedulability tests have their value in practice, they are not applicable for on-line use neither can they be used as a means of determining processor utilization bounds.

The need for executing urgent routines at higher priority levels have been identified before [13, 14] in the context of fixed-priority real-time systems. Our current work differs from such approaches in two main aspects. First, we consider an hybrid of EDF and fixed-priority. Second, this previous work aims at finding the highest priority level capable of dealing with urgent routines. Instead, we are assuming the execution of urgent routines at the highest priority level and checking for system schedulability.

Four of the tests described in this paper has been previously presented [17]. We now significantly extend such results, deriving three new schedulability tests, comparing them both theoretically and experimentally. Moreover, a least upper bound of $2(\sqrt{2}-1)$ on processor utilization for the assumed task model is now given.

One of the schedulability tests derived in this paper, namely Test 3, has been used in the context of multiprocessor scheduling [18]. This highlights that the results presented here may be applied in a broader context. Indeed, we have derived a least upper bound on the system utilization for the C=D scheduling algorithm [7], an open problem until recently.

3 Notation and system model

We consider a set of sporadic tasks $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\}$ to be scheduled on a single processor according to EDF and assume that there is a high priority task, τ_0, that may interfere in the execution of any task in Γ. Task τ_0 represents urgent routines that require minimum latencies. Examples of urgent routines are error checking, interrupt handlers, etc. Tasks in Γ are assumed to be fully preemptable and all tasks are independent of one another.

Any task τ_i in the system is denoted by a tuple (C_i, T_i), $i \geq 0$, where $C_i \leq T_i$ represents its maximum required computation time and T_i is its minimum inter-arrival time, also called period for historical reasons. That is, any task τ_i is assumed to release possibly an infinity number of instances each of which at least T_i apart from the other. Since tasks have implicit deadlines, T_i also represents the relative deadline of τ_i. In other words, if a task τ_i arrives at time t, the system must schedule it for execution so that C_i processor units are allocated to τ_i within $[t, t + T_i)$. We denote $U(\tau_i) = \frac{C_i}{T_i}$ the utilization of a task τ_i and the utilization of a task set Γ is denoted $U(\Gamma) = \sum_{\tau_i \in \Gamma} U(\tau_i)$.

We also assume that the period of τ_0 is not greater than the period of any other task in Γ. This assumption is not necessary for all derived tests and it mostly comes from the optimality of the RM priority assignment [15]. As τ_0 represents urgent routines, this assumption does not restrict the applicability of results presented in this paper. We note that τ_0 is sporadic, as any other task in the system is, and so its release times in the system may be more than T_0 apart.

Task τ_0 may be interpreted as a reserve at the highest priority level. Even if the system has different pieces of code as urgent routines, they all can be executed as long as no more than C_0 time units is needed within a time window of T_0. In any case, for convenience, we assume a single task (τ_0) representing the urgent routines of the system.

4 Schedulability tests

We now start deriving the new sufficient schedulability tests. Theorems 1–3 show three of them by establishing a bound on processor utilization above which the system is considered not schedulable. We then present Theorem 4, which shows that schedulability tests developed for fixed-priority systems can be adapted to analyze systems that fit the scheduling model assumed in this paper. Based on this result, other four new schedulability tests are derived in Theorems 5–8.

Theorem 1 (Test 1) *Let $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\}$ be a set of tasks scheduled by EDF and let τ_0 be the highest-priority task. There is no deadline miss provided that Eq.* (1) *holds.*

$$\left(\frac{T_0}{\min_{\tau_i \in \Gamma}(T_i)} + 1\right) U(\tau_0) + U(\Gamma) \leq 1 \tag{1}$$

Proof From the assumed model, τ_0 does not miss its deadline and so assume that some task $\tau_i \in \Gamma$ misses its deadline at some time d. Let $r < d$ be its release time. Also, consider the last time $t < d$ so that the processor is not idle within $[t, d)$ but is idle just before t. If such a time does not exist, let $t = 0$. Note that $t \leq r$ and $d - r = T_i$. To simplify notation, let $\Delta = r - t$. Let us compute the maximum demand within $[t, d)$ from those tasks that may interfere in the execution of τ_i. As for tasks in Γ, we must account for the execution of tasks whose jobs have deadlines less than or equal to d. Also, since τ_0 interferes in the execution of any task in Γ, its activation within $[t, d)$ must be accounted for. It is known that τ_0 does not arrive more than $\lceil \frac{T_i+\Delta}{T_0} \rceil$ times during $[t, d)$. Computing the total demand and considering that τ_i misses its deadline yields

$$\left\lceil \frac{T_i + \Delta}{T_0} \right\rceil C_0 + \sum_{\tau_j \in \Gamma} \left\lfloor \frac{T_i + \Delta}{T_j} \right\rfloor C_j > T_i + \Delta$$

$$\left(\frac{T_i + \Delta}{T_0} + 1 \right) C_0 + U(\Gamma)(T_i + \Delta) > T_i + \Delta$$

$$U(\tau_0) + \frac{U(\tau_0)T_0}{T_i + \Delta} + U(\Gamma) > 1$$

$$\left(\frac{T_0}{T_i} + 1 \right) U(\tau_0) + U(\Gamma) > 1 \tag{2}$$

Condition (2) must hold for any task $\tau_i \in \Gamma$ that misses its deadline. This implies that Eq. (1) is a sufficient schedulability test, as required. D

The second schedulability test takes advantage of periods that are multiple of the period of the highest priority task. It is required that $T_0 \leq \min_{\tau_i \in \Gamma}(T_i)$.

Theorem 2 (Test 2) *Let $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\}$ be a set of tasks scheduled by EDF and let τ_0 be the highest-priority task in the system such that $T_0 \leq \min_{\tau_i \in \Gamma}(T_i)$. There is no deadline miss provided Eq.* (3) *holds*.

$$U(\tau_0) + \sum_{\tau_i \in \Gamma} \frac{T_i}{\lfloor \frac{T_i}{T_0} \rfloor T_0} U(\tau_i) \leq 1 \tag{3}$$

Proof As τ_0 cannot miss its deadline, let us focus on tasks in Γ. Consider a task set Γ^t obtained from Γ as follows. For each task $\tau_i = (C_i, T_i) \in \Gamma$ there is a task $\tau^t = (C^t, T_0)$ in Γ^t, where

$$C_i' = \frac{C_i}{\lfloor \frac{T_i}{T_0} \rfloor}, \quad \text{and so} \quad U(\Gamma') = \sum_{\tau_i \in \Gamma} \frac{T_i}{\lfloor \frac{T_i}{T_0} \rfloor T_0} U(\tau_i)$$

Now consider scheduling τ_0 and Γ' with τ_0 being the highest-priority task and Γ' being scheduled by EDF. The worst-case response time of any task τ_i' is equal to $C_0 + \sum_{\tau_j \in \Gamma'} C_j'$. This means that τ_i' meets its deadline if $C_0 + \sum_{\tau_j \in \Gamma'} C_j' \leq T_0$, which in turn is equivalent to $U(\tau_0) + U(\Gamma') \leq 1$. Thus, the schedulability of Γ' is ensured by Eq. (3).

In any time interval $L \geq T_0$ the processing demand of Γ' is given by Eq. (4) whereas the maximum demand due to tasks in Γ equals $\sum_{\tau_j \in \Gamma} \lfloor \frac{L}{T_j} \rfloor C_j$.

$$\sum_{\tau_j \in \Gamma'} \left\lfloor \frac{L}{T_0} \right\rfloor C'_j \tag{4}$$

As $\lfloor \frac{L}{T_0} \rfloor = \lfloor \frac{L}{T_j} \frac{T_j}{T_0} \rfloor$ rewriting Eq. (4), we have that

$$\sum_{\tau_j \in \Gamma'} \left\lfloor \frac{L}{T_j} \frac{T_j}{T_0} \right\rfloor C'_j \geq \sum_{\tau_j \in \Gamma'} \left\lfloor \frac{L}{T_j} \right\rfloor \left\lfloor \frac{T_j}{T_0} \right\rfloor C'_j \geq \sum_{\tau_j \in \Gamma} \left\lfloor \frac{L}{T_j} \right\rfloor C_j \tag{5}$$

It follows from Eq. (5) that the processing demand due to tasks in Γ is not greater than that of tasks in Γ^t. As both task sets are scheduled according to EDF, the schedulability of Γ^t implies the schedulability of Γ. Therefore, Eq. (3) is a sufficient schedulability condition. D

The third schedulability test of interest is given by Theorem 3 and also requires that T_0 is not greater than the periods of tasks in Γ.

Theorem 3 (Test 3) *Let $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\}$ be a set of tasks scheduled by EDF and let τ_0 be the highest-priority task in the system such that $T_0 \leq \min_{\tau_i \in \Gamma}(T_i)$. There is no deadline miss provided that Eq.* (6) *holds.*

$$\left(\frac{U(\Gamma)}{\left\lfloor \frac{\min_{\tau_i \in \Gamma}(T_i)}{T_0} \right\rfloor} + 1 \right) U(\tau_0) + U(\Gamma) \leq 1 \tag{6}$$

Proof Let t^ϕ be the available time to execute the tasks in Γ within a time interval L. From Theorem 8 in [6] it is known that if $\forall L \geq \min_{\tau_i \in \Gamma}(T_i)$ Eq. (7) holds, then all tasks in Γ meet their deadlines.

$$L \sum_{\tau_i \in \Gamma} U(\tau_i) \leq t^\varphi \quad \Rightarrow \quad U(\Gamma) \leq \frac{t^\varphi}{L} \tag{7}$$

The minimum values of t^ϕ take place when τ_0 is periodically activated. Also, the values of L for minimizing the right-hand side of Eq. (7) occur when the start and ending of the interval L coincide with the activation and finishing of τ_0, respectively. This is because if L is further increased by E, $0 < E \leq T_0 - C_0$, the value of t^ϕ is also increased by E. In turn, if the value of L is decreased by a positive amount $E < C_0$, t^ϕ is kept constant. In other words, the values of L to be considered are given by $L = (k + j)T_0 + C_0$, where $k = \lfloor \frac{\min_{\tau_i \in \Gamma}(T_i)}{T_0} \rfloor$ and $j \in \mathbb{Z}_+$. In this case, for each time interval of size T_0, there are $(T_0 - C_0)$ time units available for executing tasks in Γ, which leads to $t^\phi = (k + j)(T_0 - C_0)$. Rewriting Eq. (7),

$$U(\Gamma) \leq \frac{(k+j)(T_0 - C_0)}{(k+j)T_0 + C_0} = \frac{(k+j)(T_0 - U(\tau_0)T)}{(k+j)T_0 + U(\tau_0)T_0} = \frac{1 - U(\tau_0)}{1 + \frac{U(\tau_0)}{k+j}} \tag{8}$$

The right-hand side of Eq. (8) is an increasing function of j. Letting $j = 0$ makes Eq. (8) become Eq. (6), as required. D

As can be noted, all the above schedulability tests run in $O(n)$ and are based on the processor utilization required by the whole task set. We use a different strategy to derive the fourth schedulability test. First, we show that checking the schedulability of a task $\tau_i \in \Gamma$ taking into consideration that τ_0 is executed at the highest priority level can be done via checking the schedulability of a system composed of only two tasks, τ_0 and a virtual-task $\tau_i^{\dagger}$. This latter task is to model the amount of computation resources required by tasks in Γ. Second, we apply response-time analysis to test the schedulability of n virtual two-task systems. If all of them are schedulable, system $\Gamma \cup \{\tau_0\}$ is also guaranteed to be schedulable.

Theorem 4 offers the basis that allows us to associate the schedulability of system $\{\tau_0\} \cup \Gamma$ with the schedulability of virtual system $\{\tau_0, \tau_i^{\dagger}\}$ scheduled in a fixed-priority manner. It is interesting to observe that this result hold independently of the period of τ_0.

Theorem 4 *Let $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\}$ be a set of tasks scheduled by EDF and let τ_0 be the highest priority task executed by the system. No task $\tau_i \in \Gamma$ misses its deadline provided that $\forall i \in \{1, \ldots, n\}$ task $\tau_i^{\dagger} = (U(\Gamma)T_i, T_i)$ does not miss its deadline when scheduled with τ_0 running at the highest priority level.*

Proof The maximum processing demand within any interval $[t_0, t_0 + t)$ due to tasks scheduled by EDF is given by

$$C(t) = \sum_{i=1}^{n} \left\lfloor \frac{t}{T_i} \right\rfloor C_i \tag{9}$$

A sufficient and necessary test for the schedulability of Γ is

$$C(t) \le S(t), \quad \forall t \ge 0$$

where $S(t)$ is a function providing a lower bound of the processing time supplied to Γ over an interval of length t.

However, since $S(t)$ is non-decreasing and since $C(t)$ is a "staircase" function, incrementing only for t that are integer multiples of some T_i, the above is equivalent to

$$C(t) \le S(t), \quad \forall t \in \bigcup_{\substack{\tau_i \in \Gamma \\ k=1,2,\ldots}} \{kT_i\} \tag{10}$$

By inspection, it holds that

$$C(t) \not> \sum_{i=1}^{n} \frac{C_i}{T_i} t = U(\Gamma)t, \quad \forall t \ge 0 \tag{11}$$

Combining this with the sufficient and necessary condition of Inequality (10), a sufficient test for the schedulability of Γ is

$$U(\Gamma)t \le S(t), \quad \forall t \in \bigcup_{\substack{\tau_i \in \Gamma \\ k=1,2,\ldots}} \{kT_i\} \tag{12}$$

Now let us suppose that for all $\tau_i \in \Gamma$ it holds that $\tau_i^{\dagger} = (U(\Gamma)T_i, T_i)$ is schedulable with τ_0 running at the highest priority level. Given that the supply of processing time to background tasks depends only on τ_0, it follows that it is lower-bounded by the same function $S(t)$ as before. Then, the schedulability of all $\tau_i^{\dagger}$ implies that

$$\left\lfloor \frac{t}{T_i} \right\rfloor C_i' \le S(t), \quad \forall i \in \{1, \ldots, n\}, \quad \forall t \in \{T_i, 2T_i, 3T_i, \ldots\} \tag{13}$$

Substituting $U(\Gamma)T_i$ for C_i^t and taking into account that for all $t \in \{T_i, 2T_i, 3T_i, \ldots\}$, it holds that $\lfloor \frac{t}{T_i} \rfloor = \frac{t}{T_i}$. Hence, the above expression, which follows from the schedulability of all τ_i^t, can be rewritten as:

$$\forall i \in \{1, \ldots, n\}, \quad \forall t \in \{T_i, 2T_i, 3T_i, \ldots\} : \frac{t}{T_i} U(\Gamma) T_i \leq S(t),$$

In turn, this can be equivalently rewritten as

$$U(\Gamma)t \leq S(t), \quad \forall t \in \bigcup_{\substack{\tau_i \in \Gamma \\ k=1,2,\ldots}} \{kT_i\}$$

which is equivalent to the sufficient condition for the schedulability of Γ expressed by Inequality (12). This proves the theorem. D

Theorem 4 implies that testing the schedulability of another system, created from Γ, suffices to determine that tasks in the original system may miss their deadlines. This provides interesting ways of checking the schedulability of task sets by checking the schedulability of 2 tasks using well known results. For example, Liu and Layland's schedulability test [15] could be used in the form

$$U(\tau_0) + U(\Gamma) \leq 2(\sqrt{2} - 1) \qquad (14)$$

Another option is by Bini *et al.* [5],

$$\big(U(\tau_0) + 1\big)\big(U(\Gamma) + 1\big) \leq 2 \qquad (15)$$

In both cases, the tests are applied as if the system was composed of two tasks, one of which consuming $U(\Gamma)$ of processor. These tests can only be applied, though, if the period of τ_0 does not exceed the period of any other task in the system. This is because Eqs. (14) and (15) were originally derived based on the RM priority assignment [5, 15]. Moreover, these equations provide only sufficient conditions. It is possible to use exact schedulability tests, which give more precise results. For this purpose, we apply the well known response-time analysis [2]. Although this is a pseudo-polynomial time procedure, it is known that it usually has a very fast convergence time. Being applied to a system composed of only two tasks, it is indeed a very efficient schedulability test. Furthermore, since response time analysis does not rely on which priority assignment is in consideration, one can use the result of Theorem 4 without assuming that τ_0 is the task of minimum period. Theorem 5 states such a result.

Theorem 5 (Test 4) *Let $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\}$ be a set of tasks scheduled by EDF and let τ_0 be the highest priority task executed by the system. No task $\tau_i \in \Gamma$ misses its deadline provided that the smallest fixed-point solution to Eq. (16) is not greater than T_i for all $\tau_i \in \Gamma$, where R_i is the worst-case response time for $\tau_i^t = (U(\Gamma)T_i, T_i)$.*

$$R_i = U(\Gamma)T_i + \left\lceil \frac{R_i}{T_0} \right\rceil C_0 \qquad (16)$$

Proof The smallest solution to Eq. (16) gives the worst-case response time for $\tau_i^t = (U(\Gamma)T_i, T_i)$ [2]. From Theorem 4 we know that the schedulability of τ_i^t implies the schedulability of τ_i and so the theorem follows. D

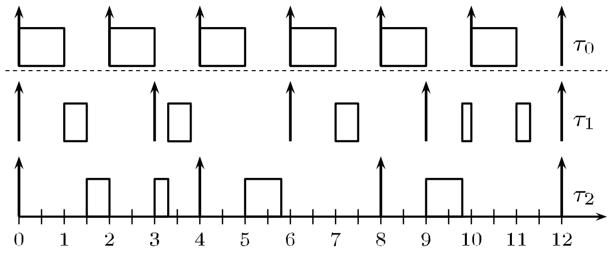

Fig. 3 Illustration that Test 4 is sufficient but not exact. The original task set, $\tau_0 = (1, 2)$, $\tau_1 = (0.5, 3)$, $\tau_2 = (0.8, 4)$, is schedulable while task set $\tau_0 = (1, 2)$, $\tau_1^t = (1.1, 3)$ is not, where $C_1^t = T_1 U(\{\tau_1, \tau_2\})$

It is important to stress that although response time analysis provides an exact schedulability test for the transformed system $\{\tau_0, \tau_i^t\}$, its result serves only as a sufficient condition for task set $\{\tau_0\} \cup \Gamma$. This is because the non-schedulability of $\{\tau_0, \tau_i^t\}$ does not imply the non-schedulability of $\{\tau_0\} \cup \Gamma$. To see this consider $\tau_0 = (1, 2)$, $\tau_1 = (0.5, 3)$ and $\tau_2 = (0.8, 4)$. As can be noted, task set $\{\tau_0, \tau_1^t\}$ is not schedulable, where $\tau_1^t = (1.1, 3)$. However, the original system $\{\tau_0, \tau_1, \tau_2\}$ is schedulable, as shown in Fig. 3.

Theorem 4 also opens up new possibilities for deriving new schedulability tests, as we now show in Theorems 6–8. We also observe that these tests can be performed in $O(n)$.

Theorem 6 (Test 5) *Let $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\}$ be a set of tasks scheduled by EDF and let τ_0 be the highest priority task. There is no deadline miss provided that Eq.* (17) *holds.*

$$\max_{\tau_i \in \Gamma} \left(\left\lceil \frac{T_i}{T_0} \right\rceil \frac{T_0}{T_i} \right) U(\tau_0) + U(\Gamma) \le 1 \tag{17}$$

Proof From Theorem 4, a sufficient condition such that no task in Γ ever misses a deadline is that for all τ_i^t:

$$C_i' + \left\lceil \frac{T_i}{T_0} \right\rceil C_0 \le T_i$$

This means that for all $\tau_i \in \Gamma$:

$$U(\Gamma) T_i + \left\lceil \frac{T_i}{T_0} \right\rceil U(\tau_0) T_0 \le T_i \Rightarrow U(\Gamma) + \left\lceil \frac{T_i}{T_0} \right\rceil \frac{T_0}{T_i} U(\tau_0) \le 1 \tag{18}$$

Since Eq. (18) is equivalent to Eq. (17), the claim follows. D

Theorem 7 (Test 6) *Let $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\}$ be a set of tasks scheduled by EDF and let τ_0 be the highest priority task in the system. There is no deadline miss provided that Eq.* (19) *holds.*

$$\max_{\tau_i \in \Gamma} \left(\frac{T_i}{\left\lfloor \frac{1 - U(\Gamma)}{U(\tau_0)} \frac{T_i}{T_0} \right\rfloor} \right) \frac{1}{T_0} \le 1 \tag{19}$$

Proof Define $\tau_i^t = (U(\Gamma) T_i, T_i)$ considering task $\tau_i \in \Gamma$. If τ^t never misses its deadline, then

$$C_i' + \left\lceil \frac{T_i}{T_0} \right\rceil C_0 \le T_i \quad \Rightarrow \quad \left\lceil \frac{T_i}{T_0} \right\rceil \le \frac{T_i - C_i'}{C_0}$$

As the left hand side of the above inequality is integer, it follows that

$$\left\lceil \frac{T_i}{T_0} \right\rceil \leq \left\lfloor \frac{T_i - C_i'}{C_0} \right\rfloor \tag{20}$$

This means that the number of instances of τ_0 within the period T_i of τ_i' is bounded by $\lfloor \frac{T_i - C_i'}{C_0} \rfloor$. From Theorem 4 we know that if no τ_i' misses its deadline, then all τ_i also meet their deadlines. Therefore from Inequality (20) and the fact that $\lceil \frac{T_i}{T_0} \rceil \frac{C_0}{T_i} \geq \frac{C_0}{T_0} = U(\tau_0)$, in order for the task in Γ to never miss deadlines, a sufficient condition is that for all $\tau_i \in \Gamma$, for all $\tau_i' = (U(\Gamma)T_i, T_i)$,

$$U(\tau_0) \leq \left\lfloor \frac{T_i - C_i'}{C_0} \right\rfloor \frac{C_0}{T_i}$$

$$U(\tau_0) \leq \left\lfloor \frac{(1 - U(\Gamma))T_i}{U(\tau_0)T_0} \right\rfloor \frac{U(\tau_0)T_0}{T_i}$$

$$T_i \leq \left\lfloor \frac{1 - U(\Gamma)}{U(\tau_0)} \frac{T_i}{T_0} \right\rfloor T_0 \tag{21}$$

Given that Eq. (21) is equivalent to Eq. (19), the claim follows. D

The next theorem states our seventh schedulability test. The arguments behind this test follow from Theorem 3 by Liu and Layland [15].

Theorem 8 (Test 7) *Let $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\}$ be a set of tasks scheduled by EDF and let τ_0 be the highest priority task in the system such that $T_0 \leq \min_{\tau_i \in \Gamma}(T_i)$. There is no deadline miss provided that Eq.* (22) *holds.*

$$U(\Gamma) + U(\tau_0) \leq \min_{\tau_i \in \Gamma}\left(\beta(\tau_0, T_i)\right) \tag{22}$$

Where

$$\beta(\tau_0, T_i) = \begin{cases} 1 + U(\tau_0)\left(1 - \frac{T_0}{T_i}\lceil \frac{T_i}{T_0} \rceil\right) & \textit{if } U(\tau_0) \leq \frac{T_i}{T_0} - \lfloor \frac{T_i}{T_0} \rfloor \\ \frac{T_0}{T_i}\lfloor \frac{T_i}{T_0} \rfloor + U(\tau_0)\left(1 - \frac{T_0}{T_i}\lfloor \frac{T_i}{T_0} \rfloor\right) & \textit{otherwise} \end{cases} \tag{23}$$

Proof Let τ_0 and τ_i' be two tasks with their periods being T_0 and T_i and their run-times being C_0 and $U(\Gamma)T_i$, respectively. According to the RM priority assignment, τ_0 has higher priority than that of τ_i'. In a critical time zone of τ_i', there are $\lceil \frac{T_i}{T_0} \rceil$ requests for τ_0. Let us now adjust C_i' to fully utilize the available processor time within the critical time zone. Two cases occur:

Case 1. The run-time C_0 is short enough that all requests for τ_0 within the critical time zone of T_i are completed before the second request for τ_i'. That is,

$$\left\lfloor \frac{T_i}{T_0} \right\rfloor T_0 + C_0 \leq T_i \quad \Rightarrow \quad C_0 \leq T_i - T_0 \left\lfloor \frac{T_i}{T_0} \right\rfloor \quad \Rightarrow \quad U(\tau_0) \leq \frac{T_i}{T_0} - \left\lfloor \frac{T_i}{T_0} \right\rfloor$$

Thus, the largest possible value of C_i' is

$$C_i' \leq T_i - C_0 \left\lceil \frac{T_i}{T_0} \right\rceil$$

The corresponding processor utilization factor is

$$\beta(\tau_0, T_i) = 1 + U(\tau_0)\left(1 - \frac{T_0}{T_i}\left\lceil \frac{T_i}{T_0} \right\rceil\right)$$

Case 2. The execution of the $\lceil \frac{T_i}{T_0} \rceil$th request for τ_0 overlaps the second request for τ'. In this case

$$U(\tau_0) > \frac{T_i}{T_0} - \left\lfloor \frac{T_i}{T_0} \right\rfloor$$

It follows that the largest possible value of C'_i is

$$C'_i = -C_0\left\lfloor \frac{T_i}{T_0} \right\rfloor + T_i\left\lfloor \frac{T_i}{T_0} \right\rfloor$$

and the corresponding utilization factor is

$$\beta(\tau_0, T_i) = \frac{T_0}{T_i}\left\lfloor \frac{T_i}{T_0} \right\rfloor + U(\tau_0)\left(1 - \frac{T_0}{T_i}\left\lfloor \frac{T_i}{T_0} \right\rfloor\right)$$

Integrating the values of $\beta(\tau_0, T_i)$ for the two cases above leads to Eq. (23). Using the minimum value of β gives then a least upper bound on the system processor utilization so that the schedulability of all tasks τ'_i is ensured. Since the schedulability of τ'_i implies the schedulability of τ_i by Theorem 4, the theorem follows. □

5 Discussion

In this section some properties and characteristics of the described schedulability tests are discussed. We first present how one test relates to another taking into consideration the *dominance* property. A schedulability test T is said to dominate another test T' if all systems deemed schedulable by T' are also deemed schedulable by T. If tests T and T' dominate one another they are said to be *dominance-equivalent*. Figure 4 illustrates the derived relations between the tests. The dotted triangle indicates a set of tests chosen as the dominant set. In this section we show that Tests 2, 3 and 7 dominate all other and so they can be jointly used as an efficient schedulability test. Table 1 also summarizes the dominance relation discussed here as long as some characteristics of the tests.

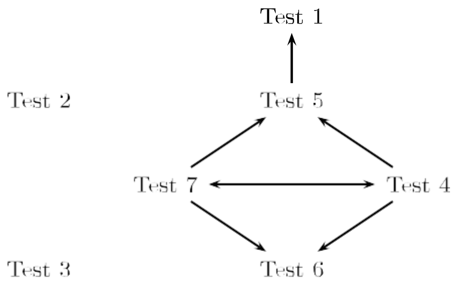

Fig. 4 Dominance relation between the derived schedulability tests

Table 1 Summary of the proposed schedulability tests

Test	Expression	Notes
Test 1 (Theorem 1)	$(\frac{T_0}{\min\limits_{\tau_i \in \Gamma}(T_i)} + 1)U(\tau_0) + U(\Gamma) \leq 1$	Dominated by Tests 4, 5, and 7. No assumption on T_0.
Test 2 (Theorem 2)	$U(\tau_0) + \sum\limits_{\tau_i \in \Gamma} \frac{T_i}{\lfloor \frac{T_i}{T_0} \rfloor T_0} U(\tau_i) \leq 1$	Better results when T_0 divides T_i. It is assumed that $T_0 \leq T_i$.
Test 3 (Theorem 3)	$(\frac{U(\Gamma)}{\lfloor \frac{\min_{\tau_i \in \Gamma}(T_i)}{T_0} \rfloor} + 1)U(\tau_0) + U(\Gamma) \leq 1$	Better results when $\min(T_i) \gg T_0$. It reduces to Eq. (15) when $T_0 < 2\min(T_i)$. It assumes that $T_0 \leq T_i$.
Test 4 (Theorem 5)	$R_i = U(\Gamma)T_i + \lceil \frac{R_i}{T_0} \rceil C_0$	Fixed-point equation to be solved for each $\tau_i \in \Gamma$. Dominance-equivalent to Test 7. No assumption on T_0.
Test 5 (Theorem 6)	$\max\limits_{\tau_i \in \Gamma} (\lceil \frac{T_i}{T_0} \rceil \frac{T_0}{T_i}) U(\tau_0) + U(\Gamma) \leq 1$	Dominated by Tests 4 and 7. No assumption on T_0.
Test 6 (Theorem 7)	$\max\limits_{\tau_i \in \Gamma} (\frac{T_i}{\lfloor \frac{1-U(\Gamma)}{U(\tau_0)} \frac{T_i}{T_0} \rfloor}) \frac{1}{T_0} \leq 1$	Dominated by Tests 4 and 7. No assumption on T_0.
Test 7 (Theorem 8)	$U(\Gamma) + U(\tau_0) \leq \min\limits_{\tau_i \in \Gamma}(\beta(\tau_0, T_i))$	Dominance-equivalent to Test 4. It assumes that $T_0 \leq T_i$.

Another issue addressed in this section is the maximum processor utilization that can be ensured by the scheduling model assumed in this paper. We show that systems that use up to $2(\sqrt{2} - 1)$ of the processor are guaranteed to be schedulable. This is equivalent to a system with two tasks scheduled by the RM policy. That is, having *n* EDF-scheduled tasks suffering the interference of a high priority task reduces about 17 % the schedulability bound of the system as compared to what EDF would provide. Despite this bound being tight, we show that much higher bounds can be achieved when using the schedulability tests described in this paper since they explore specific characteristics of the system under analysis.

5.1 Test dominance

As noticed, some tests (2, 3, and 7) were derived based on the assumption that the highest priority task τ_0 has the shortest period whereas other tests (1, 4, 5, and 6) do not need such a restriction. In order to establish dominance relations between the tests, though, a standard model for the system is needed and so in this section we take the assumption on minimum T_0 for granted. Since assigning lower priorities to tasks with shorter periods reduces the schedulability of fixed-priority systems, this restriction does not compromise the results we derive here. After deriving dominance relations for the seven tests, this section ends presenting a set of three tests (out of seven) that can be jointly used for checking system schedulability.

First, we observe that Tests 1–3 do not dominate one another since there are systems that pass the schedulability condition stated by one but not by the others'. Three simple examples illustrate this. System $\tau_0 = (1.1, 11)$ and $\Gamma = \{\tau_1 = (25.8, 30)\}$ is validated by Test 1 but not by the other two tests. Only Test 2 deems system $\tau_0 = (0.1, 1)$ and $\Gamma =$

$\{\tau_1 = (9, 10)\}$ schedulable. Also, Tests 1 and 2 fail on system substitute $\tau_0 = (8.8, 11)$ and $\Gamma = \{\tau_1 = (3, 30)\}$ for $\tau_0 = (0.5, 2)$ and $\Gamma = \{\tau_1 = (1.8, 3)\}$ whereas Test 3 is capable of successfully checking that it is schedulable.

Dominance relations considering the other four tests can be derived, as shown hereafter.

Theorem 9 *The schedulability test stated in Theorem* 6 (*Test* 5) *dominates the schedulability test stated in Theorem* 1 (*Test* 1) *but the reverse does not hold.*

Proof As

$$\begin{aligned}\max_{\tau_i\in\Gamma}\left(\left\lceil\frac{T_i}{T_0}\right\rceil\frac{T_0}{T_i}\right)U(\tau_0)+U(\Gamma) &\le \max_{\tau_i\in\Gamma}\left(\left(\frac{T_i}{T_0}+1\right)\frac{T_0}{T_i}\right)U(\tau_0)+U(\Gamma) \\ &= \left(\max_{\tau_i\in\Gamma}\left(\frac{T_0}{T_i}\right)+1\right)U(\tau_0)+U(\Gamma) \\ &= \left(\frac{T_0}{\min_{\tau_i\in\Gamma}(T_i)}+1\right)U(\tau_0)+U(\Gamma)\end{aligned}$$

we have that Test 5 dominates Test 1. To see that the Test 1 does not dominate Test 5, consider $\Gamma = \{\tau_1 = (9, 10)\}$, and $\tau_0 = (0.1, 1)$. For such a system, Test 1 fails but Test 5 does not. D

It can also be seen that Test 7 does not dominate Test 2. Using $\tau_0 = (1, 2)$ and $\Gamma = \{\tau_1 = (0.5, 3), \tau_2 = (1.5, 6)\}$, from Test 2 we have that $0.5 + \frac{5}{12} < 1$. That is $\{\tau_0\} \cup \Gamma$ is deemed schedulable. However, from Test 7 we have to check the system $\{\tau_0\} \cup \{\tau_1^{\dagger} = (1.25, 3)\}$, which is not schedulable.

Test 7, on the other hand, dominates Tests 1, 4–6.

Theorem 10 *The schedulability test stated in Theorem* 8 (*Test* 7) *is dominance-equivalent to the test stated in Theorem* 5 (*Test* 4) *and dominates the schedulability tests stated in Theorems* 6 *and* 7 (*Tests* 5 *and* 6, *respectively*).

Proof The dominance-equivalence between Tests 7 and 4 come from the fact that they both use exact schedulability conditions to check whether or not n two-task systems are schedulable and then rely on the result from Theorem 4 to infer the schedulability of the original system. Although Tests 5 and 6 also rely on Theorem 4, they use sufficient conditions to check the schedulability of the n two-task systems and so they are dominated by Test 7 (and Test 4). As Test 5 dominates Test 1 (Theorem 9), Test 7 also does. D

The above results imply that one may consider using only three tests to check the schedulability of the system, namely Tests 2, 3, and 7 (or equivalently Test 4 instead of Test 7). If a given system is validated by at least one of these tests, then the system is guaranteed to be schedulable. If one test detects that the system is schedulable, carrying out the others is not necessary.

5.2 Least upper bound on system utilization

Least upper bounds on processor utilization for schedulability tests are a widely accepted way of expressing the limits for using processor resources. For example, the Liu and Layland test [15] for RM states that any system with n tasks and utilization up to $n(2^{\frac{1}{n}} - 1)$ is schedulable. This is a decreasing function approaching to about 69.3 % for large values of n. For EDF, the bound is known to be 100 % independently of the number of tasks. Based on Theorem 4 we derive a processor bound for the scheduling model assumed in this paper. We also show that this bound is tight meaning that there exist non-schedulable systems with utilisation just above the derived bound.

Corollary 1 *The least upper bound on processor utilization for a set of n tasks Γ scheduled by EDF under the interference of task τ_0 running at the highest priority level is $2(\sqrt{2}-1)$ when T_0 is not greater than the period of the other n tasks. Moreover, this bound is tight.*

Proof We give sets Γ composed of n tasks for which the stated bound holds. For $n = 1$, it follows that the system scheduler behaves as if it were scheduled by RM and so the bound corresponds to the Liu and Layland bound for a system with two tasks [15]. Now consider $n > 1$. Theorem 4 tells us that one can check the schedulability of the system using a schedulability test applied to a 2-task system, with task utilization equal to $U(\tau_0)$ and $U(\Gamma)$, scheduled by RM. By applying once more the Liu and Layland test, we find the desired bound.

To see that this bound is tight it suffices to show a non-schedulable system that uses barely more than $2(\sqrt{2}-1)$ of processor resources. Let the highest priority task in this system be $\tau_0 = (\sqrt{2}-1, 1)$ and consider Γ composed of the following n tasks: $\tau_n = (2-\sqrt{2}, \sqrt{2})$ and $\tau_i = (\frac{E}{n-1}, \sqrt{2})$, $i = 1, 2, \ldots, n-1$, where $E > 0$ is an arbitrarily small constant. The utilization of this system is

$$U(\tau_0) + U(\Gamma) = \sqrt{2} - 1 + (n-1)\frac{\epsilon}{(n-1)\sqrt{2}} + \frac{2-\sqrt{2}}{\sqrt{2}} = 2(\sqrt{2}-1) + \frac{\epsilon}{\sqrt{2}}$$

The defined system may miss some deadlines since within time interval $\sqrt{2}$ τ_0 must be executed twice and all tasks in Γ once, which takes $\sqrt{2} + E$ of execution time. D

The above corollary tells us that by using the scheduling model described in this paper one can reach the same schedulability bounds provided by RM as if there were only two tasks in the system. This per se is a good result. The known schedulability tests based on processor utilization for RM give bounds much lower than $2(\sqrt{2}-1)$ (recall Eqs. (14) and (15)). Although the found bound of $2(\sqrt{2}-1)$ is tight, it is likely that schedulable systems use more than that. The schedulability tests derived in this paper are capable of identifying a large portion of those high utilization schedulable systems since they are able to explore some specific characteristics of the tasks under analysis. In the next section we will show that indeed the derived tests offer a very good average performance.

6 Assessment

In this section we compare the derived tests in terms of schedulability by applying them to a large set of synthetic systems. We first present the results given by Tests 1–3, and 5–7 (Sect. 6.1). Test 4 was not considered because it is dominance-equivalent to Test 7, as shown in Theorem 10. We then compare in Sect. 6.2 the results obtained by dominant Tests 2, 3, and 7 against what would be given by an exact (but of exponential complexity) schedulability test.

For the evaluations, we generated 66,000 task sets with $n = 2, 4, 8, 16, 32, 64$ tasks each. The values for the task set utilization were varying between 70 % and 100 %. For each value of task set utilization 1,000 task sets were generated. All these synthetic task sets were generated according to a random task generator described elsewhere [4], a procedure that ensures the uniformity of task set utilization. Task periods were generated according to a log-uniform integer distribution in the interval $[10, 1000]$, as recommended by other authors [7].

6.1 Comparing schedulability Tests 1–7

Figure 5 depicts the behavior of the proposed schedulability tests in terms of *success ratio*, that is, the percentage of task sets accepted as schedulable. For the sake of comparison the performance of the test given by Eq. (15) was also plotted. Equation (14) was not considered

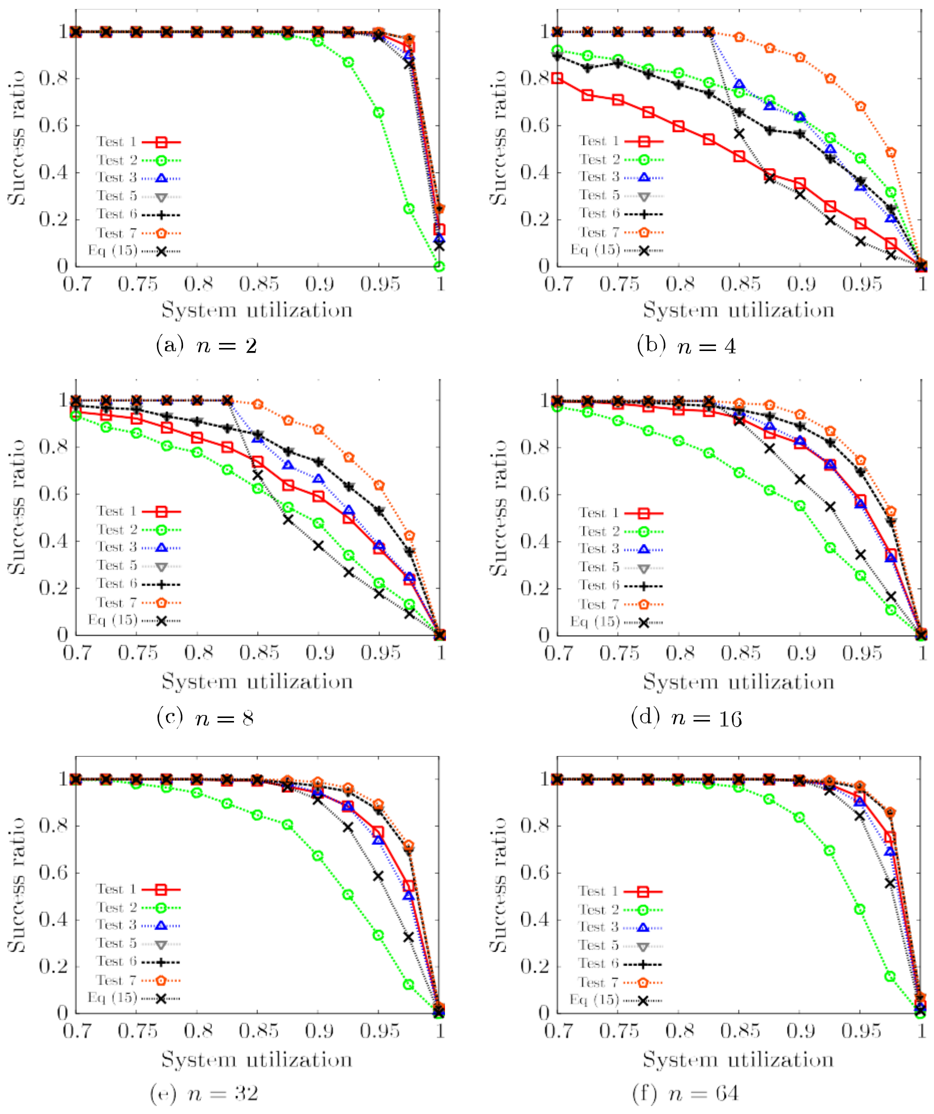

Fig. 5 Comparison between the proposed schedulability tests for $n = 2, 4, 8, 16, 32, 64$

since it leads to a bound of about $2(\sqrt{2}-1) \approx 0.83$, which is not better than the values found by the other tests.

As can be seen in the figure, the larger the task set, the better the performance of the schedulability tests. This is because the generated task utilization of each task tends to be lower when n increases. In particular, the lower the utilization of the highest priority task, the lower its interference in the execution of the other tasks. On average, Eq. (15) behaves worse than Tests 1–7. Also, Tests 1 and 3 have similar performance and Tests 5 and 6 perform equivalently giving better results than Tests 1 and 3 for $n > 2$. Note that these tests use a relation between the periods of two tasks only whereas Test 2 inflates the utilization

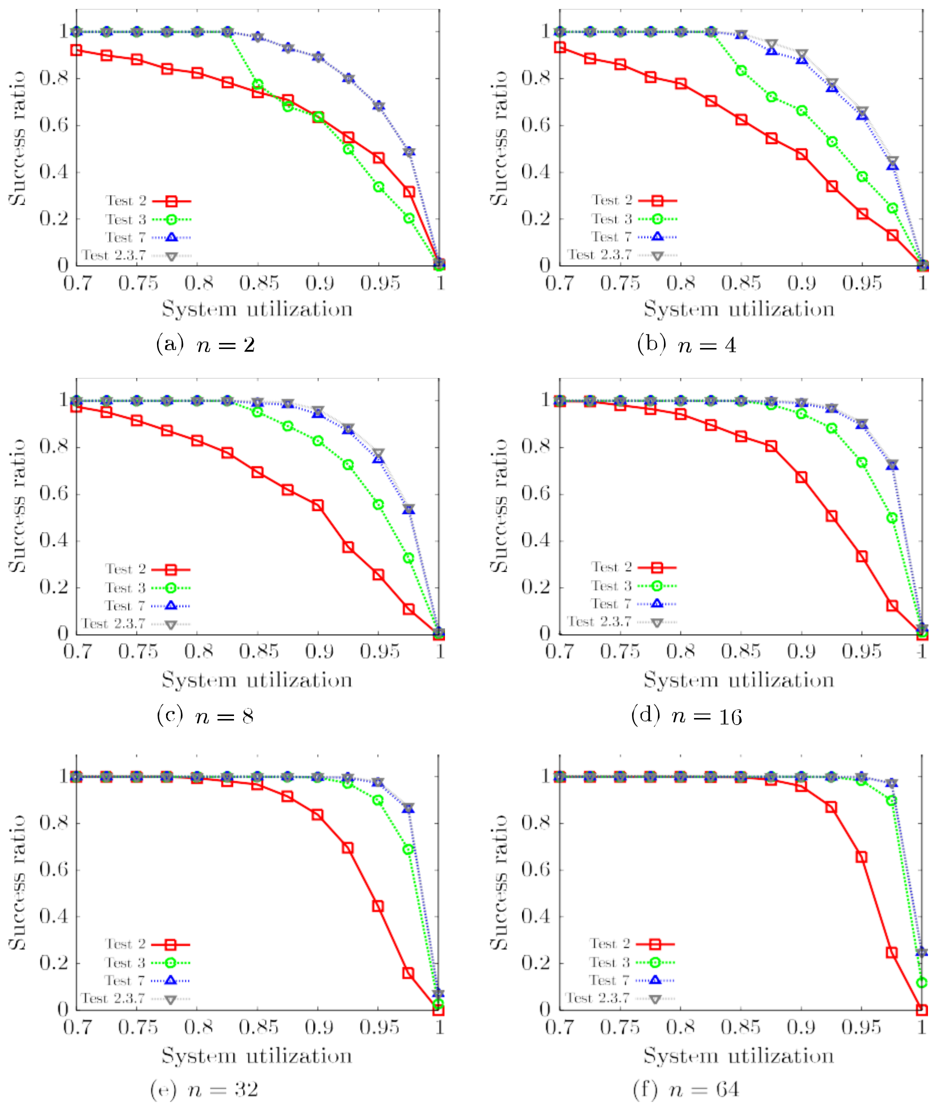

Fig. 6 Comparison between the combined Test 2.3.7 against the Tests 2, 3 and 7 taken individually for $n = 2, 4, 8, 16, 32, 64$

of the task set considering all tasks. As for Test 7, it can be seen that it usually gives better results than the other tests. However, Tests 7 is outperformed for some scenarios, as expected by the known dominance relations.

The results presented in the graphs motivate the combination of the proposed tests so as to obtain the best performance out of them. That is, a combined test accepts a system as schedulable if at least one of the tests used in the combination does. Since Tests 2, 3, and 7 make the other redundant due to the dominance relation, we chose their combination. The results are plotted in the graphs of Fig. 6. As can be seen, the combined test, named Test 2.3.7, gives better performance than the other three tests considered individually, as expected.

6.2 Comparing the proposed schedulability tests against an exact test

A very efficient exact schedulability test for EDF-scheduled systems, called QPA, has recently been described [22]. We chose to use an improved version of QPA [21] so that we could check the performance of our proposed sufficient tests in terms of detected schedulable task sets. QPA can deal with the assumed model by artificially making the relative deadline of τ_0 equal to C_0, as has been used for the C=D algorithm [7]. Although QPA presents worst-case exponential complexity, it has been shown that it performs very well on average. Indeed, considering synthetic generated task sets with at most 64 tasks, we observed that QPA took at most 276 times as much as the time to run our combined Test 2.3.7.

Figure 7 summarizes the obtained results. Its *y*-axis represents the percentage of feasible task sets that are considered schedulable by the combined Test 2.3.7, which was applied to systems that were considered schedulable by the QPA exact test. As can be seen in the figure, the higher the value of *n* the more precise is the performance of the proposed tests, which is in line with the results shown in Fig. 5. The exception is for $n = 2$ because Test 7 works as an exact test when the system has only two tasks. It can be also seen that up to values of utilization around 0.95, the performance Test 2.3.7 is comparable to that of an exact test for $n \geq 32$ tasks. This is a very good result since polynomial tests can be carried out on-line, which may not be the case for exact schedulability tests.

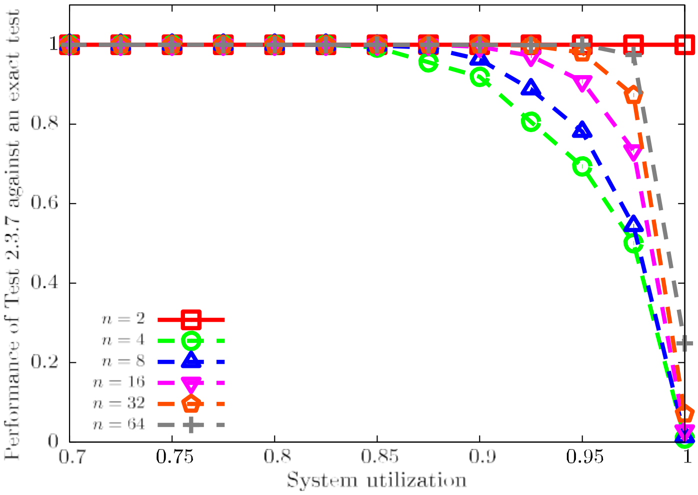

Fig. 7 Performance comparison of the proposed sufficient tests against an exact schedulability test

7 Conclusion

We have derived seven new sufficient schedulability tests for uniprocessor real-time systems. The considered system is composed of tasks scheduled by EDF which suffer interference of the execution of a high priority task, which models urgent routines that need to be executed within minimum delay. All proposed tests are proved correct. We have shown that three of these tests dominate the others and so they can be jointly used to check schedulability of the system.

We also have shown that the assumed scheduling model provides at least $2(\sqrt{2} - 1) \approx 1.83$ of processor utilization. Experiment results have indicated that the proposed tests give much higher schedulability bounds, though. Schedulable systems that use up to around 95 % of processor are identified as such. As the system model considered here is found in practice when urgent routines are not to be delayed by application tasks, the proposed tests have relevance from both theoretical and practical perspective.

Acknowledgements This work was partially supported by the Brazilian funding agencies CAPES and CNPq.

References

1. Almeida L, Pedreiras P (2004) Scheduling within temporal partitions: response-time analysis and server design. In: Proc of the 4th ACM international conference on embedded software. ACM, New York, pp 95–103
2. Audsley NC, Burns A, Richardson M, Tindell K, Wellings AJ (1993) Applying new scheduling theory to static priority pre-emptive scheduling. Softw Eng J 8(5):284–292
3. Baruah S, Mok A, Rosier L (1990) Preemptively scheduling hard-real-time sporadic tasks on one processor. In: Proc of the 11th IEEE real-time systems symposium, pp 182–190
4. Bini E, Buttazzo GC (2005) Measuring the performance of schedulability tests. Real-Time Syst 30:129–154
5. Bini E, Buttazzo GC, Buttazzo GM (2003) Rate monotonic analysis: the hyperbolic bound. IEEE Trans Comput 52(7):933–942
6. Bletsas K, Andersson B (2011) Preemption-light multiprocessor scheduling of sporadic tasks with high utilisation bound. Real-Time Syst 47:319–355
7. Burns A, Davis RI, Zhang FPW (2011) Partitioned edf scheduling for multiprocessors using a C=D scheme. Real-Time Syst 48:3–33
8. Davis RI, Burns A (2008) An investigation into server parameter selection for hierarchical fixed priority pre-emptive systems. In: International conference on real-time and network systems, pp 19–28
9. Facchinetti T, Buttazzo GC, Marinoni M, Guidi G (2005) Non-preemptive interrupt scheduling for safe reuse of legacy drivers in real-time systems. In: Proc of the 17th euromicro conference on real-time systems, pp 98–105
10. Feng XA, Mok AK (2002) A model of hierarchical real-time virtual resources. In: Proc of the 23rd IEEE real-time systems symposium, pp 26–35
11. Gonzalez-Harbour M, Palencia JC (2003) Response time analysis for tasks scheduled under edf within fixed priorities. In: Proc of the 24th IEEE real-time systems symposium, pp 200–209
12. Jeffay K, Stone D (1993) Accounting for interrupt handling costs in dynamic priority task systems. In: Proc of the 14th IEEE real-time systems symposium, pp 212–221
13. Lima G, Burns A (2003) An optimal fixed-priority assignment algorithm for supporting fault tolerant hard real-time systems. IEEE Trans Comput 52(10):1332–1346
14. Lima G, Burns A (2005) Scheduling fixed-priority hard real-time tasks in the presence of faults. In: Proc of 2nd Latin-American symposium on dependable computing. LNCS, vol 3747. Springer, Berlin, pp 154–173
15. Liu CL, Layland JW (1973) Scheduling algorithms for multiprogram in a hard real-time environment. J ACM 20(1):46–61
16. Mercer CW, Savage S, Tokuda H (1994) Processor capacity reserves: operating system support for multimedia applications. In: Proc of the international conference on multimedia computing and systems, pp 90–99

17. Santos JA Jr, Lima G (2012) Sufficient schedulability tests for edf-scheduled real-time systems under interference of a high priority task. In: Proc of the 2nd Brazilian symposium on systems engineering, SBC, Natal, Brazil, pp 131–136
18. Santos JA Jr, Lima G, Bletsas K (2013) On the processor utilisation bound of the C=D scheduling algorithm. In: Audsley N, Baruah S (eds) Real-time systems: the past, the present and the future. CreateSpace independent publishing platform, pp 119–132
19. Shin I, Lee I (2003) Periodic resource model for compositional real-time guarantees. In: Proc of the 24th IEEE international real-time systems symposium, pp 2–13
20. Spuri M, Buttazzo GC (1996) Scheduling aperiodic tasks in dynamic priority systems. Real-Time Syst 10(2):1–32
21. Zhang F, Burns A (2009) Improvement to quick processor-demand analysis for edf-scheduled real-time systems. In: Proc of the of the 21st euromicro conference on real-time systems, pp 76–86
22. Zhang F, Burns A (2009) Schedulability analysis for real-time systems with edf scheduling. IEEE Trans Comput 58(9):1250–1258