Simple and General Methods for Fixed-Priority Schedulability in Optimization Problems

Paolo Pazzaglia, Alessandro Biondi, and Marco Di Natale
Scuola Superiore Sant'Anna, Pisa, Italy
E-mail: {paolo.pazzaglia, alessandro.biondi, marco}@santannapisa.it

***Abstract*—This paper presents a set of sufficient-only, but accurate schedulability tests for fixed-priority scheduling. The tests apply to the general case of scheduling with constrained deadline where tasks can incur in blocking times, be subject to release jitters, activated with fixed offsets, or involved in transactions with other tasks. The proposed tests come in a linear closed-form with a number of conditions polynomial in the number of tasks. All tests are targeted for use when encoding schedulability constraints within Mixed-Integer Linear Programming for the purpose of optimizing real-time systems (e.g., to address task partitioning in a multicore system). The tests are evaluated with a large-scale experimental study based on synthetic workload, revealing a failure rate (with respect to the state-of-the-art reference tests) of less than 1% in average, and at most of 2% in a very small number of limit-case configurations.**

I. Introduction

Timing and schedulability analysis is often seen as a verification and analysis tool, to be used to certify that a given system configuration is safe and feasible with respect to timing constraints. However, especially in the context of systems that are feature rich and considering the availability of platforms with multiple cores and multiple processing nodes, the burden is often on the shoulders of the designer, who has to define the execution platform and the software configuration, including the design and allocation of the tasks. In this context, the results of timing analysis should be used to define constraints in a design optimization process, helping select the feasible configuration with the best performance.

Examples of problems that require such a formulation are the need to move applications from single core to multicores or even manycores, partitioning the software tasks, or the design of logical execution time frames, to restore determinism in the communication between tasks in multicore systems [1], [2].

Feasibility constraints have been encoded for use in design optimization in previous works. However, a common drawback of most formulations is their complexity, which may prevent an effective use in the optimization of problems of industrial size. This is the reason why approximate (sufficient-only) formulations with good accuracy could be quite helpful in allowing a mathematical formalization and solution to several problems of practical relevance.

This paper. In this work, we provide several extensions to a schedulability formulation that was found to be effective in the case of Rate Monotonic fixed-priority scheduling [3] and only required a number of constraints that is quadratic in the number of tasks. We show how the concept can be extended to deal with a large number of cases of practical interest including: scheduling with jitter, offsets, blocking times and even transactional scheduling, in which tasks are bound to execute in precedence constrained chains. Most important, we provide a number of experiments that show how the proposed formulation is extremely tight for task sets with randomly generated periods (the most used benchmark in research literature) with a pessimism that is always below 2% and only for very heavily loaded systems. In addition, when periods are selected from a set of pseudo-harmonic options, the error drops to below 1%.

The rest of the paper is organized as follows: an analysis of existing literature on the subject is followed by a definition of the system model in Section II; the available analysis methods are presented in Section III, and the proposed extensions in Section IV. The experimental evaluation for the test cases is discussed in Section V, and Section VI ends the paper.

A. Related Work

The worst-case response time R_i of a real-time task is often computed with an iterative procedure [4] that is particularly inefficient when implemented in a standard mathematical optimization framework. The problem is discussed in [5], where several options are considered. Possible approximations include sufficient-only tests like the Liu and Layland utilization bound [6] or the Hyperbolic bound [7], or the response time bound in [8]. However, these sufficient bounds can be quite pessimistic and the resulting solutions may be often sub-optimal. More recently, efforts have been spent in finding more efficient tests [9] and tighter upper bounds with quadratic-time complexity [10], [11]. Some of these bounds applies also to tasks with jitter, blocking, and cache-related preemption delays, but are not expressed as linear conditions and hence, not suitable for being encoded in optimization formulations such as MILPs.

The test proposed in [12] is the source of a number of other solutions, exact and approximate. In this work, the number of conditions to be checked to ensure schedulability is found to be finite and defined by a set of *schedulability points*. The condition for each checking point can then be encoded as a constraint to be combined in a disjunction with the constraints from the other points. The set of points in [12] was first found to be redundant in [13], where a subset of points for a necessary and sufficient condition is defined. Finally, in [5], even the set in [13] is found to be redundant and possibly further reducible by solving a set of optimization problems.

However, even if these points are now a relatively small subset of the original set, the encoding can still be prohibitively large, especially considering that the computation of the set of schedulability points requires presolving a set of additional optimization problems.

Given that feasibility can be defined as the OR-combination of constraints, the authors of [13] suggest that sufficient-only tests can be easily derived by reducing the set of points to be checked, but how to select the points to be retained is not discussed. This selection is exploited in [3], where the authors present a sufficient-only analysis for the schedulability of Rate Monotonic task sets, restricting the test to a very small subset of scheduling points. The proposed test is noticeably less pessimistic than existing sufficient ones, it is linear, and shows a performance extremely close to the exact test.

In this paper, we show how an intuition similar to the one proposed in [3] is applicable to a wide range of different task sets (including constrained deadlines, offsets, jitters, and blocking times) and fixed-priority schedulers. The resulting tests are polynomial-time in most cases, maintain low pessimism, good scaling properties and are directly suitable to be used in optimization algorithms as a set of linear constraints.

II. System Model

We consider a task set $\mathcal{T}$, running on a uniprocessor, composed of n periodic tasks τ_i, with $i = 1, \ldots, n$. The scheduling of the task set is preemptive, and tasks are ordered following a Fixed Priority algorithm (thus including Rate and Deadline Monotonic). The task index reflects the given ordering, where τ_1 has highest priority. In addition, $hp(i)$ denotes the set of tasks with higher priority than τ_i, while $lp(i)$ is the set of lower priority tasks. Arbitrary tie-breaking for tasks with the same priority is assumed (i.e., in the worst-case they can interfere each other). To lighten the adopted notation, interfering tasks with the same priority of the one under analysis are simply denoted as higher-priority tasks.

Each task τ_i is defined by a triplet $\{C_i, D_i, T_i\}$, where C_i represents its worst-case execution time, D_i is the relative deadline and T_i the period. The assumption of constrained deadline, i.e. $D_i \leq T_i$, will be applied throughout the paper. R_i is the worst-case response time experienced by τ_i. If the condition $R_i \leq D_i$ holds for all $\tau_i \in \mathcal{T}$, the task set is deemed as *schedulable*; otherwise it is not schedulable.

While illustrating the schedulability tests in the following sections, the complexity of the task set under analysis is gradually increased. We will assume that the tasks $\tau_i \in \mathcal{T}$ may experience release jitter with values in $[0, J_i]$, that their periodic activation may be defined with a fixed initial offset $\phi_i \in [0, T_i)$, and that each task τ_i may also suffer from blocking bounded by a quantity B_i from lower priority tasks.

Furthermore, this work also considers the *transactional* task model [14], in which a system consists of a set of N_T *transactions* $\Gamma_1, \ldots, \Gamma_{N_T}$. Each transaction Γ_i is defined as a set of N_i tasks, where each task is denoted as $\tau_{i,j}$, and is periodically activated with period T_i. Each task $\tau_{i,j}$ is activated with a fixed offset $\phi_{i,j} < T_i$ from the activation time of the transaction, and can incur in a release jitter bounded by $J_{i,j}$. Tasks are further characterized by a worst-case execution time $C_{i,j}$ and a deadline $D_{i,j} \leq T_i$ such that $J_{i,k} \leq D_{i,j} - C_{i,j}$. The tasks belonging to the same transaction share the same priority. Hence, in this case, the set $hp(i)$ is also used to denote high-priority transactions.

III. Analysis of fixed-priority scheduling

The classical analysis of fixed-priority tasks executed upon a single processor can be performed by computing the response time R_i of each task and then verifying whether the inequality $R_i \leq D_i$ holds for all of them. The response time of a task τ_i is given by the least positive fixed point of the following recursive equation [4]: $R_i = C_i + \sum_{\tau_j \in hp(i)} \lceil R_i/T_j \rceil C_j$.

Solving this equation is a NP-HARD problem as demonstrated by Eisenbrand and Rothvob [15]. Despite its computational complexity, this equation is also not well suited for being used in a mathematical programming framework to optimize or design a real-time system. Indeed, encoding this equation in a MILP requires introducing *integer* variables to model the ceiling term, which may significantly limit scalability as the number of tasks increases. An alternative formulation for an exact schedulability test under fixed-priority scheduling consists in verifying the following condition for each task τ_i:

$$\exists t \in [0, D_i] \mid C_i + \sum_{j \in hp(i)} \lceil t/T_j \rceil C_j \leq t. \tag{1}$$

Lehoczky et al. [12] demonstrated that the test of Equation (1) does not require dealing with an impractical continuum, i.e., the range $[0, D_i]$, but rather it is sufficient to check a limited number of points for an exact schedulability test. The set is defined by $\mathcal{S}_i = \{aT_k \,|\, k = 1, \ldots, i;\ a = 1, \ldots, \lfloor T_i/T_k \rfloor\}$, with a size that depends on the ratio of the periods of the tasks. The elements in $\mathcal{S}_i$ are called *schedulability points*. Checking that condition (1) holds for at least one $t \in \mathcal{S}_i, t \leq D_i$ for each $\tau_i \in \mathcal{T}$ is then a sufficient and necessary test for schedulability [12]. Anyway, the complexity of the exact analysis of fixed-priority scheduling remains NP-HARD as recently proven by Ekberg and Yi [16]. Furthermore, the use of this test in a MILP formulation requires one constraint for each point in the set $\mathcal{S}_i$ and combining these constraints in a single disjunction. A reduced sets of check points with respect to $\mathcal{S}_i$ has been found in [13], and finally in [5] a procedure for a further reduction of the set of points for a necessary and sufficient condition is proposed. Both methods improve substantially on the size of the constraints, but can still result in a quite large set in the worst case (formally, still of exponential size). In addition, the reduced set in [5] requires solving a set of nested optimization problems to be computed.

A simpler but tight sufficient-only condition could be useful in many cases. This condition can be computed by leveraging an observation originally provided in [13]. Given that feasibility is computed as the OR-combination of a set of conditions, dropping some of them (hence, some of the schedulability points) results in a sufficient-only test. Therefore, an efficient schedulability test can be defined by finding a subset of $\mathcal{S}_i$, possibly with a polynomial size in the number of tasks, which guarantees schedulability performance very close to the exact

test, i.e., failing in detecting a schedulable task set only in a very small number of cases.

In the case of tasks with a Rate Monotonic priority assignment, Park and Park [3] showed that the adoption of a set of check-points with a quadratic size in the number of tasks results in a near-exact schedulability test. That set, denoted hereafter as $\mathcal{V}_i^{\text{RM}}$, is defined for each task $\tau_i \in \mathcal{T}$ as follows:

$$\mathcal{V}_i^{\text{RM}} = \left\{ aT_k \,|\, k = 1, \ldots, i;\ a = \lfloor T_i/T_k \rfloor \right\}. \tag{2}$$

Intuitively, this set contains the points in the time interval $[0, T_i]$, when the *last activation* of each high-priority task occurs, under the assumption of synchronous release and tasks released as soon as possible. $\mathcal{V}_i^{\text{RM}}$ is indeed a subset of $\mathcal{S}_i$. The use of this set transforms Equation (1) into an $\mathcal{O}(n^2)$ number of linear conditions, and is suitable for use in MILP formulations in which the task periods are fixed.

Despite the practical usefulness of this approach, unfortunately and to the best of our knowledge, no similar results have been presented under less restrictive assumptions such as (i) tasks with arbitrary priority assignments, (ii) tasks with release jitter or offsets, (iii) tasks that can incur in blocking times, and (iv) task transactions. The next section aims at filling this gap, offering a more general efficient analysis platform for Fixed Priority schedulability based on linear formulas.

IV. Efficient schedulability tests

This section presents a set of schedulability tests for different task models with gradually-increasing complexity. Following, in spirit, the work of Park and Park [3], our objective consists in finding equations for the last activation of high-priority tasks with respect to a scheduling window to be analyzed, which changes model by model. Such points will then be used to build approximate schedulability tests. As it emerges in the following sections, the derivation of such check-points is typically not straightforward, as it requires an explicit consideration of the characteristics of the task sets such as release jitters and offsets. Nonetheless, the approach is general enough to be applied to any fixed priority ordering.

A. Tasks with constrained deadline

First, consider a task set $\mathcal{T}$ with arbitrary offsets and constrained deadlines. In this case, the formulation of the check-points in (2) can easily be extended by restricting the set to the points that lay before the deadline. Hence, the check-points for a $\tau_i \in \mathcal{T}$ is $\mathcal{V}_i^{\text{CD}} = \left\{ aT_k \,|\, k = 1, \ldots, i;\ a = \lfloor D_i/T_k \rfloor \right\} \cup \{D_i\}$.

The corresponding schedulability test is then given by:

$$\exists v \in \mathcal{V}_i^{\text{CD}} \mid C_i + \sum_{j \in hp(i)} \left\lceil \frac{v}{T_j} \right\rceil C_j \leq v. \tag{3}$$

Similarly to [3], the number of linear conditions to be checked for the entire task set $\mathcal{T}$ is $\mathcal{O}(n^2)$.

B. Task sets with release jitter

Consider the case in which each periodic task $\tau_k \in \mathcal{T}$ may incur in a *release jitter* bounded by $J_k \leq (T_k - C_k)$. In this case, the analysis presented above may be optimistic and incorrect. Following the results in [17], to safely analyze tasks with jitters, it is required to study the critical instant in which (i) the task under analysis τ_i, and all high-priority tasks have the first activation with maximum jitter at the same time; and (ii) all successive instances are released with zero jitter. This scheduling pattern is denoted as $\mathcal{J}$.

Considering $\mathcal{J}$, our objective is to propose an approximate test based on the check-points that result from the last activation of high-priority tasks within the scheduling window under analysis. The test is derived for the case of constrained-deadline tasks (no multiple pending jobs of the same task can exist at the critical instant). The check-points of interest strongly depend on the maximum release jitters of tasks, and can be derived with the following theorem.

Theorem 1. *Consider a task $\tau_i \in \mathcal{T}$ under analysis and assume that all tasks in $\mathcal{T}$ are released according to $\mathcal{J}$. Without loss of generality, let $t = 0$ be the time at which the first activations of the tasks occur. The scheduling window under analysis is $[0, D_i - J_i]$. A higher priority task τ_k has more than one activation in $[0, D_i - J_i]$ if $T_k - J_k < D_i - J_i$, and its last activation in the interval is defined as*

$$V_{i,k} = \left\lfloor \frac{D_i - J_i + J_k}{T_k} \right\rfloor T_k - J_k. \tag{4}$$

Proof. After the first activation at $t = 0$, the second activation of task τ_k occurs at time $T_k - J_k$. Hence, τ_k has more than one activation in $[0, D_i - J_i]$ if $T_k - J_k < D_i - J_i$. The first periodic instance of τ_k starts at time $-J_k$. Hence, the length of the interval in which periodic instances of τ_k overlap with the scheduling window under analysis is $D_i - J_i + J_k$. Since there are $\lfloor \frac{D_i - J_i + J_k}{T_k} \rfloor$ activations of τ_k that are fully-contained in this interval, the last one occurs at the time defined by (4). □

The set of check-points for analyzing τ_i can be computed as the union of the points $V_{i,k}$ defined by the above theorem, plus its deadline, i.e.

$$\mathcal{V}_i^J = \bigcup_{k \in hp(i)} \{V_{i,k}\} \cup \{D_i - J_i\}. \tag{5}$$

The resulting schedulability test [17] consists in verifying the following condition for each task $\tau_i \in \mathcal{T}$:

$$\exists v \in \mathcal{V}_i^J \mid C_i + \sum_{j \in hp(i)} \left\lceil \frac{v + J_j}{T_j} \right\rceil C_j \leq v. \tag{6}$$

corresponding to an $\mathcal{O}(n^2)$ number of linear conditions.

C. Task sets with offsets

In several applications, the periodic activation of some tasks is defined with a fixed starting offset; for instance, when it is necessary to ensure an implicit synchronization of events. In these cases, computing the worst-case response time under the synchronous release hypothesis may be too pessimistic.

An effective approach to analyze such task sets is provided by Pellizzoni and Lipari (Theorem 2 in [18]). To analyze a generic task τ_i the possible critical instants and busy periods are constructed by starting with the activation of an instance of generic higher priority task τ_x, and considering all the other tasks phased with the minimum activation offset that they may

experience with respect to any instance of τ_x. The response-time analysis must then be performed by considering each possible activation of a higher priority task τ_x as the beginning (at $t = 0$) of a busy period that is a candidate critical instant. The test is pessimistic, since such minimum offsets may never occur at the same time, but is a good compromise between simplicity and accuracy. Nevertheless, its complexity is still polynomial and an approximate alternative is highly desirable.

The minimum distance between any release of $\tau_x \in \mathcal{T}$ and a the release of a different task τ_k can be computed as [18]

$$\Delta_{x,k} = \phi_k - \phi_x + \left\lceil \frac{\phi_x - \phi_k}{\text{GCD}(T_x, T_k)} \right\rceil \text{GCD}(T_x, T_k), \quad (7)$$

where $\text{GCD}(T_x, T_k)$ is the greatest common divisor between T_x and T_k. Note that $\Delta_{x,k}$ cannot be negative.

Each time a new task τ_x is considered for the start of the busy period, a new task set $\mathcal{T}_x$ is created by transforming the tasks in the original set $\mathcal{T}$, into corresponding tasks τ'_k with offsets computed as $\phi'_k = \Delta_{x,k}$. The set $\mathcal{T}$ is schedulable if all $\mathcal{T}_x$ are schedulable (from [18]).

For each choice of τ_x, it is now possible to identify the check-points that correspond to the last activations of all the high-priority tasks with respect to τ_i.

Theorem 2. *Consider one set $\mathcal{T}_x$ and a task $\tau'_i \in \mathcal{T}_x$ under analysis in a problem window in which τ'_i is released at time ϕ'_i and $[\phi'_i, \phi'_i + D_i]$ is its scheduling window.*

If $\phi'_i < \phi'_k \leq \phi'_i + D_i$, or $\phi'_k \leq \phi'_i \wedge T_k + \phi'_k < \phi'_i + D_i$, then task $\tau'_k \in \mathcal{T}_x$ has at least one activation in $[\phi'_i, \phi'_i + D_i]$, and its last activation within this interval occurs at time

$$V_{i,k,x} = \left\lfloor \frac{D_i + \phi'_i - \phi'_k}{T_k} \right\rfloor T_k + \phi'_k. \quad (8)$$

Proof. There can be two cases: **(i)** the first release of τ'_k happens in $[\phi'_i, \phi'_i + D_i]$; **(ii)** otherwise. In case (i), by definition, there is at least one activation of τ'_k in $[\phi'_i, \phi'_i + D_i]$. The distance between the first activation and the end of the interval of interest is given by $D_i + \phi'_i - \phi'_k$. Task τ'_k has $\lfloor \frac{D_i + \phi'_i - \phi'_k}{T_k} \rfloor$ periodic activations that are fully contained into the scheduling window of τ'_i. Hence, the last activation happens at the time defined by Equation (8). In case (ii), an activation of τ'_k in $[\phi'_i, \phi'_i + D_i]$ is possible only if the first activation of τ'_k happens before the first activation of τ'_i, i.e., $\phi'_k \leq \phi'_i$, and its second activation is within the scheduling window of τ'_i, i.e., $T_k + \phi'_k < \phi'_i + D_i$. Under these conditions, the same reasoning discussed above can be applied to find the time of the last activation of τ'_k in the interval of interest. □

A set of check-points to study the schedulability of task $\tau'_i \in \mathcal{T}_x$ can be computed as the union of all the points $V_{i,k,x}$ provided by the above theorem, plus the deadline of τ'_i

$$\mathcal{V}^{\text{OFF}}_{i,x} = \bigcup_{k \in hp(i)} \{V_{i,k,x}\} \cup \{\phi'_i + D_i\}.$$

Leveraging the set $\mathcal{V}^{\text{OFF}}_{i,x}$, the following schedulability test [18] must be performed for every set $\mathcal{T}_x$ and every task $\tau'_i \in \mathcal{T}_x$:

$$\exists v \in \mathcal{V}^{\text{OFF}}_{i,x} \mid C_i + \sum_{j \in hp(i)} \left\lceil \frac{v - \phi'_j}{T_j} \right\rceil C_j \leq v. \quad (9)$$

If none of these condition is violated, then the original task set $\mathcal{T}$ is schedulable. Since n sets $\mathcal{T}_x$ need to be checked with a quadratic procedure, the total number of conditions is $\mathcal{O}(n^3)$.

D. Task sets with blocking

When considering a task set in which some tasks may suffer *blocking* from lower priority ones, e.g., due to locking protocols, the schedulability tests presented above can be extended by simply inflating C_i by the blocking term B_i. Note that the formulation of the last activation of high-priority tasks before the deadline is not affected by the blocking times. Hence the computed check-points can also be used to implement a schedulability test under blocking. The test has the same complexity as the case without blocking.

E. Transactional task model

Transactional tasks (possibly also scheduled with offsets and release jitter) are particularly interesting when dealing with parallelized computations partitioned on multicore systems. For instance, recent works on the LET model to guarantee causality in multicore scheduling require the partitioning of tasks into periodic sub-tasks (or LET frames) separated by synchronization barriers, where shared data are communicated between the cores [1], [19]. This architecture can be represented as a transactional model.

Following Theorem 2 in [14], a task $\tau_{i,j}$ in a transaction can be analyzed by studying the critical instants arising when:

- $\tau_{i,j}$ is released with maximum jitter $J_{i,j}$ at the beginning of the problem window under analysis ($t = 0$ without loss of generality);
- other tasks $\tau_{i,k}$ ($k \neq j$) belonging to the same transaction Γ_i are consequently released with the corresponding offsets with respect to $\tau_{i,j}$, and no jitter;
- in all the other transactions Γ_x ($x \neq i$), one task $\tau_{x,s}$ is synchronously released with $\tau_{i,j}$ with its maximum jitter, and the remaining ones are released with their offsets with respect to $\tau_{x,s}$, and no jitter.

Let $\Lambda_{i,j}$ be the set of all possible combinations of scheduling patterns according to the above rules. In the following, we denote by $\tau(\lambda)$ the set of tasks that in a pattern $\lambda \in \Lambda_{i,j}$ have the first activation synchronized with $\tau_{i,j}$. The analysis in [14] is based on the derivation of a bound on the maximum interference that can be generated in each pattern $\lambda \in \Lambda_{i,j}$.

Our approach aims at computing the time of the last activation of each high-priority task in all the possible busy periods originating from a candidate critical instant with respect to the task $\tau_{i,j}$ under analysis.

First, we compute the time of the first activation of each high-priority task $\tau_{x,k}$, belonging to an arbitrary transaction Γ_x, and occurring after the release of $\tau_{i,j}$. This instant depends on the task $\tau_{x,s}$ activated synchronously with $\tau_{i,j}$ [14]

$$\Phi_{x,k,s} = T_x - (\phi_{x,s} + J_{x,s} - \phi_{x,k}) \bmod T_x. \quad (10)$$

The interference generated by $\tau_{x,k}$ is then derived by decomposing the part before $\Phi_{x,k,s}$ (also referred to as *carry-*

in by some authors), and denoted by $I^{\text{res}}_{x,k,s}$; and the following interference (computed as in (9))

$$I_{x,k,s}(t) = \begin{cases} I^{\text{res}}_{x,k,s} + \left\lceil \frac{t-\Phi_{x,k,s}}{T_x} \right\rceil C_{x,k} & \text{if } \Phi_{x,k,s} \leq t \\ I^{\text{res}}_{x,k,s} & \text{otherwise,} \end{cases}$$

where t denotes the length of the analysis interval of interest. The carry-in interference can be upper-bounded as

$$I^{\text{res}}_{x,k,s} = \begin{cases} \tilde{C}_{x,k} & \text{if } \Phi_{x,k,s} > (T_x - D_{x,k}) \\ 0 & \text{otherwise,} \end{cases} \quad (11)$$

where $\tilde{C}_{x,k} = \min\{C_{x,k}, (\Phi_{x,k,s} - (T_x - D_{x,k}))\}$.

Next, we compute the check-points of interest. In patterns $\lambda \in \Lambda_{i,j}$, the scheduling window of the task under analysis is $[0, D_{i,j} - J_{i,j}]$. The interval that lasts from the first release of a high-priority task $\tau_{x,k}$ within $[0, D_{i,j} - J_{i,j}]$, and $D_{i,j} - J_{i,j}$ is given by $D_{i,j} - J_{i,j} - \Phi_{x,k,s}$. Hence, there are $\lfloor \frac{D_{i,j} - J_{i,j} - \Phi_{x,k,s}}{T_x} \rfloor$ instances of $\tau_{x,k}$ that are fully-contained into the scheduling window of $\tau_{i,j}$. Consequently, the last activation of $\tau_{x,k}$ in the window of interest is given by

$$V_{i,j,x,k,s} = \left\lfloor \frac{D_{i,j} - J_{i,j} - \Phi_{x,k,s}}{T_x} \right\rfloor T_x + \Phi_{x,k,s}. \quad (12)$$

Clearly, such points are valid only if task $\tau_{x,k}$ has at least one activation in $[0, D_{i,j} - J_{i,j}]$, i.e., $\Phi_{x,k,s} \leq D_{i,j} - J_{i,j}$.

The set of check-points for verifying the schedulability of $\tau_{i,j}$ assuming a scheduling pattern λ is finally given by

$$\mathcal{V}^{\text{T}}_{i,j,\lambda} = \left\{ \bigcup_{\tau_{x,s} \in \tau(\lambda)} \bigcup_{x \in hp(i)} \bigcup_{k=1}^{N_x} \{V_{i,j,x,k,s}\} \right\} \cup \{D_{i,j} - J_{i,j}\}. \quad (13)$$

In conclusion, the proposed analysis consists in leveraging the interference bound $I_{x,k,s}(t)$ and the check-points in (13) for each scheduling pattern to be analyzed. Formally, a task $\tau_{i,j}$ is said to be schedulable if, $\forall \lambda \in \Lambda_{i,j}$,

$$\exists v \in \mathcal{V}^{\text{T}}_{i,j,\lambda} \mid C_{i,j} + \sum_{\substack{\tau_{i,q} \in \Gamma_i \\ q \neq i}} I^{\text{res}}_{i,q,j} + \sum_{\substack{\tau_{x,s} \in \tau(\lambda) \\ x \in hp(i)}} \sum_{k=1}^{N_x} I_{x,k,s}(v) \leq v.$$

where $I^{\text{res}}_{i,q,j}$ is the interference from tasks of the same transaction of $\tau_{i,j}$, computed as in (11). The combinations in $\Lambda_{i,j}$ are $\mathcal{O}(\max_{i=1}^{N_T}\{N_i\}^{N_T})$, and each of them originates $\mathcal{O}(n)$ scheduling points (one for each high-priority task).

V. Experimental Evaluation

Large-scale experiments have been performed to assess the performance of the proposed tests with respect to state-of-the-art analysis methods for the models covered in this paper, i.e., response-time analysis exact tests for tasks with and without blocking; the analyses in [17], [18], and [14] for tasks with jitters, offsets, and involved in transactions, respectively.

We tested systems with $n \in [5, 25]$ tasks and utilization U from 0.7 to 0.95 with step 0.05 (relevant performance differences have been observed only at high utilizations and for small task sets). For every combination of parameters (n, U), $\nu = 50000$ task sets have been tested, for a total of about 1 million sets. The utilizations of the tasks have been generated with the UUnifast [20] algorithm, enforcing a minimum per-task utilization of 1%. Periods have been randomly chosen (i) between 1 ms and 1 second, with uniform distribution; or (ii) in a limited set of pseudo-harmonic periods $\{1, 2, 5, 10, 15, 20, 25, 30, 45, 50, 75, 100\}$ ms, to test more realistic configurations. The execution time is then accordingly computed as $C_i = T_i \cdot U_i$ (rounded down to the first integer).

For the case of transactional task sets, we tested systems with $N_T \in [3, 6]$ transactions, where every transaction has $N_i \in [2, 5]$ tasks, thus obtaining a number of total tasks ranging from 6 to 30. For every combination of transactions and utilizations U, 5000 tests have been performed. Deadlines D_i have been randomly generated in the interval $[C_i + \alpha_D(T_i - C_I), T_i]$, with $0 \leq \alpha_D \leq 1$, offsets ϕ_i in $[0, \alpha_o T_i]$, with $0 \leq \alpha_o < 1$, and jitters J_i in $[0, \alpha_J(D_i - C_i)]$, with $0 \leq \alpha_J \leq 0.2$, all with uniform distribution. α_D, α_o, and α_J are tunable parameters. For the general case of Fixed Priority scheduling (i.e., not Rate Monotonic), task priorities are randomly assigned. As a comparison metric we selected the *failure rate*, computed as the normalized difference between the number of task sets deemed schedulable by our tests, denoted by A_p, and those accepted by necessary and sufficient state-of-the-art tests, denoted by A_e. The failure rate is $f = (A_e - A_p)/\nu$ and provides the fraction of task sets for which our tests fail in detecting a schedulable condition.

TABLE I
Maximum failure rates of the proposed tests.

	U	0.7	0.75	0.8	0.85	0.90	0.95
RM, ID		$< 1\%$	$< 1\%$	$< 1\%$	**1.3%**	**1.7%**	**1.2%**
RM, CD		$< 1\%$	$< 1\%$	$< 1\%$	$< 1\%$	$< 1\%$	$< 1\%$
RM, ID, O		$< 1\%$	$< 1\%$	$< 1\%$	**1.2%**	**1.7%**	**1.2%**
RM, ID, J		$< 1\%$	$< 1\%$	$< 1\%$	**1.3%**	**1.6%**	**1.2%**
RM, ID, B		$< 1\%$	$< 1\%$	$< 1\%$	$< 1\%$	$< 1\%$	$< 1\%$
FP, ID		$< 1\%$	$< 1\%$	$< 1\%$	$< 1\%$	$< 1\%$	$< 1\%$
RM, ID, Ps		$< 1\%$	$< 1\%$	$< 1\%$	$< 1\%$	$< 1\%$	$< 1\%$
RM, CD, Ps		$< 1\%$	$< 1\%$	$< 1\%$	$< 1\%$	$< 1\%$	$< 1\%$
Trans,RM,ID		$< 1\%$	$< 1\%$	$< 1\%$	$< 1\%$	$< 1\%$	$< 1\%$
Trans, RM, CD		$< 1\%$	$< 1\%$	$< 1\%$	$< 1\%$	$< 1\%$	$< 1\%$
Trans, RM, ID, J		$< 1\%$	$< 1\%$	$< 1\%$	$< 1\%$	$< 1\%$	$< 1\%$

Legend: RM: Rate Monotonic; FP: Fixed Priority; ID: Implicit deadline; CD: Constrained deadline; Trans: Transactional tasks; O: with offsets; J: with jitter; B: with blocking; Ps: Pseudo-harmonic sets

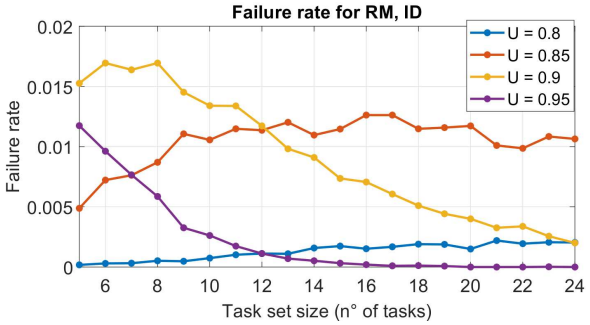

Fig. 1. Failure rate as a function of the number of tasks under RM, ID.

A. Experimental results

In the case of Rate Monotonic, implicit-deadline task sets, Fig. 1 shows the failure rates as a function of the number

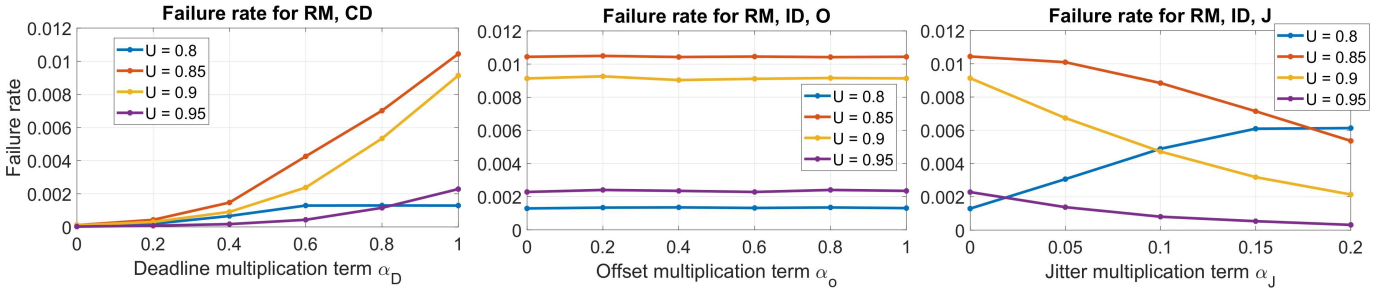

Fig. 2. Failure rate with variable deadlines (left), offsets (center) and jitter (right).

of tasks n. Quite interestingly, the maximum failure never exceeds 2%. Fig. 2 shows the failure rate for task sets with constrained deadlines, offsets, and jitters, as a function of parameters α_D, α_o and α_J, respectively. Each point in the plots reports the results aggregated for all the tested task set sizes, i.e., averaging among all the tested task sets with $n \in [5, 25]$. These parameters impact the failure rate of our tests in different ways: while changing the offsets does not affect the precision of the proposed test, tightening the deadline and increasing the jitter generally corresponds to a decrease of the failure rate. In all the tested configurations, the failure rate resulted around (or below) 1%. Surprisingly enough, low failure rates ($< 1\%$) are also found for task sets that include blocking, and in general all cases of Fixed Priority scheduling (not Rate Monotonic), combining different configurations of the parameters that control jitter, offsets, and constrained deadlines. Extremely low values are also found for all the task sets with pseudo-harmonic tasks.

In summary, the proposed tests revealed high schedulability performance both for hard-to-schedule task sets (with blocking, high jitter, tight deadlines and arbitrary priorities) and for easy-to-schedule sets (with pseudo-harmonic tasks), with failure rates that are quite below uncertainties that are typically associated with the determination of task parameters such as the worst-case execution time. Finally, for all transactional task sets, our experiments show extremely low failure rates for all the jitter and deadlines values in the chosen ranges. A selection of the results for the most relevant tests is reported in Table I.

VI. Conclusion

This paper presented approximate schedulability tests for Fixed Priority scheduling under a wide range of assumptions on the task model. The tests can be implemented with a limited number of linear closed-form equations, and are suitable for use in Mixed-Integer Linear Programs for the optimization of real-time systems. A large-scale experimentation revealed very low failure rates, showing a pessimistic outcome only in less than 1% of the proposed tests cases with respect to state-of-the-art schedulability tests (including exact ones).

References

[1] M. Lowinski, D. Ziegenbein, and S. Glesner, "Splitting tasks for migrating real-time automotive applications to multi-core ecus," in *Industrial Embedded Systems (SIES), 2016 11th IEEE Symposium on*. IEEE, 2016, pp. 1–8.

[2] A. Biondi, P. Pazzaglia, A. Balsini, and M. Di Natale, "Logical execution time implementation and memory optimization issues in autosar applications for multicores," in *International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS)*, 2017.

[3] M. Park and H. Park, "An efficient test method for rate monotonic schedulability," *IEEE Transactions on Computers*, vol. 63, no. 5, pp. 1309–1315, 2014.

[4] M. Joseph and P. Pandya, "Finding response times in a real-time system," *The Computer Journal*, vol. 29, no. 5, pp. 390–395, 1986.

[5] H. Zeng and M. Di Natale, "An efficient formulation of the real-time feasibility region for design optimization," *IEEE Transactions on Computers*, vol. 62, no. 4, pp. 644–661, 2013.

[6] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the ACM (JACM)*, vol. 20, no. 1, pp. 46–61, 1973.

[7] E. Bini, G. C. Buttazzo, and G. M. Buttazzo, "Rate monotonic analysis: the hyperbolic bound," *IEEE Transactions on Computers*, vol. 52, no. 7, pp. 933–942, 2003.

[8] R. I. Davis and A. Burns, "Response time upper bounds for fixed priority real-time systems," in *Real-Time Systems Symposium, 2008*. IEEE, 2008, pp. 407–418.

[9] R. I. Davis, A. Zabos, and A. Burns, "Efficient exact schedulability tests for fixed priority real-time systems," *IEEE Transactions on Computers*, vol. 57, no. 9, pp. 1261–1276, 2008.

[10] E. Bini, A. Parri, and G. Dossena, "A quadratic-time response time upper bound with a tightness property," in *Real-Time Systems Symposium, 2015 IEEE*. IEEE, 2015, pp. 13–22.

[11] J.-J. Chen, W.-H. Huang, and C. Liu, "k2q: A quadratic-form response time and schedulability analysis framework for utilization-based analysis," in *Real-Time Systems Symposium (RTSS), 2016 IEEE*. IEEE, 2016, pp. 351–362.

[12] J. Lehoczky, L. Sha, and Y. Ding, "The rate monotonic scheduling algorithm: Exact characterization and average case behavior," in *Real Time Systems Symposium, 1989., Proceedings.* IEEE, 1989, pp. 166–171.

[13] E. Bini and G. C. Buttazzo, "Schedulability analysis of periodic fixed priority systems," *IEEE Transactions on Computers*, vol. 53, no. 11, pp. 1462–1473, 2004.

[14] J. C. Palencia and M. G. Harbour, "Schedulability analysis for tasks with static and dynamic offsets," in *Real-Time Systems Symposium, 1998. Proceedings. The 19th IEEE*. IEEE, 1998, pp. 26–37.

[15] F. Eisenbrand and T. Rothvoß, "Static-priority real-time scheduling: Response time computation is np-hard," in *Real-Time Systems Symposium, 2008*. IEEE, 2008, pp. 397–406.

[16] P. Ekberg and W. Yi, "Fixed-priority schedulability of sporadic tasks on uniprocessors is np-hard," in *Real-Time Systems Symposium (RTSS), 2017 IEEE*. IEEE, 2017, pp. 139–146.

[17] N. Audsley, A. Burns, M. Richardson, K. Tindell, and A. J. Wellings, "Applying new scheduling theory to static priority pre-emptive scheduling," *Software Engineering Journal*, vol. 8, no. 5, pp. 284–292, 1993.

[18] R. Pellizzoni and G. Lipari, "Feasibility analysis of real-time periodic tasks with offsets," *Real-Time Systems*, vol. 30, no. 1-2, pp. 105–128, 2005.

[19] A. Biondi and M. D. Natale, "Achieving predictable multicore execution of automotive applications using the LET paradigm," in *In Proc. of the 24th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2018)*. IEEE, 2018.

[20] E. Bini and G. C. Buttazzo, "Measuring the performance of schedulability tests," *Real-Time Systems*, vol. 30, no. 1-2, pp. 129–154, 2005.