Proceedings of the 27th Euromicro Conference on Real-Time Systems (ECRTS 2015), Lund, Sweden, July 8-10, 2015.

Feasibility Analysis of Engine Control Tasks under EDF Scheduling

Alessandro Biondi, Giorgio Buttazzo, Stefano Simoncelli
Scuola Superiore Sant'Anna, Pisa, Italy
Email: alessandro.biondi@sssup.it, giorgio.buttazzo@sssup.it, simoncelli.stefano@hotmail.it

***Abstract*—Engine control applications include software tasks that are triggered at predetermined angular values of the crankshaft, thus generating a computational workload that varies with the engine speed. To avoid overloads at high rotation speeds, these tasks are implemented to self adapt and reduce their computational demand by switching mode at given rotation speeds. For this reason, they are referred to as adaptive variable rate (AVR) tasks. Although a few works have been proposed in the literature to model and analyze the schedulability of such a peculiar type of tasks, an exact analysis of engine control applications has been derived only for fixed priority systems, under a set of simplifying assumptions. The major problem of scheduling AVR tasks with fixed priorities, however, is that, due to engine accelerations, the interarrival period of an AVR task is subject to large variations, therefore there will be several speeds at which any fixed priority assignment is far from being optimal, significantly penalizing the schedulability of the system.**

This paper proposes for the first time an exact feasibility test under the Earliest Deadline First scheduling algorithm for tasks sets including regular periodic tasks and AVR tasks triggered by a common rotation source. In addition, a set of simulation results are reported to evaluate the schedulability gain achieved in this context by EDF over fixed priority scheduling.

I. Introduction

Engine control systems include two types of computational activities: *periodic tasks*, activated in a time-driven fashion at regular time intervals, and *angular tasks*, activated in an event-driven fashion at specific angular values of the crankshaft [1]. As a consequence, the activation rate of angular tasks is proportional to the engine rotation speed. To prevent overload conditions at high engine speeds, such angular tasks are often implemented to adapt their computational demand as a function of the rotation speed [2]. In particular, they are designed as a set of modes, each executed within a predefined speed range. For this reason, they will also be referred to as *adaptive variable-rate* (AVR) tasks.

The schedulability analysis of real-time task sets including both periodic and AVR tasks has recently been addressed by several authors, under a variety of models and simplifying assumptions. Kim, Lakshmanan, and Rajkumar [3] presented a first schedulability test for fixed priority (FP) systems, but their result only applies to a single AVR task running at the highest priority and having interarrival times smaller than the other periods. Relative deadlines are assumed to be equal to periods and priorities are assigned according to the Rate-Monotonic algorithm. Pollex et al. [4] derived a sufficient schedulability test for fixed priorities, under the assumption of a constant engine speed. Davis et al. [5] analyzed the dynamic behavior of AVR tasks under fixed priority assignments and proposed a sufficient schedulability test using an Integer Linear Programming (ILP) formulation. This approach, however, is based on speed quantization, which adds more pessimism in the analysis. A method for computing the exact interference of an AVR task under fixed priorities has been presented for the first time by Biondi et al. [6], who used a search based approach in the speed domain. In this approach, the concept of dominant speeds is used to contain the complexity and avoid speed quantization. Then, the exact response time analysis of AVR tasks and periodic tasks in the presence of AVR tasks has been presented by Biondi et al. [7].

For this type of applications, however, fixed priorities scheduling is not the best choice. In fact, given the range of values in which the engine speed can vary, typically from 500 to 6500 rotations per minute (rpm), the interarrival time of an AVR task varies from 9 to 120 ms (assuming a single activation per cycle). Considering that there exists other tasks with fixed period ranging from a few milliseconds up to 100 ms (see [1], page 152), there will be several engine speeds at which any fixed priority assignment is far from being optimal, significantly penalizing the system schedulability. In this context, scheduling the task set by Earliest Deadline First (EDF) [8] would allow achieving a much higher schedulability, independently on the period values.

The analysis of engine control applications under EDF scheduling has been addressed by Buttazzo, Bini, and Buttle [9], but under the assumption that AVR tasks are triggered by independent rotation sources. This assumption, however, introduces some pessimism, because it considers situations that can never occur when tasks are linked to the same rotation source. The EDF schedulability of AVR tasks linked to a common rotation source has been analyzed by Biondi and Buttazzo [10], but their test is only sufficient, since derived from a utilization upper bound. Recently, Guo and Baruah [11] studied the EDF scheduling of AVR tasks proposing a speedup factor analysis and sufficient schedulability tests.

Contributions. This work has three main contributions:

1) An exact schedulability analysis is presented under EDF scheduling for mixed task sets consisting of classic periodic tasks and AVR tasks triggered by a common rotation source. The analysis is based on demand bound functions derived as a search problem exploiting pruning rules.
2) A set of experimental results is reported to evaluate the schedulability gain achieved by the exact EDF test proposed in this paper over the exact FP test [7].
3) A set of simulation experiments are carried out through a scheduling simulator to evaluate the average performance gain of EDF over FP scheduling.

Paper structure: The rest of the paper is organized as follows. Section II presents the system model. Section III

1068-3070/15 $31.00 © 2015 IEEE
DOI 10.1109/ECRTS.2015.20

introduces our approach for analyzing the schedulability of a system composed of periodic and AVR tasks. Section IV describes how to compute the worst-case workload for an AVR task. Section V presents a set of experimental results aimed at comparing EDF and FP scheduling. Finally, Section VI summarizes the results and concludes the paper.

II. System model

For the purpose of the analysis, this paper considers a single rotation source (the engine) characterized by the following state variables:

- θ the current rotation angle of the crankshaft;
- ω the current angular speed of the crankshaft;
- α the current angular acceleration of the crankshaft.

We assume that the speed ω is limited within a given range $[\omega_{min}, \omega_{max}]$ and the acceleration α is limited within a given range $[\alpha^-, \alpha^+]$ and its rate of change (jerk) is bounded.

A. Task model

The application consists of a set of n real-time preemptive tasks $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\}$. Each task can be either *periodic* (i.e., activated at fixed time intervals), or an *angular task* (i.e., activated at specific crankshaft rotation angles). Since angular tasks have a variable interarrival time linked to the engine speed and adapt their workload for different speeds, they are also referred to as adaptive variable-rate (AVR) tasks. In the following, the subset of regular periodic tasks is denoted as Γ_P and the subset of angular AVR tasks is denoted as Γ_A, so that $\Gamma = \Gamma_P \cup \Gamma_A$ and $\Gamma_P \cap \Gamma_A = \emptyset$. The overall utilization of Γ_P is denoted as U_P. For the sake of clarity, whenever needed, an AVR task may also be denoted as τ_i^*.

Both types of tasks are characterized by a worst-case execution time (WCET) C_i, an interarrival time (or period) T_i, and a relative deadline D_i. However, while for regular periodic tasks such parameters are fixed, for angular tasks they depend on the engine rotation speed ω. In particular, an angular task τ_i^* is characterized by an *angular period* Θ_i and an *angular phase* Φ_i, so that it is activated at the following angles:

$$\theta_i = \Phi_i + k\Theta_i, \quad \text{for } k = 0, 1, 2, \ldots$$

This means that the period of an AVR task is inversely proportional to the engine speed ω and can be expressed as

$$T_i(\omega) = \frac{\Theta_i}{\omega}. \tag{1}$$

The angular phase Φ_i is relative to a reference position called *Top Dead Center* (TDC) corresponding to the crankshaft angle for which at least one piston is at the highest position in its cylinder. Without loss of generality, the TDC position is assumed to be at $\theta = 0$. An angular task τ_i^* is also characterized by a relative *angular deadline* Δ_i expressed as a fraction δ_i of the angular period ($\delta_i \in [0, 1]$). In the following, $\Delta_i = \delta_i \Theta_i$ represents the relative angular deadline.

An AVR task τ_i^* is typically implemented as a set $\mathcal{M}_i$ of M_i execution modes with decreasing functionality, each operating in a predetermined range of rotation speeds. Mode m of an AVR task τ_i^* is characterized by a WCET C_i^m and is valid in a speed range $(\omega_i^{m+1}, \omega_i^m]$, where $\omega_i^{M_i+1} = \omega^{min}$ and $\omega_i^1 = \omega^{max}$. Hence, the set of modes of task τ_i^* can be expressed as

$$\mathcal{M}_i = \{(C_i^m, \omega_i^m), m = 1, 2, \ldots, M_i\}.$$

The computation time of a generic AVR job $J_{i,k}$ can be expressed as a non-increasing step function $\mathcal{C}_i$ of the instantaneous speed ω at its release, that is,

$$C_{i,k} = \mathcal{C}(\omega) \in \{C_i^1, \ldots, C_i^{M_i}\}. \tag{2}$$

An example of $\mathcal{C}$ function is illustrated in Figure 1.

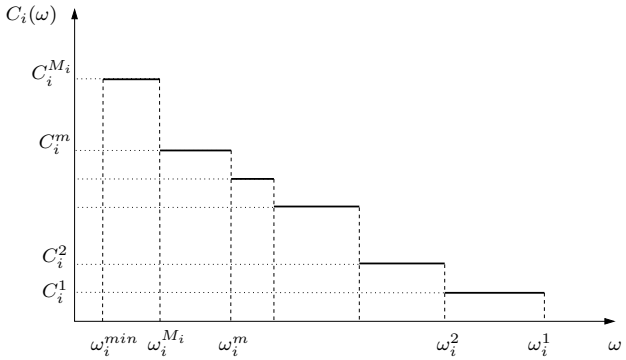

Figure 1. Computation time of an AVR task as a function of the speed at the job activation.

The implementation of such a type of tasks is typically performed as a sequence of conditional `if` statements, each executing a specific subset of functions [2], [9]. Figure 2 illustrates a sample AVR task with four modes reported in Table I. In this example, $\omega_{min} = 500$ rpm, and $\omega_{max} = 8000$ rpm. Note that we assume that function `read_rotation_speed()` returns the instantaneous speed ω at the task activation time (not at the execution time of the function).

Mode	rotation (rmp)	functions to be executed
Mode 1	(6000, 8000]	`f1();`
Mode 2	(4000, 6000]	`f1(); f2();`
Mode 3	(2000, 4000]	`f1(); f2(); f3();`
Mode 4	[500, 2000]	`f1(); f2(); f3(); f4();`

Table I. SAMPLE AVR TASK WITH FOUR MODES.

```
#define W1 8000
#define W2 6000
#define W3 4000
#define W4 2000
task  sample_AVR_task {
      omega = read_rotation_speed();
      if (omega ≤ W1) f1();
      if (omega ≤ W2) f2();
      if (omega ≤ W3) f3();
      if (omega ≤ W4) f4();
}
```

Figure 2. Implementation of the AVR task reported in Table I.

Note that, when using EDF as a scheduler, an absolute deadline must be assigned to each job at its activation time in order to be scheduled. Although the angular deadline is constant, the temporal deadline is a function of ω and (for constant rotation speed) is equal to

$$D_i(\omega) = \frac{\Delta_i}{\omega} = \frac{\delta_i \Theta_i}{\omega} = \delta_i T_i(\omega). \tag{3}$$

However, for an incoming job, the next arrival time is not known, since ω may not be constant over time. To achieve

a safe schedule, we make the conservative assumption to always assign each job the earliest possible deadline among those compatible with the speed at its activation, that is, the one derived assuming the maximum acceleration α^+. A more precise deadline assignment could be achieved by considering the engine dynamics, but this is left to future work due to lack of space. In general, it is worth observing that the optimal deadline assignment for a job of an angular task requires clairvoyance to determine the exact interarrival period (which depends on the future engine evolution).

In the following, the schedulability analysis is presented for a single AVR task. Note however that, as shown by Biondi et al. [7], multiple AVR tasks with the same phase and angular period triggered by a common rotation source are equivalent to a singe AVR task having as modes the union of the various modes. Thanks to this equivalence, all the results derived in this paper apply to a set AVR tasks with zero phase and the same angular period.

B. Rotation source model

For the purpose of analyzing the schedulability of engine control applications consisting of AVR tasks, it is crucial to characterize the relation between the task parameters and the dynamics of the engine. Following the approach proposed by Buttazzo et al. [9], if (ω_k, α_k) is the state of the engine at time t_k, the next job release time of an AVR task can be computed (assuming a constant acceleration α_k) as:

$$T_i(\omega_k, \alpha_k) = \frac{\sqrt{\omega_k^2 + 2\Theta_i\alpha_k} - \omega_k}{\alpha_k}. \tag{4}$$

The corresponding relative deadline in the time domain can be computed by considering the earliest value given by the maximum acceleration α^+, that is

$$D_i(\omega_k) = \frac{\sqrt{\omega_k^2 + 2\Delta_i\alpha^+} - \omega_k}{\alpha^+}. \tag{5}$$

In a similar way, the instantaneous rotation speed Ω at the next job release can be computed (assuming constant acceleration during the angular period) as $\Omega(\omega_k, \alpha_k) = \omega_k + \alpha_k T_i(\omega_k, \alpha_k)$, which gives:

$$\Omega_i(\omega_k, \alpha_k) = \sqrt{\omega_k^2 + 2\Theta_i\alpha_k}. \tag{6}$$

It also convenient to define the inverse function of Equation (6), representing the speed ω_k needed to reach ω_{k+1} with acceleration α_k, that is,

$$\Omega_i^-(\omega_{k+1}, \alpha_k) = \sqrt{\omega_{k+1}^2 - 2\Theta_i\alpha_k}. \tag{7}$$

In the analysis, we also need to compute the engine speed after n job releases (following the k^{th} job), with constant acceleration α; such a value is denoted as Ω^n and can be computed as $\Omega^n(\omega_k, \alpha) = \Omega(\Omega^{n-1}(\omega_k, \alpha), \alpha)$, where $\Omega^0(\omega_k, \alpha) = \omega_k$. Similarly, we define $\Omega^{-n}(\omega_{k+1}, \alpha) = \Omega^-(\Omega^{-(n-1)}(\omega_{k+1}, \alpha), \alpha)$, where $\Omega^{-0}(\omega_{k+1}, \alpha) = \omega_{k+1}$.

If a job $J_{i,k}$ is released when the engine has an instantaneous speed ω_k, the interarrival time $\widetilde{T}_i(\omega_k, \omega_{k+1})$ to the next job $J_{i,k+1}$ released with instantaneous speed ω_{k+1} (if reachable with the acceleration bounds), can be obtained by Equation (4), substituting α_k from Equation (6), which gives:

$$\widetilde{T}_i(\omega_k, \omega_{k+1}) = \frac{2\Theta_i}{\omega_k + \omega_{k+1}}. \tag{8}$$

III. Worst-case workload as a search problem

The schedulability analysis of a generic real-time workload under EDF scheduling can be performed by the processor demand criterion [12], according to which a task set is schedulable by EDF if and only if its computational demand in any time interval $[t_1, t_2]$ does not exceed the available processor supply. The computational demand of a real-time task in an interval of time $[t_1, t_2]$ can be quantified by the *demand bound function* (dbf), which expresses the maximum execution requirement of a task having both release time and deadline in $[t_1, t_2]$. The cumulative workload on the processor can then be obtained by summing the dbf of each task in the system. When no initial phases are provided for tasks (i.e., when it is possible to have simultaneous releases for all tasks), an exact schedulability test can be achieved by considering all tasks released simultaneously at time $t = 0$ and then apply the processor demand criterion for all intervals $[0, t]$, $\forall t > 0$. The release pattern at time $t = 0$ is denoted as the *critical instant*.

In the following, the concept of critical instant is generalized for task sets including an AVR task, and the workload of an AVR task is computed in a given time window $[0, t]$.

Since the rotation source triggering the release of AVR tasks is independent of the mechanism activating the other periodic (sporadic) tasks, it is always possible to have an instant of time at which the release of the AVR task occurs simultaneously with the activation of all the other tasks. Unfortunately, in the presence of an AVR task, there can be potentially infinite critical instants, one for each possible instantaneous speed ω_0 at which the AVR task is triggered when all the other tasks are simultaneously released at $t = 0$. This occurs because, depending on the speed ω_0, the AVR task generates a different worst-case workload related to all possible speed evolution patterns starting from ω_0.

To characterize the workload of an AVR task we introduce the following definitions.

Definition 1 (AVR job sequence): An AVR job sequence s is an ordered list of jobs $J_0^{(s)}, \ldots, J_{n_s}^{(s)}$ and a corresponding list of speeds $\omega_0^{(s)}, \ldots, \omega_{n_s}^{(s)}$, where each speed $\omega_i^{(s)}$ represents the instantaneous engine speed at the activation of job $J_i^{(s)}$.

For each AVR job sequence s, the sequence of speeds $\omega_0^{(s)}, \ldots, \omega_{n_s}^{(s)}$ must be compatible with the acceleration range allowed for the rotation source.

Let $\mathcal{S}(t)$ be the set of all possible job sequences in $[0, t]$, which are those having the release of the first job at time $t = 0$ and the deadline of the last job no later than time t. Please note that, being the speed domain a continuum, the set $\mathcal{S}(t)$ is generally infinite.

Definition 2 (Single-Job Demand): Given a job J of an AVR task, released at relative time instant $\lambda = 0$ with instantaneous speed ω, the demand of J (its workload contribution to the system) in a generic interval $[0, \lambda]$ is

$$\mathsf{sjd}_\omega(\lambda) = \mathcal{C}(\omega)\ \mathsf{step}\left(\lambda - D(\omega)\right), \tag{9}$$

where

$$\mathsf{step}(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}.$$

Note that the single-job demand is the smallest unit contributing to the workload of a job sequence s, which can be expressed as the sum of the time-shifted computational demands of the jobs in the sequence. We now introduce the concept of *demand bound function* $\mathsf{dbf}^{(s)}(t)$ of an AVR job sequence $s \in \mathcal{S}(t)$:

$$\mathsf{dbf}^{(s)}(t) = \sum_{k=0}^{n_s} \mathcal{C}(\omega_k^{(s)}) \; \mathsf{step}\left(t - \sum_{j=1}^{k} \widetilde{T}(\omega_{j-1}^{(s)}, \omega_j^{(s)}) - D(\omega_k^{(s)})\right) \tag{10}$$

An example of $\mathsf{dbf}^{(s)}(t)$ function is reported in Figure 3 for a simple job sequence s composed of three jobs, released at speeds ω_0, ω_1 and ω_2, respectively. Dashed arrows indicate the instantaneous speed at the time instant corresponding to the job release.

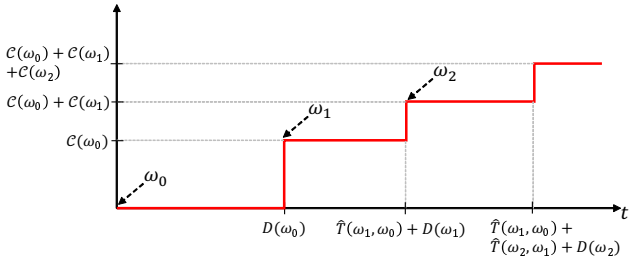

Figure 3. Example of $\mathsf{dbf}^{(s)}(t)$ function for an AVR job sequence composed of three jobs.

Being all job sequences possible scenarios for the system, all of them must be taken into account to analyze the feasibility of the system. For this purpose, the *demand bound function* $\mathsf{dbf}^*(t)$ of an AVR task is defined as the maximum demand of all possible job sequences, that is:

$$\mathsf{dbf}^*(t) = \max_{s \in \mathcal{S}(t)} \left\{ \mathsf{dbf}^{(s)}(t) \right\}. \tag{11}$$

Intuitively, function $\mathsf{dbf}^*(t)$ represents the *envelope* of the demand of all possible job sequences allowed for the AVR task. Hence, the feasibility test for a system including an AVR task can be expressed as follows:

$$\forall t > 0, \; \sum_{\tau_i \in \Gamma^P} \mathsf{dbf}_i(t) + \mathsf{dbf}^*(t) \leq t \tag{12}$$

where $\mathsf{dbf}_i(t)$ is the demand bound function of a classical periodic (sporadic) task, defined as

$$\mathsf{dbf}_i(t) = \left(\left\lfloor \frac{t - D_i}{T_i} \right\rfloor + 1 \right) C_i. \tag{13}$$

Unfortunately, computing $\mathsf{dbf}^*(t)$ through Equation (11) would require a maximum among an infinite number of job sequences. The characterization of all possible job sequences can be seen as a search problem in the speed domain; that is, given an initial speed ω_0, we can collect all possible speeds ω_1 following ω_0 that are compatible with the acceleration range allowed for the rotation source. Then, the same reasoning can be applied to speed ω_1, leading to the definition of a *search tree*, schematically illustrated in Figure 4.

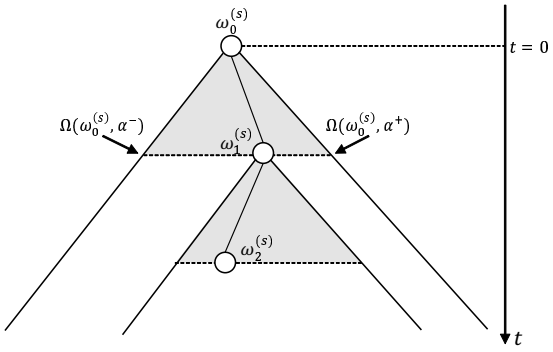

Figure 4. Search tree representing all possible job sequences.

Such a search problem can be solved by brute-force applying quantization on the speed domain. However, besides making the problem intractable for most practical cases, this approach will always provide an approximated solution, resulting in an unsafe analysis.

The next section presents a technique to compute $\mathsf{dbf}^*(t)$ avoiding quantization on the speed domain, thus significantly reducing the complexity of the search problem by enforcing dominance conditions.

IV. Worst-Case Workload of an AVR Task

The goal of this section is to derive dominance conditions among job sequences, thus identifying *pruning rules* for the search problem introduced in the previous section. Then, the algorithms for computing function $\mathsf{dbf}^*(t)$ will be presented. The following theorem gives a dominance condition for a single-job demand.

Theorem 1 (Single-job dominance condition): Let J_a and J_b be two jobs of an AVR task released at relative time instant $\lambda = 0$ with speed ω_a and ω_b, respectively. If $\omega_a \geq \omega_b$ and $\mathcal{C}(\omega_a) = \mathcal{C}(\omega_b)$, then $\forall \lambda > 0$, $\mathsf{sjd}_{\omega_a}(\lambda) \geq \mathsf{sjd}_{\omega_b}(\lambda)$.

Proof: The deadline of an AVR task is assigned as $D(\omega) = \delta T(\omega, \alpha^+)$. Being $T(\omega, \alpha^+)$ a non-increasing function of ω, we have that $D(\omega_a) \leq D(\omega_b)$. Hence, we can conclude that $\forall \lambda \in [0, D(\omega_b))$, $\mathsf{sjd}_{\omega_a}(\lambda) \geq \mathsf{sjd}_{\omega_b}(\lambda)$, being function $\mathsf{sjd}_{\omega_b}(\lambda) = 0$ for $\lambda < D(\omega_b)$. For $\lambda \geq D(\omega_b)$, $\mathsf{sjd}_{\omega_b}(\lambda) = \mathcal{C}(\omega_b)$, while $\mathsf{sjd}_{\omega_a}(\lambda)$ is holding value $\mathcal{C}(\omega_a)$. Then, under the hypothesis $\mathcal{C}(\omega_a) = \mathcal{C}(\omega_b)$, we have that $\forall \lambda \geq D(\omega_b)$, $\mathsf{sjd}_{\omega_a}(\lambda) = \mathsf{sjd}_{\omega_b}(\lambda)$. Hence the theorem follows. ■

To better clarify the result expressed by Theorem 1, Figure 5-a illustrates an example in which the single-job dominance condition is satisfied, while Figure 5-b illustrates an example in which it is not.

Unfortunately, a dominance condition on a single-job demand is not enough to identify a pruning rule for the addressed search problem. In fact, it could be that, besides having a dominance of J_a on J_b, a set of jobs following J_b are not dominated by any job following J_a. In the search tree, this

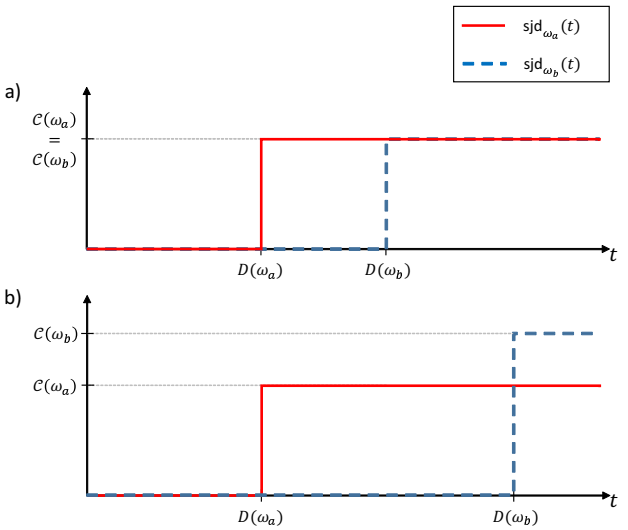

Figure 5. a) Example at which Theorem 1 applies, b) Example at which Theorem 1 does not apply.

means that the sub-tree of job sequences following J_b is not dominated by the one following J_a. To address this issue in a more formal way, we identify a set of *critical job sequences* that allows computing $\mathsf{dbf}^*(t)$ without involving the (infinite) set $\mathcal{S}(t)$.

Definition 3 (Set of Critical Job-Sequences): The set $\mathcal{CS}(t) \subseteq \mathcal{S}(t)$ of critical job sequences in $[0,t]$ is a set composed of job sequences such that, for each non-critical job sequence $s' \notin \mathcal{CS}(t)$ there exists a critical job sequence $s \in \mathcal{CS}(t)$ whose demand bound function dominates the one of s'. That is,

$$\forall s' \notin \mathcal{CS}(t)\ \exists s \in \mathcal{CS}(t) \mid \forall \ell \in [0,t]\ \mathsf{dbf}^{(s)}(\ell) \geq \mathsf{dbf}^{(s')}(\ell).$$

Once $\mathcal{CS}(t)$ is identified, the *demand bound function* $\mathsf{dbf}^*(t)$ can be computed as $\mathsf{dbf}^*(t) = \max_{s \in \mathcal{CS}(t)}\{\mathsf{dbf}^{(s)}(t)\}$. In the following we present a theorem expressing a dominance condition that allows computing the set $\mathcal{CS}(t)$.

To simplify the presentation of the following results, we introduce the *demand bound function* for a speed ω_0, that is the envelope of the demand bound functions for all possible job sequences starting at speed ω_0 at relative time instant $\lambda = 0$. Formally,

$$\mathsf{dbf}_{\omega_0}(\lambda) = \max_{s \in \mathcal{S}(\lambda)} \left\{ \mathsf{dbf}^{(s)}(\lambda) \mid \omega_0^{(s)} = \omega_0 \right\}. \quad (14)$$

Function $\mathsf{dbf}_{\omega_0}(\lambda)$ can be also recursively expressed by considering all possible speeds following ω_0, that are the ones in the range $A(\omega_0) = [\Omega(\omega_0, \alpha^-), \Omega(\omega_0, \alpha^+)]$. This can be achieved by splitting the outer sum in Equation (10) in two terms, one considering the first job in the sequence ($k = 0$), and another considering the remaining jobs ($1 \leq k \leq n_s$). Assuming a sequence started at speed ω_0, the first term can be replaced by $\mathsf{sjd}_{\omega_0}(\lambda)$, while the second can be treated as a family of sub-sequences starting at speeds following ω_0 defined by the set $A(\omega_0)$. This reasoning leads to

$$\mathsf{dbf}_{\omega_0}(\lambda) = \mathsf{sjd}_{\omega_0}(\lambda) + \max_{\omega' \in A(\omega_0)} \left\{ \mathsf{dbf}_{\omega'}(\lambda - \widetilde{T}(\omega_0, \omega')) \right\}. \quad (15)$$

Theorem 2 (Dominance Condition): Let J_a and J_b be two jobs of an AVR task released at the relative time instant $\lambda = 0$ with speed ω_a and ω_b, respectively. If $\omega_a \geq \omega_b$ and $\forall n \geq 0$, $n \in \mathbb{Z}$, such that $\mathcal{C}(\Omega^n(\omega_a, \alpha^-)) = \mathcal{C}(\Omega^n(\omega_b, \alpha^-))$, then $\forall \lambda > 0$, $\mathsf{dbf}_{\omega_a}(\lambda) \geq \mathsf{dbf}_{\omega_b}(\lambda)$.

Proof: First notice that, being $\mathcal{C}(\omega_a) = \mathcal{C}(\omega_b)$ ($n = 0$) and $\omega_a \geq \omega_b$, by Theorem 1 we can conclude that $\mathsf{sjd}_{\omega_a}(\lambda) \geq \mathsf{sjd}_{\omega_b}(\lambda)$. Hence, the dominance of the first term in Equation (15) is satisfied. It remains to show that the dominance condition is satisfied for each speed $\omega' \in A(\omega_b) = [\Omega(\omega_b, \alpha^-), \Omega(\omega_b, \alpha^+)]$. We split the proof in two parts.

Part 1. Consider speeds $\omega' \in [\Omega(\omega_a, \alpha^-), \Omega(\omega_b, \alpha^+)]$, reachable by both ω_a and ω_b. Exploring the search tree from ω_a, each speed ω' originates a workload contribution described by function $\mathsf{dbf}_{\omega'}(\lambda - \widetilde{T}(\omega_a, \omega'))$. Similarly, when each speed ω' is reached from ω_b, we have a workload contribution equal to $\mathsf{dbf}_{\omega'}(\lambda - \widetilde{T}(\omega_b, \omega'))$. Being $\omega_a \geq \omega_b$ we have $\widetilde{T}(\omega_b, \omega') \geq \widetilde{T}(\omega_a, \omega')$, hence we can conclude that $\mathsf{dbf}_{\omega'}(\lambda - \widetilde{T}(\omega_a, \omega')) \geq \mathsf{dbf}_{\omega'}(\lambda - \widetilde{T}(\omega_b, \omega')$, satisfying the dominance.

Part 2. Now consider the remaining speeds $\omega' \in [\Omega(\omega_b, \alpha^-), \Omega(\omega_a, \alpha^-))$, that are non reachable from ω_a. We start by looking at the first job following J_a and J_b, i.e., $n = 1$. Assume to look at the instance of the J_a's follower released at speed $\Omega(\omega_a, \alpha^-)$: in this case, since we have the hypothesis $\mathcal{C}(\Omega(\omega_a, \alpha^-)) = \mathcal{C}(\Omega(\omega_b, \alpha^-))$, Theorem 1 can be applied for each possible instance of the J_b's follower released at speeds $\omega' \in [\Omega(\omega_b, \alpha^-), \Omega(\omega_a, \alpha^-))$. Again, the dominance of the first term in Equation (15) is satisfied, but it remains to show the dominance for the following jobs ($n > 1$), considering the range $A(\omega'), \forall \omega' \in [\Omega(\omega_b, \alpha^-), \Omega(\omega_a, \alpha^-))$. Such a range can be split in two sub-ranges: $[\Omega^2(\omega_b, \alpha^-), \Omega^2(\omega_a, \alpha^-)]$ and $(\Omega^2(\omega_a, \alpha^-), \Omega(\Omega(\omega_a, \alpha^-), \alpha^+)]$. In the latter range there are speeds reachable from both J_a and J_b and we can apply the same considerations expressed in the first part of the proof. For the former range, the same argument of *Part 2* can be repeated for each $n > 1, n \in \mathbb{Z}$, thus obtaining the dominance condition. Hence, the theorem follows. ■

Theorem 2 allows identifying a limited set of speeds in the range $[\omega^{min}, \omega^{max}]$ for which the dominance condition holds. Such speeds are denoted as *dominant speeds*. As a consequence, set $\mathcal{CS}(t)$ can be computed by identifying the critical job sequences such that each job in the sequence is released at a dominant speed. Then, the approach proposed for computing the $\mathsf{dbf}^*(t)$ function of an AVR task consists in visiting a search-tree (in the speed domain) with pruning. Such a pruning is performed by exploiting the dominance condition stated in Theorem 2, which allows exploring the speed domain considering only the critical job sequences.

Conceptually, given a job J_0 released at speed ω_0, the next job J_1 can be activated in a range of infinite possible times, released at speed $\omega_1 \in [\Omega(\omega_0, \alpha^-), \Omega(\omega_0, \alpha^+)]$. Each possible activation of J_1 gives rise to a different job sequence started from speed ω_0. As expressed by Equation (15), the worst-case workload produced by all possible job sequences following J_0 results as the maximum of an infinite number of functions $\mathsf{dbf}_{\omega_1}(t)$.

However, by applying Theorem 2 in the range $[\Omega(\omega_0, \alpha^-), \Omega(\omega_0, \alpha^+)]$, it is possible to compute such a maximum restricting to the dominant speeds in that range, thus also limiting the possible activations of the subsequent job J_2 following J_1. This reasoning can be recursively applied for J_2 and all possible followers, so reducing the search domain to a discrete (finite) domain.

The next section provides the algorithms for computing the dominant speeds and the $\mathsf{dbf}^*(t)$ function.

A. Algorithms

Dominant speeds in a given range $[\omega_b, \omega_a]$ can be found by leveraging the result of Theorem 2 as follows: let ω^* be highest speed less than ω_a for which the condition of Theorem 2 does not hold. This means that all the speeds $\omega \in (\omega^*, \omega_a]$ are dominated by ω^a.

The minimum speed ω^* can be immediately found by inverting the conditions of Theorem 2. That is, for each value of n (number of look ahead steps in the search tree), we compute the speed $\omega_a^n = \Omega^n(\omega_a, \alpha^-)$, and then the maximum speed $\omega^{*,n} < \omega_a^n$ such that $\mathcal{C}(\omega_a^n) \neq \mathcal{C}(\omega^{*,n})$. Since function $\mathcal{C}(\omega)$ is non-increasing, such a speed $\omega^{*,n}$ can always be found[1]. Also note that speed $\omega^{*,n}$ has to be a switching speed for the AVR task, being the first speed for an adjacent mode in deceleration.

Given a value for n, it is possible to compute a *candidate* for speed ω^*, denoted as ω_n^*. Speed ω_n^* can then be easily computed by using the inverted physical equation of $\Omega^n(\omega, \alpha^-)$, that is $\omega_n^* = \Omega^{-n}(\omega^{*,n}, \alpha^-)$. Finally, since conditions of Theorem 2 has to be true $\forall n$, speed ω^* is computed as the maximum of all the candidates, that is, $\omega^* = \max_n\{\omega_n^*\}$. Such a speed is then stored as a *dominant speed*. The same reasoning is applied starting from speed ω^*, until reaching the minimum speed ω_b of the considered interval. Being the speeds domain limited in the range $[\omega^{min}, \omega^{max}]$, the maximum value for n is bounded to $\max\{n \in \mathbb{Z} \mid \Omega^n(\omega_a, \alpha^-) \geq \omega^{min}\}$.

The technique for computing dominant speeds is summarized in the algorithm reported in Figure 6.

```
1:  procedure GETDOMINANTS(ω_b, ω_a)
2:      ω* ← ω_a;
3:      while ω* > ω_b do
4:          DOMINANTS.ADD(ω*);
5:          maxN ← max{n ∈ ℤ | Ω^n(ω*, α^-) ≥ ω^min};
6:          for i = 0 to maxN do
7:              ω^i = Ω^i(ω*, α^-);
8:              ω^{*,i} = max_{m=1,...,M}{ω^m < ω^i};
9:              ω_i^* = Ω^{-i}(ω^{*,i}, α^-);
10:         end for
11:         ω* = max_{i=0,...,maxN}{ω_i^*};
12:     end while
13:     return DOMINANTS;
14: end procedure
```

Figure 6. Algorithm for computing the dominant speeds in a generic speed range $[\omega_b, \omega_a]$.

[1]The only exception is related to speeds $\omega_a^n < \omega^1$, i.e., lower than the first switching speed of the AVR task. In this case, the dominance is automatically satisfied since it is not possible to violate the hypothesis $\mathcal{C}(\omega_a^n) = \mathcal{C}(\omega^{*,n})$ of Theorem 2. In other words, it is not possible to have a mode change decelerating from speed ω_a^n.

By using Algorithm GETDOMINANTS it is possible to explore the search space through a finite number of (dominant) speeds, still achieving a correct characterization of the worst-case workload. To explore the search space, we start by computing the set of *dominant initial speeds* at the critical instant by executing GETDOMINANTS($\omega^{min}, \omega^{max}$). Then, for each dominant initial speed ω_0, we explore the job sequences started at ω_0 limiting the search to the ones having all jobs released at a dominant speed. This reasoning leads to the definition of a recursive procedure. Note that, given a generic speed range $[\omega_b, \omega_a]$, the derivation of dominant speeds results from: (i) a computation that is a function of a switching speed of the AVR task (lines 8-9 in Figure 6); and (ii) the maximum speed ω_a in the considered interval (line 2 in Figure 6). Due to point (i), the same dominant speed will be present in several speed ranges considered in the search problem. This enables taking advantage of *dynamic programming* by storing functions $\mathsf{dbf}_{\omega^*}(t)$ for each dominant speed ω^*.

```
1:  procedure DEMAND(ω, t)
2:
3:      if t > MAXTIME then
4:          return NULLDEMAND;
5:      end if
6:
7:      dbf_ω = sjd_ω;
8:
9:      dominants ← GETDOMINANTS(Ω(ω, α^-), Ω(ω, α^+));
10:
11:     for ω^next in dominants do
12:         if NOTSTORED(ω^next, t) then
13:             T^next ← T̃(ω, ω^next);
14:             dbf_{ω^next} ← DEMAND(ω^next, t + T^next);
15:             STOREDEMAND(ω^next, dbf_{ω^next}, t);
16:         else
17:             dbf_{ω^next} ← GETSTOREDDEMAND(ω^next, t);
18:         end if
19:         UPDATEDEMAND(dbf_ω, dbf_{ω^next}, T^next);
20:     end for
21:     return dbf_ω;
22: end procedure
```

Figure 7. Procedure for computing function $\mathsf{dbf}_\omega(t)$ performing a visit of the search tree considering only critical job sequences.

Figure 7 reports the recursive algorithm we used to explore the search-tree starting from a generic speed ω.

Each time DEMAND is called, t is the absolute release time of a job (released at speed ω) in a critical job sequence. In the pseudocode reported for such an algorithm, dbf_ω refers to a data structure able to represent and store a demand bound function. In addition, we assume the existence of a dynamic programming table used to store the demand bound functions dbf_{ω^*} for dominant speeds ω^*; such a table is accessed by procedure STOREDEMAND to store a new demand bound function at time t and by GETSTOREDDEMAND to retrieve a previously stored demand bound function. Function NOTSTORED allows checking whether the demand bound function has been stored for a given speed and can be used at time t.

At the beginning of the algorithm, dbf_ω is initialized with the single-job demand for a job released at speed ω (line 7). Then, dominant speeds are computed in the range allowed for speeds following ω (line 9). For each dominant speed ω^{next}, if

the demand for such a speed has not been already computed, the algorithm prepares for the next recursive call (lines 12-5), and then the demand for speed ω^{next} is stored to exploit dynamic programming. Otherwise, if the demand for the dominant speed has been already computed, then it is simply retrieved from the dynamic programming table. Finally, the resulting demand bound function dbf_ω is updated including the contribution of function $\mathsf{dbf}_{\omega^{\text{next}}}$ shifted by T^{next} time units (line 18). This is done by invoking procedure UPDATEDEMAND, which computes $\max\{\mathsf{dbf}_\omega(t), \mathsf{dbf}_{\omega^{\text{next}}}(t - T^{\text{next}})\}$.

```
1: procedure COMPUTEDBF*()
2:     dbf* = NULLDEMAND;
3:     initDominants ← GETDOMINANTS(ω^min, ω^max);
4:     for ω_0 in initDominants do
5:         dbf_ω0 ← DEMAND(ω_0, 0);
6:         UPDATEDEMAND(dbf*, dbf_ω0, 0);
7:     end for
8:     return dbf*;
9: end procedure
```

Figure 8. Main procedure for computing function dbf^*.

In conclusion, Algorithm DEMAND allows us to compute the overall demand bound function $\mathsf{dbf}^*(t)$ for an AVR task. Figure 8 reports the procedure to compute $\mathsf{dbf}^*(t)$. Such a procedure computes all the initial dominant speeds at the critical instant ($t = 0$) and then calls procedure DEMAND for each of such speeds. Similar considerations made for algorithm DEMAND applies to explain the behavior of Algorithm COMPUTEDBF*. Also in this case, we assume the existence of a data structure dbf^* able to store a demand bound function.

B. Upper-bound for the processor demand criterion

Equation (12) expresses an exact schedulability test for EDF scheduling, verifying that the processor is not overloaded in every time window $[0, t]$ following the critical instant. However, a bound L^* on the maximum time window length is needed to practically implement such a schedulability test.

In the context of EDF scheduling of periodic (sporadic) tasks, a linear upper-bound of the demand bound function has been exploited to compute L^*. Following the results reported in [12], the overall demand periodic (sporadic) task can be upper-bounded by:

$$\sum_{\tau_i \in \Gamma^P} \mathsf{dbf}_i(t) \leq \sum_{\tau_i \in \Gamma^P} (T_i - D_i)U_i + U_P t.$$

Unfortunately, the demand bound function $\mathsf{dbf}^*(t)$ of an AVR task has not a closed form expression, thus it is not easy to compute a linear upper-bound for it. To address this issue, we converted an AVR task into a *Digraph Real-Time* (DRT) task [13] and then applied the existing techniques for such task model to compute the upper-bound for $\mathsf{dbf}^*(t)$. A DRT task τ is characterized by a directed graph $\mathcal{G}(\tau) = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of vertices of the digraph and $\mathcal{E}$ is the set of edges connecting the vertices. Each vertex $v_i \in \mathcal{G}(\tau)$ represents a possible type of job for τ and is characterized by an ordered pair $< C(v_i), D(v_i) >$ representing the WCET and deadline of the vertex, respectively. Each directed edge $e_{i,j} \in \mathcal{E}$ connecting vertices v_i to v_j represents a possible order of job releases and it is labeled with the minimum inter-arrival time for a job of v_j after a job of v_i.

The underlying idea is that, following the results presented in the previous section, the worst-case behavior of an AVR task can be expressed by considering only the dominant speeds. Given an AVR task τ^*, the corresponding DRT task $D\tau^*$ is constructed as follows:

1) All the dominant speeds in the range $[\omega^{min}, \omega^{max}]$ are computed by using the algorithm in Figure 6;
2) For each dominant speed ω^* we construct a vertex in $D\tau^*$ having computation time $\mathcal{C}(\omega^*)$ and deadline $D(\omega^*)$;
3) For each vertex in $D\tau^*$ associated to dominant speed ω^* we compute all dominant speeds ω' in the range $[\Omega(\omega^*, \alpha^-), \Omega(\omega^*, \alpha^+)]$ (i.e., the range of speeds reachable with the acceleration allowed for the rotation source). Then we provide an edge connecting the vertex associated to each dominant speed ω' reachable from ω^* having inter-arrival time $T(\omega^*, \omega')$.

Please note that the resulting digraph expressing $D\tau^*$ will be a strongly connected graph, where each vertex can be reached from another (arbitrary) vertex. Moreover, each vertex includes a self-loop, because the case of zero acceleration is always possible for the rotation source.

Zeng and Di Natale [14] proposed a method for computing a tight upper-bound for the demand bound function of a DRT task: let X and U^∞ the parameters describing such a linear upper-bound, such that:

$$\mathsf{dbf}^*(t) \leq X + U^\infty t.$$

Finally, the maximum time window length L^* in which Equation (12) has to be checked can be computed by summing the linear upper-bounds of both periodic tasks and the AVR task and equating them to t, so obtaining

$$L^* = \frac{X + \sum_{\tau_i \in \Gamma^P} (T_i - D_i)U_i}{1 - (U_P + U^\infty)}.$$

Considering that an AVR task can be expressed as a DRT task, we can exploit the result derived by Stigge et al. [13] to claim that the feasibility of a task set consisting of periodic/sporadic tasks and an AVR task can be decided in pseudo-polynomial time, if the total utilization $U = U_P + U^\infty$ is not greater than c, for some constant $c < 1$.

V. EXPERIMENTAL RESULTS

This section presents two sets of experiments aimed at comparing FP and EDF scheduling for a system consisting of periodic (sporadic) tasks and an AVR task. The first set of experiments compares the two algorithms in terms of schedulability analysis, using for FP the exact test presented in [7] and for EDF the test proposed in this paper. The second set of experiments aims at comparing the schedulability of FP and EDF in the average case, making use of the RTSIM scheduling simulator [15], which has been extended to deal with AVR tasks.

We assume a rotation source ranging from $\omega^{min} = 500$ RPM to $\omega^{max} = 6500$ RPM as typical values for a production car engine. Similarly, unless differently specified, we selected the acceleration range [5] such that the engine is able to reach the maximum speed starting from the minimum one in 35 revolutions, obtaining $\alpha^+ = -\alpha^- = 1.62\ 10^{-4}$ rev/msec^2.

A. Task sets generation

Given an overall target utilization U_P for the set of periodic tasks, each periodic tasks is generated as follows:

- The utilization U_i of each task τ_i is randomly generated using the UUniFast algorithm [16] imposing that $\sum_{i=1}^{n} U_i = U_P$. The minimum utilization of each periodic task is set at $U^{min} = 0.005$.
- Task periods T_i are randomly generated with a uniform distribution in the range $[3, 100]$ ms; relative deadlines are set equal to periods (i.e., $D_i = T_i$).
- Computation times are computed as $C_i = U_i T_i$.

The parameters of the AVR task are generated as follows.

- The maximum task utilization is fixed at a U_A, whose value depends on the specific experiment.
- The angular period is set at $\Theta = 2\pi$; the angular deadline is set at $\Delta = \Theta$.
- The number M of modes is randomly generated in a desired range $[M^{min}, M^{max}]$, defined in the specific experiment.
- A random mode m' is selected to have the maximum utilization $U^{m'} = U_A$. The utilization U^m of the other modes $m \neq m'$ is randomly generated in the range $[0.85U_A, U_A]$.
- The maximum speed ω^m of each mode $m < M$ is randomly generated in the range $[1000, 6000]$ RPM. The maximum speed for mode 1 is always set at the maximum speed ω^{max}. Once the mode-transition speeds are generated, they are checked to ensure a minimum separation between any two values. If the actual separation is below $3000/M$ RPM, then all speeds are discarded and the set is generated again.
- The computation time C^m of each mode m is set as $C^m = U^m \Theta / w^m$. If the generated computation times are not monotonically increasing with respect to the modes, they are discarded and the set is generated again.

The variable $U = U_P + U_A$ denotes the total utilization of the overall task set. We also define the parameter $\rho_u = U_A/U$, expressing the ratio of utilization resulting from the AVR task. When tasks are scheduled by FP, task priorities are assigned according to the Rate Monotonic order (i.e., the lower the period, the higher the priority), where the priority of the AVR task is assigned according to its lowest possible inter-arrival time (Θ/ω^{max}).

B. Experiments on schedulability analysis

Both the schedulability tests (for FP and EDF) have been implemented in MATLAB and are able to discriminate 1 RPM in the computation of the dominant speeds. Tests have been executed on a desktop PC with processor Intel i7 running at 3.5 GHz. In all the reported experiments, both the schedulability tests have been executed on 500 randomly generated task sets for each value of the varied parameter.

A first experiment has been carried out to compare the schedulability ratio of EDF and FP when varying the overall utilization U from 0.3 to 0.95 with step 0.05. The results reported in Figure 9 were derived with $\rho_u = 0.4$, $M^{min} = 3$ and $M^{max} = 5$, whereas those in Figure 10 were derived with $\rho_u = 0.6$, $M^{min} = 3$ and $M^{max} = 5$. A further run was performed by increasing the number of modes, using $\rho_u = 0.6$, $M^{min} = 4$ and $M^{max} = 8$, and the results are shown in Figure 11. As evident from the graphs, while EDF is able to schedule all the generated task sets, FP starts degrading for utilizations values that are much lower than those observed in classical periodic task sets. Moreover, such a degradation is more significant for higher values of ρ_u (compare Figure 9 with Figure 10) and for higher numbers of modes of the AVR task (compare Figure 10 with Figure 11).

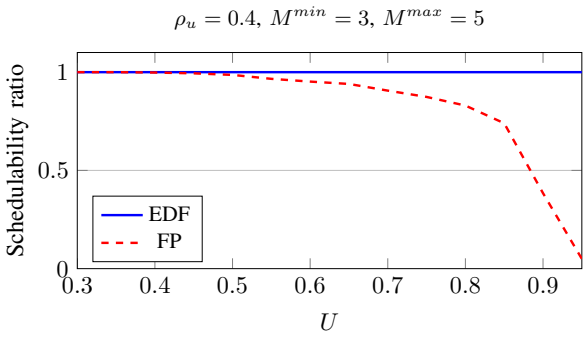

Figure 9. Schedulability ratio as a function of U (by analysis).

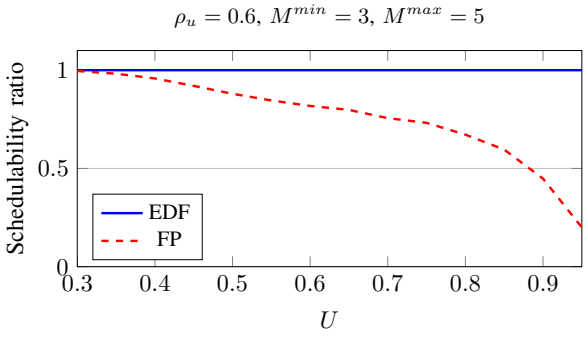

Figure 10. Schedulability ratio as a function of U (by analysis).

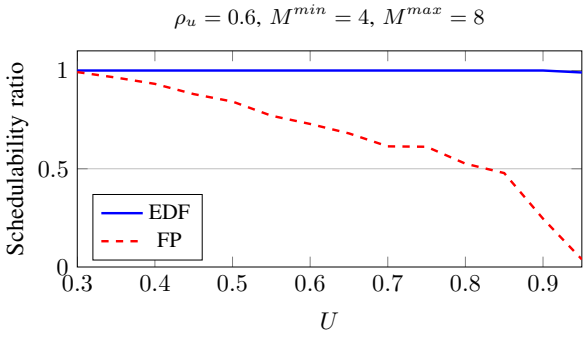

Figure 11. Schedulability ratio as a function of U (by analysis).

A second experiment was carried out to better evaluate the dependency of the schedulability ratio on the fraction of utilization ρ_u imposed by the AVR task. The results of this experiment are reported in Figure 12 for $U = 0.8$, $M^{min} = 3$

and $M^{max} = 5$. Again, while EDF is able to schedule all the tested task sets, the performance of FP scheduling significantly degrades for increasing values of ρ_u.

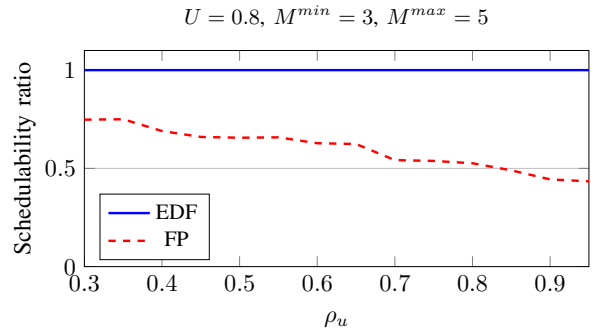

Figure 12. Schedulability ratio as a function of ρ_u (by analysis).

As discussed in the introduction, the sub-optimality of EDF when scheduling AVR tasks derives from the fact that an optimal deadline assignment requires clairvoyance, predicting the future speed evolution pattern of the rotation source. To be conservative, we assumed that each job of an AVR task is assigned the earliest possible deadline, that is the one considering the maximum acceleration. The consequence of this choice is that the analysis is safe, but the deadline assignment error increases with the acceleration range. To show the dependency of this effect on the schedulability, we carried out an experiment where we varied the acceleration range $[\alpha^-, \alpha^+]$ by using the parameter $\alpha = \alpha^+ = -\alpha^-$, holding $U = 0.9$, $\rho_u = 0.4$, $M^{min} = 3$ and $M^{max} = 5$. The results of this experiment are reported in Figure 13.

As predicted, a marginal degradation of the EDF performance starts appearing for accelerations higher than 10^{-3} rev/msec^2. It is worth observing, however, that the results of this experiment have a pure theoretical validity (at least for today's cars), because the values of α at which EDF starts degrading are far beyond the maximum acceleration of real car engines (about $2 \cdot 10^{-3}$ rev/msec^2). In this setting, FP shows a even greater degradation in terms of schedulability ratio as α increases.

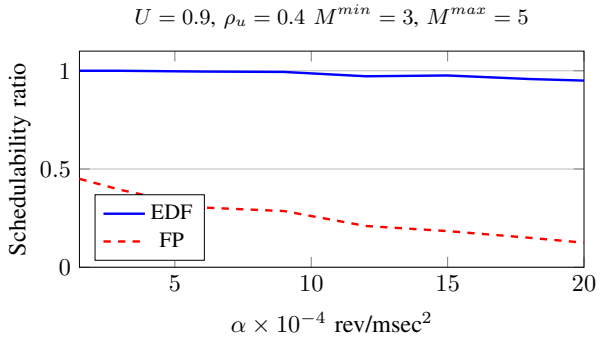

Figure 13. Schedulability ratio as a function of $\alpha = \alpha^+ = -\alpha^-$ (by analysis).

C. Experiments with the scheduling simulator

This section reports some additional experiments that have been carried out on the RTSIM scheduling simulator [15] to evaluate the average-case performance of FP and EDF scheduling in the presence of AVR tasks.

For each initial speed in the range $[500, 6500]$ RPM, with step 500 RPM, the scheduling simulator has been programmed to enforce a critical instant at time $t = 0$ and look at the resulting schedule up to $t = 5$ seconds. We also imposed that all the generated jobs execute for their WCET. Each value plotted in the graphs was obtained as the average over 1000 repetitions.

Figure 14 reports the results of a simulation executed with $\rho_u = 0.4$, $M^{min} = 3$, $M^{max} = 5$, to monitor the schedulability ratio as a function of the overall utilization U. As it can be seen by comparing with the corresponding Figure 9 (obtained under the same parameters), the simulation confirms the same trends observed in the analysis.

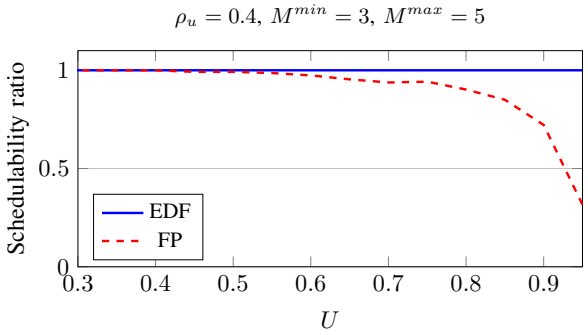

Figure 14. Schedulability ratio as a function of U (by RTSIM).

Figure 15 is the analogous of Figure 12 and reports the schedulability ratio of EDF and FP as a function of ρ_u, for $U = 0.8$, $M^{min} = 3$, $M^{max} = 5$. Again, the same trend is observed, although with a less pronounced degradation of FP, due to the less critical conditions appearing in the simulation.

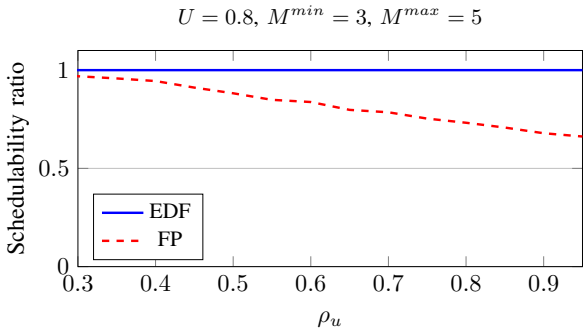

Figure 15. Schedulability ratio as a function of ρ_u (by RTSIM).

A final simulation experiment has been carried out to monitor the tardiness achieved by the two algorithms during overload conditions (up to $U = 1.5$), while the other parameters were fixed at $\rho_u = 0.4$, $M^{min} = 3$, $M^{max} = 5$. Note that the tardiness cannot be derived by the current EDF schedulability test. In this experiment the periods of the periodic tasks were randomly generated in the range $[10, 100]$, thus resulting always greater than the minimum interarrival time of the AVR task, which is then scheduled at the highest priority by FP. The results reported in Figure 16 show the

tardiness achieved by the two schedulers for the AVR task and the lowest priority periodic task.

The results show that, under FP, the AVR tardiness is kept equal to zero even for $U = 1.5$, because the overload does not affect the AVR task (always scheduled at the highest priority), but only the lower priority tasks (the tardiness of the lowest priority task is indicated by the FP-LP curve). On the other hand, under EDF, the AVR tardiness increases with the overload, reaching a value higher than 60 times the deadline for utilizations greater than 1.4. This happens because EDF tends to automatically distribute the exceeding workload to all the tasks [17], hence the lowest priority task (EDF-LP curve) is not so penalized as in the case of FP scheduling.

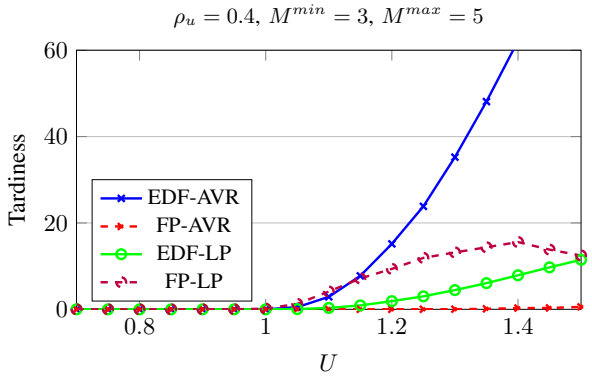

Figure 16. Tardiness as a function of the system load U (by RTSIM).

VI. Conclusions

This paper presented an exact feasibility analysis under EDF scheduling for engine control applications including classic periodic tasks and an AVR task. Extensive experiments have been carried out to compare the schedulability of these types of applications under both EDF and FP scheduling.

The experiments confirmed (providing quantitative measures) that for this type of applications, fixed priorities scheduling is not the best choice, due to the large range in which the interarrival time of an AVR task can vary. Under FP scheduling, this means that there are several engine speeds at which any fixed priority assignment is far from being optimal, significantly penalizing the system schedulability.

In particular, the experiments showed that while EDF is able to guarantee the schedulability of the system up to high utilization values close to 95%, FP scheduling exhibits a significant degradation for utilization values that are much lower than those observed in classical periodic scheduling.

Given the presented schedulability results, and considering that today EDF is actually available in a few operating systems (e.g., Erika Enterprise [18], which is also OSEK-certified for automotive systems, Linux, using the SCHED_DEADLINE scheduling class [19], or the EDF plugin for OSEK/VDX proposed in [20]), we believe that the use of dynamic scheduling in engine control applications would allow a better exploitation of the computing platform, turning into a higher control performance.

Acknowledgements

The authors like to thank Martin Stigge for the interesting exchange of ideas on AVR and DRT task models, Marco Di Natale for valuable suggestions and fruitful discussions and Haibo Zeng for having shared the code for computing the upper bound on the demand bound function for a DRT task.

References

[1] L. Guzzella and C. H. Onder, *Introduction to Modeling and Control of Internal Combustion Engine Systems*. Springer-Verlag, 2010.

[2] D. Buttle, "Real-time in the prime-time," in *Keynote speech at the 24th Euromicro Conference on Real-Time Systems*, Pisa, Italy, July 12, 2012.

[3] J. Kim, K. Lakshmanan, and R. Rajkumar, "Rhythmic tasks: A new task model with continually varying periods for cyber-physical systems," in *Proc. of the Third IEEE/ACM Int. Conference on Cyber-Physical Systems (ICCPS 2012)*, Beijing, China, April 17-19, 2012, pp. 28–38.

[4] V. Pollex, T. Feld, F. Slomka, U. Margull, R. Mader, and G. Wirrer, "Sufficient real-time analysis for an engine control unit with constant angular velocities," in *Proc. of the Design, Automation and Test Conference in Europe*, Grenoble, France, March 18-22, 2013.

[5] R. I. Davis, T. Feld, V. Pollex, and F. Slomka, "Schedulability tests for tasks with variable rate-dependent behaviour under fixed priority scheduling," in *Proc. 20th IEEE Real-Time and Embedded Technology and Applications Symposium*, Berlin, Germany, April 15-17, 2014.

[6] A. Biondi, A. Melani, M. Marinoni, M. D. Natale, and G. Buttazzo, "Exact interference of adaptive variable-rate tasks under fixed-priority scheduling," in *Proceedings of the 26th Euromicro Conference on Real-Time Systems (ECRTS 2014)*, Madrid, Spain, July 8-11, 2014.

[7] A. Biondi, M. D. Natale, and G. Buttazzo, "Response-time analysis for real-time tasks in engine control applications," in *Proceedings of the 6th International Conference on Cyber-Physical Systems (ICCPS 2015)*, Seattle, Washington, USA, April 14-16, 2015.

[8] C. Liu and J. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the Association for Computing Machinery*, vol. 20, no. 1, pp. 46–61, January 1973.

[9] G. Buttazzo, E. Bini, and D. Buttle, "Rate-adaptive tasks: Model, analysis, and design issues," in *Proc. of the Int. Conference on Design, Automation and Test in Europe (DATE 2014)*, Dresden, Germany, March 24-28, 2014.

[10] A. Biondi and G. Buttazzo, "Engine control: Task modeling and analysis," in *Proc. of the International Conference on Design, Automation and Test in Europe (DATE 2015)*, Grenoble, France, March 9-13, 2015.

[11] Z. Guo and S. Baruah, "Uniprocessor EDF scheduling of AVR task systems," in *Proc. of the ACM/IEEE 6th International Conference on Cyber-Physical Systems (ICCPS 2015)*, Seattle, USA, April 2015.

[12] S. Baruah, L. Rosier, and R. Howell, "Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor," *Journal of Real-Time Systems*, vol. 2, 1990.

[13] M. Stigge, P. Ekberg, N. Guan, and W. Yi, "The digraph real-time task model," in *Proc. 17th Real-Time and Embedded Technology and Applications Symposium (RTAS'11)*, 2011.

[14] H. Zeng and M. D. Natale, "Using max-plus algebra to improve the analysis of non-cyclic task models," in *Proceedings of the 25th Euromicro Conference on Real-Time Systems (ECRTS 2013)*, July 2013.

[15] The RTSIM scheduling simulator. [Online]. Available: http://rtsim.sssup.it/

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

[17] G. C. Buttazzo, "Rate monotonic vs. EDF: Judgment day," *Real-Time Systems*, vol. 29, no. 1, pp. 5–26, January 2005.

[18] "Erika enterprise: an OSEK compliant real-time kernel." [Online]. Available: http://erika.tuxfamily.org/drupal/

[19] D. Faggioli, F. Checconi, M. Trimarchi, and C. Scordino, "An edf scheduling class for the linux kernel," in *Proc. of the 11th Real-Time Linux Workshop (RTLWS)*, Dresden, Germany, September 28-30, 2009.

[20] C. Diederichs, U. Margull, F. Slomka, and G. Wirrer, "An application-based edf scheduler for OSEK/VDX," in *Proc. of the Design, Automation and Test Conference in Europe (DATE 2008)*, Munich, Germany, March 10-14 2008.