Response-Time Analysis of Engine Control Applications under Fixed-Priority Scheduling

Alessandro Biondi, Marco Di Natale *Senior Member, IEEE*, Giorgio Buttazzo *Fellow, IEEE*
Scuola Superiore Sant'Anna, Pisa, Italy
Email: {alessandro.biondi, marco, giorgio}@sssup.it

Abstract—Engine control systems include computational activities that are triggered at predetermined angular values of the crankshaft, and therefore generate a workload that tends to increase with the engine speed. To cope with overload conditions, a common practice adopted by the automotive industry is to design such angular tasks with a set of modes that switch at given rotation speeds to adapt the computational demand. This paper presents an exact response time analysis for engine control applications consisting of periodic and engine-triggered tasks scheduled by fixed priority. The proposed analysis explicitly takes into account the physical constraints of the considered systems and is based on the derivation of dominant speeds, which are particular engine speeds that are proved to determine the worst-case behavior of engine-triggered tasks from a timing perspective. Experimental results are finally reported to validate the proposed approach and compare it against an existing sufficient test.

Index Terms—engine-control, cyber-physical systems, real-time analysis

1 Introduction

Engine control application belong to an interesting class of real-time applications that are not suitably represented by a periodic or sporadic task model, since the activation of one or more tasks of interest occurs on a given angular position of the engine shaft. In addition, to leverage at best the computational resources, more complex control functions are defined for low rates of the engine. When the engine speed increases, computational load is shed, giving rise to the Adaptive Variable Rate (AVR) task model.

In a 4-cylinder engine, for example, the injection of the fuel for the odd numbered cylinders follows a cycle of two rotations and is in phase opposition with the corresponding injection for the even cylinders. Conventionally, the rotation of the engine crankshaft and the phase within it are referred to the Top Dead Center (or TDC) position of one of the cylinders. When the engine speed increases, the code complexity of some tasks is reduced and correspondingly, their worst-case execution time is lowered. These modes of execution of variable complexity operate within given engine speed ranges, defined at design time.

The timing analysis of applications that include AVR tasks is not trivial, since the identification of the possible worst-case scenario depends on the initial speed, the transition speeds, and the worst-case execution times for each mode. In addition, the worst-case scenario for a task also depends on the possible evolution of the engine speed according to the physics of the engine, defined at least by boundaries on the maximum and minimum angular acceleration.

The definition of the transition speeds and the control task implementations for the different modes are defined to optimize a set of performance indexes, related to power, fuel consumption, and emissions (among others) within schedulability constraints. This process requires a fine tuning of a significant number of configuration parameters, often performed manually at the test bench.

Contribution. This paper presents an exact analysis (with respect to a general physical model of the engine dynamics) for a mixed task set that includes both regular periodic/sporadic tasks and AVR tasks managed under fixed-priority scheduling, the policy mandated by the AUTOSAR standard (adopted by the vast majority of automotive companies). The analysis is valid for uniprocessor systems and multiprocessor systems managed by partitioned fixed-priority scheduling.

The main purpose of the presented analysis is to *explicitly* take into account the *physical constraints* of the considered system during the characterization of the maximum computational demand generated by AVR tasks. In this way, it is possible to precisely study mode-changes and release patterns of AVR tasks, thus enabling the derivation of a method for precisely computing their temporal *interference* on low-priority tasks. The analysis integrates, extends and clarifies previous work by the same authors. A full description of the novel contributions is in Section 8 together with the discussion of the state of the art.

A model of engine-control applications for the purpose of real-time analysis is first presented in Section 2, including a model for the dynamics of a rotating crankshaft. The latter is generalized in Section 7. The proposed analysis technique is based on approaching the computation of the interference as a search problem in the speed domain, which is discussed and formalized in Section 3. Then, in Section 4, the search space is studied to identify a set of pruning conditions. The problem is demonstrably solvable by only considering a limited set of engine speeds (denoted as *dominant speeds*), which allows computing the maximum response time of tasks by studying some specific scenarios. Based on these results, an algorithm is designed to efficiently perform response-time analysis (Section 5). Section 6 reports an experimental study that has been conducted to assess the performance of the proposed approach and evaluate it against the previous work. Section 8 discusses the related work and Section 9 concludes the paper.

2 System model

This work considers a single rotation source (the crankshaft of one engine) characterized by the following state variables:

- the rotation angle (θ);
- the angular velocity (ω);
- the angular acceleration (α).

It is assumed that the angular velocity ω is limited within the range [ω^-, ω^+] and the acceleration α is limited within the range [α^-, α^+].

Section 2.1 introduces a model for engine control applications for the purpose of real-time (timing) analysis. To the best of our knowledge and experience with a number of automotive industries, the proposed model is appropriate for describing a wide representative set of engine control applications.

Then, Section 2.2 presents a model for the physical dynamics of the rotation source. To ease the presentation, such a model is based on the simplifying assumption of *constant* acceleration during a given angular interval within one crankshaft revolution. This assumption is relaxed in a generalized model (reported in Section 7), which allows computing minimum inter-arrival times under arbitrary acceleration profiles with unbounded jerk (i.e., infinite rate of change of the acceleration) — a conservative assumption that avoids incurring in excessive complications for the purpose of this work. Thanks to a set of monotonicity properties derived in the following, *the presented results are compatible with both models.*

2.1 Application model

The considered engine-control applications consist of a set $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\}$ of n real-time preemptive tasks. Each task can either be *periodic* (i.e., activated at fixed time intervals), *sporadic* (i.e., activated with a minimum inter-arrival time) or an *angular task* (i.e., activated at specific crankshaft rotation angles). Considering that angular tasks have a variable inter-arrival 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/sporadic 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 inter-arrival time (or period) T_i, and a relative deadline D_i. However, while for regular periodic/sporadic 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 angles $\theta_i = \Phi_i + k\Theta_i$, for $k = 0, 1, 2, \ldots$. This means that, when the engine is rotating at a fixed speed ω, the inter-arrival time of an AVR task is *inversely proportional* to the engine speed and can be expressed as $T_i(\omega) = \Theta_i/\omega$.

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 [10] 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 $\omega_i^{M_i+1} = \omega^-$ and $\omega_i^1 = \omega^+$. 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 WCET $C_{i,k}$ of an arbitrary AVR job $J_{i,k}$ is expressed as a non-increasing step function $\mathcal{C}_i(\omega)$ of the instantaneous speed ω *at its release*, that is,

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

An example of a $C_i(\omega)$ function is shown in Figure 1.

The implementation of AVR tasks can be performed as a sequence of conditional `if` statements, each executing a specific subset of functions [9], [10] (also denoted as *runnables* in the automotive domain). Figure 2 illustrates a sample AVR task with four modes, $\omega_- = 500$ RPM, and $\omega_+ = 6500$ RPM. This example assumes that the `read_rotation_speed()` function returns the instantaneous speed ω at the task activation time (not at the calling time of the function).

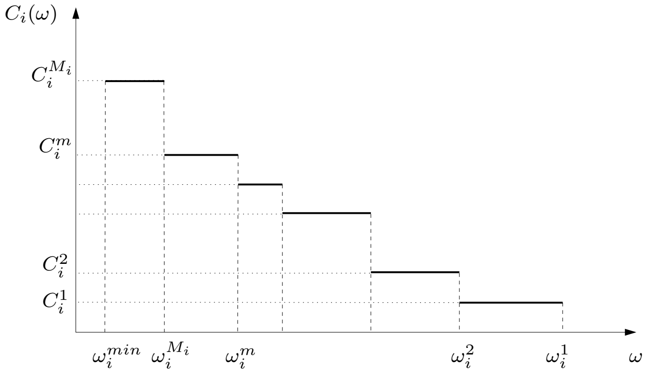


Figure 1. Worst-case execution time of an AVR task as a function of the speed at the job activation.

```
task sample_AVR_task {
    omega = read_rotation_speed();
    if (omega ≤ 6500) f1();
    if (omega ≤ 5000) f2();
    if (omega ≤ 3500) f3();
    if (omega ≤ 2000) f4();
}
```

Figure 2. Implementation of an example AVR task.

2.2 Rotation source model

For the purpose of analyzing the timing properties of engine control applications that include AVR tasks, it is crucial to characterize the relation between the AVR task parameters and the dynamics of the engine. In this section we make the simplifying assumption that the engine acceleration has a negligible variation during two consecutive jobs of an AVR task, and is hence assumed *constant* within the angular period of AVR tasks. Such an assumption significantly simplifies the derivation of the inter-arrival time between two consecutive jobs as a function of the engine state. On the other hand, it can lead to the computation of optimistic (overestimated) inter-arrival times. Although this model has this drawback, numerical evaluations reveal that the error introduced by the assumption of constant acceleration is very marginal, especially when considering realistic speed and acceleration bounds. Hence, in the following sections, this simplified model is used to ease the presentation of the problem and of the solution. However, we idenfity the fundamental properties on which all the proofs and the analysis are based and in Section we show how the method still applies to a quite general model of the dynamics that removes all the limitations and inaccuracies of the constant acceleration assumption.

Suppose that a job $J_{i,k}$ of an AVR task τ_i^* is released at time t_k with instantaneous engine speed ω_k. Following standard physical equations (e.g., as presented in [9]), the release time t_{k+1} of the next AVR job $J_{i,k+1}$ assuming a constant acceleration α_k during $(t_k, t_{k+1}]$ can be computed as $t_{k+1} = t_k + T_i(\omega_k, \alpha_k)$, where

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

In a similar way, the instantaneous engine speed $\omega_{k+1} = \Omega(\omega_k, \alpha_k)$ at the release of the next job $J_{i,k+1}$ can be computed as $\omega_k + \alpha_k T_i(\omega_k, \alpha_k)$, which gives:

$$\Omega_i(\omega_k, \alpha_k) = \sqrt{\omega_k^2 + 2\Theta_i \alpha_k}. \quad (3)$$

If two consecutive jobs $J_{i,k}$ and $J_{i,k+1}$ are respectively released when the engine has instantaneous speeds ω_k and ω_{k+1}, the inter-arrival time $\widetilde{T}_i(\omega_k, \omega_{k+1})$ between the two jobs can be obtained by Equation (2), substituting α_k from Equation (3), which gives:

$$\widetilde{T}_i(\omega_k, \omega_{k+1}) = \frac{2\Theta_i}{\omega_k + \omega_{k+1}}. \quad (4)$$

Considering a job $J_{i,k}$ released with instantaneous speed ω_k, Equation (4) can be used to compute the minimum inter-arrival time $\widetilde{T}_i^m(\omega_k)$ such that the next job $J_{i,k+1}$ is released in mode m (if reachable with the acceleration bounds)

$$\widetilde{T}_i^m(\omega_k) = \widetilde{T}_i(\omega_k, \omega_i^m) = \frac{2\Theta_i}{\omega_k + \omega_i^m}. \quad (5)$$

Finally, given a job $J_{i,k}$ released with instantaneous speed ω_k and the inter-arrival time T to the next job $J_{i,k+1}$, we define $\widetilde{\Omega}(\omega_k, T)$ as the instantaneous speed at the release $J_{i,k+1}$, computed from Equation (4):

$$\widetilde{\Omega}_i(\omega_k, T) = \frac{2\Theta_i}{T} - \omega_k. \quad (6)$$

It is also convenient to define the inverse function of Equation (3), representing the initial speed ω_k that allows reaching speed ω_{k+1} with constant acceleration α_k, that is,

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

In the analysis presented in the following sections, we also define the engine speed after n job releases (following an arbitrary k^{th} job), with constant acceleration α during $(t_k, t_{k+n}]$; such a value, denoted as Ω^n, can be recursively computed as $\Omega^n(\omega_k, \alpha) = \Omega(\Omega^{n-1}(\omega_k, \alpha), \alpha)$, where $\Omega^0(\omega, \alpha) = \omega$. Similarly as in Equation (7), we define the inverse function $\Omega^{-n}(\omega_{k+n}, \alpha) = \Omega^-(\Omega^{-(n-1)}(\omega_{k+n}, \alpha), \alpha)$, where $\Omega^{-0}(\omega, \alpha) = \omega$.

A summary of the notation is shown in Table 1.

2.3 Monotonicity of inter-arrival times

Consider two consecutive jobs $J_{i,k}$ and $J_{i,k+1}$ with $J_{i,k+1}$ released at a *given* speed ω_{k+1}. Let ω_k and ω'_k be two possible speeds at the release of $J_{i,k}$. If $\omega_k > \omega'_k$ then $\widetilde{T}_i(\omega_k, \omega_{k+1}) < \widetilde{T}_i(\omega'_k, \omega_{k+1})$. That is, the higher the speed at which $J_{i,k}$ is released, the lower the inter-arrival time to the next job. Similarly, consider now the case in which $J_{i,k}$ is released at a *given* speed ω_k and let ω_{k+1} and ω'_{k+1} be two possible speeds at the release of $J_{i,k+1}$. If $\omega_{k+1} > \omega'_{k+1}$, then also $\widetilde{T}_i(\omega_k, \omega_{k+1}) < \widetilde{T}_i(\omega_k, \omega'_{k+1})$. Finally, the function $\widetilde{T}_i(\omega_k, \omega_{k+1})$ is simultaneously decreasing in the two variables, that is, if $\omega_k > \omega'_k$ and $\omega_{k+1} > \omega'_{k+1}$ then $\widetilde{T}_i(\omega_k, \omega_{k+1}) < \widetilde{T}_i(\omega'_k, \omega'_{k+1})$.

To end of generalizing the presented results, the monotonicity of the function $\widetilde{T}_i(\omega_k, \omega_{k+1})$ (in both its variables) is used in the following sections as a fundamental hypothesis to identify the dominant speeds and construct the analysis methods presented in this paper. In Section 7, these properties are shown to also apply to a very general model of the engine dynamics.

3 Addressing interference as a search problem

Under fixed-priority scheduling, a task suffers *interference* whenever it is prevented to execute due to the execution of higher-priority tasks. This section explains how to compute the interference generated by an AVR task on a set of lower priority tasks.

Let J_0 be a job of an AVR task τ^* activated at time $t = 0$ with a speed ω_0, as shown in Figure 3, and suppose that the job executes for its WCET $\mathcal{C}(\omega_0)$. Since the engine has acceleration $\alpha \in [\alpha^-, \alpha^+]$, there can be *infinite* instants of time at which the next job can be activated. The earliest job activation time is given by the maximum acceleration α^+ and occurs after $T(\omega_0, \alpha^+)$ time units, while the latest activation time occurs at the maximum deceleration α^- after $T(\omega_0, \alpha^-)$ units of time.

The execution mode of the next job J_1 (and hence its WCET) depends on the instantaneous speed of the engine at its activation. Figure 3 reports the *single-job interference* function $i_{\omega_0}(t)$ representing the envelope of the interference contribution among all the possible subsequent jobs. The instantaneous angular velocity ω_1 at the activation of J_1 is

Table 1
Main notation introduced in the system model.

Symbol	Description
τ_i	i^{th} periodic task
τ_i^*	i^{th} AVR task
$\mathcal{C}_i(\omega)$	WCET of τ_i^* as a function of the inst. speed
C_i^m	WCET of mode m of τ_i^*
ω_i^m	Maximum speed for mode m of τ_i^*
Θ_i	Angular period of τ_i^*
Γ	Task set
Γ_P	Subset of Γ composed of periodic tasks
Γ_A	Subset of Γ composed of AVR tasks
$T_i(\omega_k, \alpha_k)$	Inter-arrival time between the k^{th} and $(k+1)^{\text{th}}$ job instances, with constant acceleration α_k
$\Omega_i(\omega_k, \alpha_k)$	Speed at the release of job $J_{i,k+1}$ assuming $J_{i,k}$ released at speed ω_k and acceleration α_k
$\widetilde{\Omega}_i(\omega_k, T)$	Speed at the release of job $J_{i,k+1}$ assuming $J_{i,k}$ released at speed ω_k and the inter-arrival time between $J_{i,k}$ and $J_{i,k+1}$ is T
$\widetilde{T}_i(\omega_k, \omega_{k+1})$	Inter-arrival time between a job released at speed ω_k and the following at speed ω_{k+1}
$\widetilde{T}_i^m(\omega_k)$	Minimum inter-arrival time between a job released at speed ω_k and the following in mode m
$\Omega_i^n(\omega_k, \alpha)$	Speed after n jobs releases following job $J_{i,k}$ released at speed ω_k with constant acceleration α

bounded in the range $[\Omega(\omega_0, \alpha^-), \Omega(\omega_0, \alpha^+)]$ and depends on the actual acceleration of the engine during the inter-arrival time. Figure 4 illustrates the *tree* of possible job sequences that results by recursively applying such reasoning to each job generated after J_1.

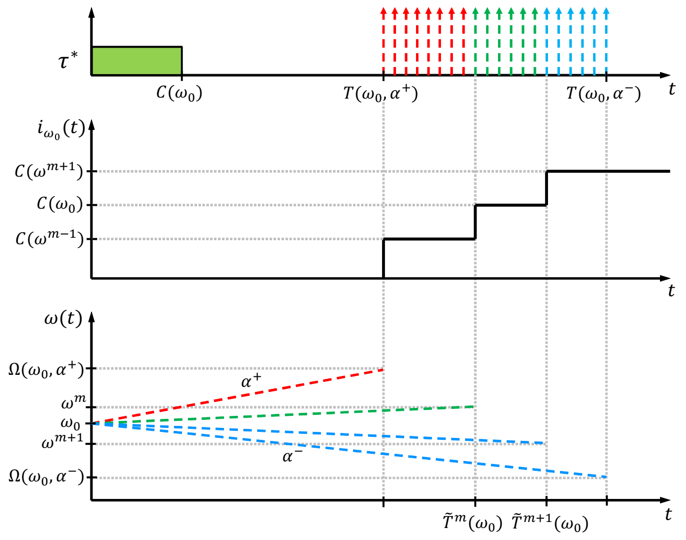
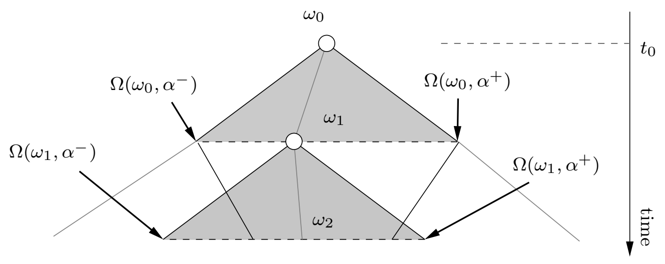


Figure 3. Possible activations after a job released at speed ω_0. Different colors indicate different modes of the AVR task. Constant acceleration between two consecutive jobs is assumed. The interference $i_{\omega_0}(t)$ generated by the next job is also shown.

The computation of the interference of an AVR task can then be considered as a search problem in the speed domain, where all possible job sequences and the composition of the corresponding single-job interferences have to be analyzed in a given time interval. Note that, being the speed domain continuous, the search tree is infinite, that is, it includes an infinite set of job sequences. This fact implies that any brute-force search algorithm must quantize the speed domain in order to produce a solution in a finite amount of time.

In addition, since the release of the first job must be considered for each instantaneous speed ω_0 of the AVR set, the search algorithm has to be applied for each speed $\omega_0 \in [\omega^-, \omega^+]$. Therefore, a speed quantization is also needed, further complicating the problem.



ω_0
t_0
$\Omega(\omega_0, \alpha^-)$
$\Omega(\omega_0, \alpha^+)$
ω_1
$\Omega(\omega_1, \alpha^-)$
$\Omega(\omega_1, \alpha^+)$
ω_2
time

Figure 4. Search tree representing the possible job sequences for an AVR task.

The pseudo code of a brute-force search of the speed tree using quantization is reported in Figure 5. Starting with a job J_0 released at $t = 0$ with speed ω_0, the procedure is called as `Interference`$(\omega_0, \mathcal{C}(\omega_0), 0)$. The MAXTIME parameter represents the length of the time interval within which the interference needs to be computed. Each recursive instance of the `Interference` procedure represents a job activated at time t with instantaneous speed ω and Π is the sum of all the computational requests imposed by the previous jobs. At each recursive step, the algorithm **(i)** terminates a branch when reaching the end of the time interval of interest (lines 3-4); **(ii)** keeps track of the computational requests accumulated at time t via the sub-procedure UPDATEINTERFERENCE (line 4); **(iii)** explores (with quantization) the speed domain allowed by the acceleration bounds α^- and α^+ by computing the inter-arrival time to the next job (lines 6), accumulating the overall computational request (line 7), and recursively calling the function INTERFERENCE to explore the sub-tree (line 8).

Besides providing only an approximate (and possibly unsafe) analysis due to quantization, this approach is very expensive in terms of computational complexity and intractable for most practical cases. In the following, the problem is formalized in order to derive a method for exploring

```
1:  procedure INTERFERENCE($\omega, \Pi, t$)
2:      if $t >$ MAXTIME then return ;
3:      end if
4:      UPDATEINTERFERENCE($\Pi, t$);
5:      for $\omega^{next} = \Omega(\omega, \alpha^-)$ to $\Omega(\omega, \alpha^+)$ step $\Delta\omega$ do
6:          $T^{next} \leftarrow \widetilde{T}(\omega, \omega^{next})$;
7:          $\Pi^{next} \leftarrow \Pi + \mathcal{C}(\omega^{next})$;
8:          INTERFERENCE($\omega^{next}, \Pi^{next}, t + T^{next}$);
9:      end for
10: end procedure
```

Figure 5. Procedure for computing the interference of an AVR task using brute-force on the search domain.

the speed domain with a tractable complexity still providing an exact interference analysis.

3.1 Formalization

Definition 1. A job sequence s of an AVR task τ^* is a sequence of consecutive jobs $J_0, \ldots, J_{n_s}$, where each job J_k is released with instantaneous engine speed ω_k.

Definition 2. A job sequence s is *valid* if any two consecutive jobs are released at speeds that are compatible with the acceleration range; that is, $\forall \omega_k, k = 1, \ldots, n_s,\ \omega_k \in [\Omega(\omega_{k-1}, \alpha^-), \Omega(\omega_{k-1}, \alpha^+)]$.

The interference of an AVR task is characterized by an infinite set of possible valid job sequences in a given time window $[0, t]$. Let $\mathcal{S}(t)$ be such a set. Intuitively, each path in the search tree represents a job sequence.

Each sequence $s \in \mathcal{S}(t)$ generates an interference $I^{(s)}(t)$, which is a function of ω_0 and $\omega_k, k = 1, \ldots, n_s$, because it depends on the speed evolution pattern experienced by the AVR task. In general, $I^{(s)}(t)$ can be expressed as

$$I^{(s)}(t) = \mathcal{C}(\omega_0) + \sum_{k=1}^{n_s} \mathcal{C}(\omega_k)\ \mathsf{step}\left(t - \sum_{j=1}^{k} \widetilde{T}(\omega_{j-1}, \omega_j) \right),$$

where

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

Ideally, to cope with all possible speed evolution patterns, *all* the job sequences $s \in \mathcal{S}(t)$ have to be considered to obtain a characterization of the interference (as in the algorithm of Figure 5). Clearly, this is not viable for practical purposes. The following sections present a technique for *drastically* reducing the number of job sequences that must be explored, while still guaranteeing an exact characterization of the interference.

4 Reducing the Search Space Through Dominant Speeds

For each job released at a given speed, only a finite set of following job releases must be taken into account to derive the maximum interference. This section first explains how to compute such critical job releases and then derives a pruning method for the search problem presented in the previous section. Before proceeding, it is necessary to formalize the notion of single-job interference.

4.1 Single-job interference

To compute the potential interference generated by a single job J_a, it is necessary to consider all the possible activations of the *next* job J_{a+1} that are compatible with the acceleration range $[\alpha^-, \alpha^+]$.

Definition 3. Given a job J_a of an AVR task released at engine speed ω_a at time t_a, the *single-job interference* $i_{\omega_a}(\delta)$ of J_a is the maximum computational request generated by J_a and the next job J_{a+1}, in the interval $[t_a, t_a + \delta]$, for all possible releases of J_{a+1} at $t_{a+1} = t_a + \delta$.

As shown in Section 3, the activations of a job are related to the engine dynamics, and the future release times and modes of J_{a+1} are constrained by the maximum/minimum acceleration of the engine. At time t_a, it is $i_{\omega_a}(0) = \mathcal{C}(\omega_a)$ to account for the computational request of J_a. If the maximum acceleration of the engine is α^+, then clearly no job can be activated in $[t_a, t_a + \delta]$, if $\delta \in [0, T(\omega_a, \alpha^+))$; hence, in this interval $i_{\omega_a}(\delta) = \mathcal{C}(\omega_a)$.

For $T(\omega_a, \alpha^+) \leq \delta \leq T(\omega_a, \alpha^-)$, a release of the next job J_{a+1} is possible and must be correspondingly considered by $i_{\omega_a}(\delta)$. The earliest possible release occurs in the case of maximum acceleration α^+, while the latest occurs in the case of maximum deceleration α^-. Depending on the engine dynamics, J_{a+1} can be activated in a number of different modes. The larger the acceleration/deceleration range, the greater the number of possible modes. Being $[\Omega(\omega_a, \alpha^-), \Omega(\omega_a, \alpha^+)]$ the range of possible engine speeds at the release of J_{a+1}, such a job can be in any mode m' such that $\omega^{m'} \in [\Omega(\omega_a, \alpha^-), \Omega(\omega_a, \alpha^+)]$.

For $\delta > T(\omega_a, \alpha^-)$, there are no releases of J_{a+1}; therefore, the interference is given by the computational request of the latest possible job release time, that is, $i_{\omega_a}(\delta) = \mathcal{C}(\Omega(\omega_a, \alpha^-))$. In general, $i_{\omega_a}(\delta)$ is a non-decreasing stepwise function, where each step represents the release of a different mode m'. An example of single-job interference is illustrated in Figure 3 (plot in the middle).

Based on the above definition, it is possible to derive a theorem that states a dominance condition between the single-job interferences of two jobs.

Theorem 1. *Let J_a and J_b be two jobs released in mode m, and let ω_a and ω_b be the instantaneous engine speeds at their respective release times. If $\omega_a \geq \omega_b$ and $\mathcal{C}(\Omega(\omega_a, \alpha^-)) = \mathcal{C}(\Omega(\omega_b, \alpha^-))$, then $\forall \delta \geq 0,\ \ i_{\omega_a}(\delta) \geq i_{\omega_b}(\delta)$.*

Proof. The proof is trivial for $\omega_a = \omega_b$. Hence, let us assume $\omega_a > \omega_b$. Since, for a given α, both $T(\omega, \alpha^+)$ and $T(\omega, \alpha^-)$ are monotonic decreasing functions in ω, we have:

(i) $T(\omega_a, \alpha^+) \leq T(\omega_b, \alpha^+)$;
(ii) $T(\omega_a, \alpha^-) \leq T(\omega_b, \alpha^-)$.

From (i) we can derive that $i_{\omega_a}(\delta) = i_{\omega_b}(\delta) = C^m$ for $\delta < T(\omega_a, \alpha^+)$. For $T(\omega_a, \alpha^+) \leq \delta < T(\omega_b, \alpha^+)$ we have $i_{\omega_b}(\delta) = C^m$ (job releases after J_b cannot occur before $T(\omega_b, \alpha^+)$), while $i_{\omega_a}(\delta)$ is larger because of the possible job releases following J_a. Hence, in the range $T(\omega_a, \alpha^+) \leq \delta < T(\omega_b, \alpha^+)$, we have $i_{\omega_a}(\delta) > i_{\omega_b}(\delta)$.

For $\delta \geq T(\omega_b, \alpha^+)$ two scenarios are possible:

- $T(\omega_b, \alpha^+) \leq T(\omega_a, \alpha^-)$, i.e., the steps of the two single-job interference functions are overlapped in time. Consider a fixed (but arbitrary) time instant δ in this range, which corresponds to the inter-arrival time to the next jobs—namely J_{a+1} and J_{b+1}, respectively. In this case, for $T(\omega_b, \alpha^+) \leq \delta \leq T(\omega_a, \alpha^-)$, we have $\widetilde{\Omega}(\omega_a, \delta) < \widetilde{\Omega}(\omega_b, \delta)$. Therefore, job J_{a+1} will always be released at an higher speed than J_{b+1}. As a result, being $\mathcal{C}(\omega)$ non-increasing, $\mathcal{C}(\widetilde{\Omega}(\omega_a, \delta)) > \mathcal{C}(\widetilde{\Omega}(\omega_b, \delta))$, hence $i_{\omega_a}(\delta) \geq i_{\omega_b}(\delta)$.
  Finally, for $\delta > T(\omega_a, \alpha^-)$, $i_{\omega_a}(\delta) = \mathcal{C}(\Omega(\omega_a, \alpha^-)) + C^m$. By hypothesis, we note that the maximum computational request of J_{b+1} is $\mathcal{C}(\Omega(\omega_b, \alpha^-)) = \mathcal{C}(\Omega(\omega_a, \alpha^-))$. Hence, $i_{\omega_a}(\delta) \geq i_{\omega_b}(\delta)$.

- $T(\omega_b, \alpha^+) > T(\omega_a, \alpha^-)$, i.e., the two single-job interference function are non-overlapped in time. This case follows as the one discussed above for $\delta > T(\omega_a, \alpha^-)$.

Having shown that $i_{\omega_a}(\delta) \geq i_{\omega_b}(\delta)$ in each possible time interval, the theorem follows. □

Figure 6 shows a typical scenario in which Theorem 1 holds, related to the case $T(\omega_b, \alpha^+) \leq T(\omega_a, \alpha^-)$.

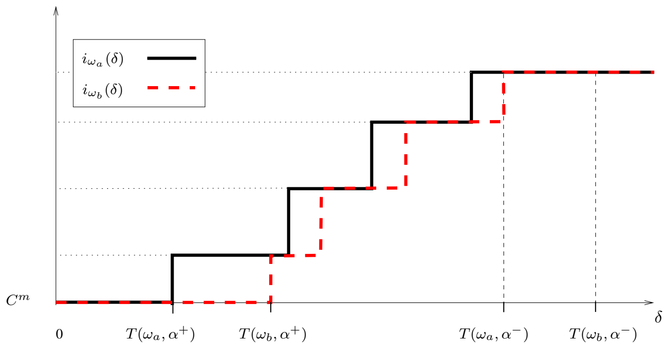


Figure 6. Example of a scenario in which Theorem 1 holds.

4.2 Pruning conditions for the search space

Unfortunately, Theorem 1 is not sufficient to discard *all* the single-job interferences generated by the jobs following a job J_b that is compliant with the theorem hypothesis.

For example, consider a generic job instance J_b for which the single-job interference is dominated by the one of job J_a. Since (by hypothesis) $\omega_a \geq \omega_b$, a job J_{b+1} immediately following J_b could be released at a speed *lower* than those of *all* the possible jobs instances J_{a+1} immediately following J_a. As a consequence, at the following step, job J_{b+2} (immediately following J_{b+1}) can also be released at speeds lower than all possible jobs instances J_{a+2} (immediately following J_{a+1}). Since function $\mathcal{C}(\omega)$ can assume higher values for lower speeds ω, the single-job interference generated by J_{b+1} can be *higher* than those of all possible jobs J_{a+1}. A situation in which this happens is illustrated in Figure 7.

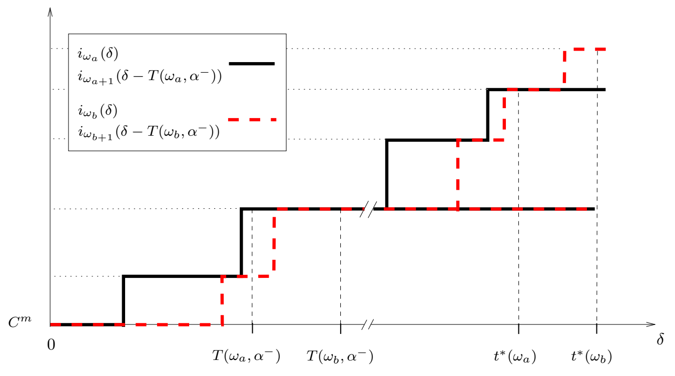


Figure 7. Example in which Theorem 1 is not sufficient to discard *all* the interferences generated by the jobs following J_b. There exists a job J_{b+2} that is released at a speed that is lower than the one at which any job J_{a+2} can be released. Consequently, the interference caused by jobs following J_b (dashed line) is not dominated by the one generated by jobs following J_a. The function $t^*(\omega)$ in the graph is defined as $t^*(\omega) = T(\omega, \alpha^-) + T(\Omega(\omega, \alpha^-), \alpha^-)$.

Formally speaking, the minimum speeds at which J_{a+2} and J_{b+2} can be respectively activated are $\omega_{a+2} = \Omega^2(\omega_a, \alpha^-)$ and $\omega_{b+2} = \Omega^2(\omega_b, \alpha^-)$ with $\omega_{a+2} \leq \omega_{b+2}$. Therefore, since it can happen that $\mathcal{C}(\omega_{b+2}) > \mathcal{C}(\omega_{a+2})$, there exist some instances of J_{b+2} that cannot be pruned in favor of all valid sequences J_a, J_{a+1}, J_{a+2}.

Taken in isolation, Theorem 1 cannot provide an effective pruning condition to compute the maximum interference. What is needed is a pruning method that allows discarding an *entire sub-tree* during the search for the maximum interference, i.e., *all* possible job sequences following a job J_b in favor of *all* possible sequences following another job J_a.

Before proceeding, it is convenient to introduce the notion of *interference envelope*.

Definition 4. The interference envelope $I_{\omega_0}(t)$ is the maximum interference produced by *all* possible valid job sequences $J_0, \ldots, J_n$ in the interval $[0, t]$, with J_0 released at time $t = 0$ and speed ω_0.

Thanks to this definition, it is possible to precisely define the objective of this section, that is finding a method to discard a job J_b in favor or another job J_a (respectively released at speeds ω_b and ω_a) such that $\forall t \geq 0,\ I_{\omega_a}(t) \geq I_{\omega_b}(t)$.

We begin by noting that, under particular conditions, the maximum interference generated by some job sequences is dominated by an interference envelope.

Lemma 1. Consider an arbitrary (but valid) job sequence $s = J_b, J_{b+1}, \ldots, J_{b+n}$. Let J_a and J_b be two jobs released in mode m at time $t = 0$ and at speeds ω_a and ω_b, respectively, with $\omega_a \geq \omega_b$. If job J_{b+1} is released at a speed $\omega_{b+1} \in [\Omega(\omega_a, \alpha^-), \Omega(\omega_a, \alpha^+)]$, then $\forall t \geq 0,\ I_{\omega_a}(t) \geq I^{(s)}(t)$.

Proof. Since $\omega_{b+1} \geq \Omega(\omega_a, \alpha^-)$, then job J_{a+1} (immediately following J_a) can also be activated at speed ω_{b+1}. Let $t_{a+1} = \widetilde{T}(\omega_a, \omega_{b+1})$ be the inter-arrival time between J_a and J_{a+1} if the latter is activated at speed ω_{b+1}. $\forall t \geq 0$, any valid job sequence starting with J_a and J_{a+1} cannot generate an interference higher than $C^m + I_{\omega_{b+1}}(t - t_{a+1})$. Being such sequences valid job sequences following J_a (released at speed ω_a), it must also be

$$\forall t \geq 0,\ C^m + I_{\omega_{b+1}}(t - t_{a+1}) \leq I_{\omega_a}(t).$$

Consider now the sequence s and let $t_{b+1} = \widetilde{T}(\omega_b, \omega_{b+1})$ be the inter-arrival time between jobs J_b and J_{b+1}. Similarly as argued above, the interference $I^{(s)}(t)$ generated by s can be bounded as $\forall t \geq 0,\ I^{(s)}(t) \leq C^m + I_{w_{b+1}}(t - t_{b+1})$.

Being $\omega_a \geq \omega_b$, due to the monotonicity property of inter-arrival times (see Sections 2.3 and 7.1), it follows that $t_{a+1} \leq t_{b+1}$, which implies

$$\forall t \geq 0,\ I_{w_{b+1}}(t - t_{a+1}) \geq I_{w_{b+1}}(t - t_{b+1}).$$

Therefore, $\forall t \geq 0$,

$$I_{\omega_a}(t) \geq C^m + I_{\omega_{b+1}}(t - t_{a+1}) \geq C^m + I_{w_{b+1}}(t - t_{b+1}) \geq I^{(s)}(t).$$

Hence the lemma follows. □

Lemma 1 can be used to derive the following key theorem that expresses a dominance relationship between two interference envelopes.

Theorem 2. *Let J_a and J_b be two jobs released in mode m, and let ω_a and ω_b be the instantaneous engine speeds at their respective release times. If $\omega_a \geq \omega_b$ and $\forall n \in \mathbb{N}_{\geq 0},\ \mathcal{C}(\Omega^n(\omega_a, \alpha^-)) = \mathcal{C}(\Omega^n(\omega_b, \alpha^-))$, then $\forall t \geq 0,\ I_{\omega_a}(t) \geq I_{\omega_b}(t)$.*

Proof. The proof is trivial for $\omega_a = \omega_b$, therefore in the following we assume $\omega_a > \omega_b$. The strategy consists in demonstrating that, for *any* valid job sequence s that starts with J_b, there exists a valid sequence s' starting with J_a whose interference dominates the one of s. We proceed by constructing an inductive argument on the number of jobs $n \geq 1$ after $J_b \in s$.

Base case ($n = 1$). Since $\omega_a \geq \omega_b$, both J_a and J_b are released in the same mode m and $\mathcal{C}(\Omega(\omega_a, \alpha^-)) = \mathcal{C}(\Omega(\omega_b, \alpha^-))$. Hence, by Theorem 1 the dominance holds for job sequences that include a single job ($n = 1$) after J_b.

Inductive step ($n > 1$). Suppose that the theorem holds for all possible job sequences starting with J_b except those that include *more than* n jobs after J_b and $\forall k = 0, \ldots, n,\ \ \omega_{b+k} < \Omega^k(\omega_a, \alpha^-)$, where ω_{b+k} is the speed at the release of J_{b+k}. Let $s = J_b, \ldots, J_{b+n}, J_{b+n+1}$ be one of such job sequences.

Consider also the job sequence $s' = J_a, \ldots, J_{a+n}$ where each job J_{a+k} is released at speed $\Omega^k(\omega_a, \alpha^-)$. Let t_{a+n} and t_{b+n} be the release times of J_{a+n} and J_{b+n}, respectively. Being all jobs in s released at lower speeds than the ones in s', due to the monotonicity property of inter-arrival times it must be $t_{b+n} > t_{a+n}$.

Since function $\mathcal{C}(\omega)$ is non-decreasing and $\forall n \in \mathbb{N}_{\geq 0},\ \mathcal{C}(\Omega^n(\omega_a, \alpha^-)) = \mathcal{C}(\Omega^n(\omega_b, \alpha^-))$ (by hypothesis), every pair of jobs J_{a+k} and J_{b+k}, with $k = 0, \ldots, n$, is released in the same mode. As a consequence, the computational request accumulated up to times $t_{a+n} - \epsilon$ and $t_{b+n} - \epsilon$ by sequences s and s' is the same, say $\Pi > 0$ (with $\epsilon > 0$ arbitrary small). That is, $I^{(s)}(t_{b+n} - \epsilon) = I^{(s')}(t_{a+n} - \epsilon) = \Pi$.

For $t \geq t_{b+n}$, the interference $I^{(s)}(t)$ generated by s can be upper-bounded by exploiting the single-job interference of J_{b+n} as follows

$$\forall t \geq t_{b+n},\ I^{(s)}(t) \leq \Pi + i_{\omega_{b+n}}(t - t_{b+n}).$$

Now, since $\mathcal{C}(\Omega^{n+1}(\omega_a, \alpha^-)) = \mathcal{C}(\Omega^{n+1}(\omega_b, \alpha^-))$ (by hypothesis), then also $\mathcal{C}(\Omega(\omega_{a+n}, \alpha^-)) = \mathcal{C}(\Omega(\omega_{b+n}, \alpha^-))$. Therefore, Theorem 1 can be applied, which allows concluding that $\forall \delta \geq 0, i_{\omega_{a+n}}(\delta) \geq i_{\omega_{b+n}}(\delta)$. Thanks to this result, it is now possible to bound $I^{(s)}(t)$ as follows:

$$\begin{aligned}\forall t \geq t_{b+n}, \quad & I^{(s)}(t) \leq \Pi + i_{\omega_{b+n}}(t - t_{b+n}) \leq \\ & \leq \Pi + i_{\omega_{a+n}}(t - t_{b+n}) \leq \Pi + i_{\omega_{a+n}}(t - t_{a+n}).\end{aligned}$$

Since $\Pi + i_{\omega_{a+n}}(t - t_{a+n})$ copes with the interference generated by the $(n+1)^{th}$ job after J_a, and J_a is released at speed ω_a, it must also be that

$$\forall t \geq t_{a+n},\ \Pi + i_{\omega_{a+n}}(t - t_{a+n}) \leq I_{\omega_a}(t).$$

As a consequence, being $t_{a+n} < t_{b+n}$, we can finally conclude that also $\forall t \geq t_{b+n},\ I^{(s)}(t) \leq I_{\omega_a}(t)$ holds.

By Lemma 1, the interference generated by all the possible job sequences with J_{b+n+1} released at speed $\omega_{b+n+1} \geq \Omega^n(\omega_a, \alpha^-)$ (possible only if $\Omega(\omega_{b+n+1}, \alpha^+) \geq \Omega^n(\omega_a, \alpha^-)$) are dominated by $I_{\omega_{a+n}}(t)$. Therefore, the theorem holds for all the possible job sequences starting with J_b except those that include *more than* $n + 1$ jobs after J_b with $\forall k = 0, \ldots, n+1,\ \ \omega_{b+k} < \Omega^k(\omega_a, \alpha^-)$. Hence, the induction has to proceed only for job sequences s under the latter conditions.

Having shown that the interference generated by sequences s—with an arbitrary number n of jobs after J_b—is dominated by the interference envelope $I_{\omega_a}(t)$, the theorem follows. □

When exploring the search tree, jobs are generally released at different times, which is not the case considered in the hypothesis of the theorem above. A simple, but useful corollary of Theorem 2 can be derived to cope with this scenario.

Corollary 1. Theorem 2 also holds if J_b is released *later* than J_a.

Proof. Let t_a and t_b be the respective release times of J_a and J_b. Without loss of generality assume $t_a = 0$. If Theorem 2 holds, then $\forall t \geq 0, I_{\omega_a}(t) \geq I_{\omega_b}(t)$. Therefore, if $t_b > t_a = 0$, then also $\forall t \geq 0, I_{\omega_a}(t) \geq I_{\omega_b}(t - t_b)$ holds. □

4.3 Dominant speeds and critical job sequences

Theorem 2 allows solving the search problem discussed in Section 3 by exploring a limited set of job sequences. During the search for the maximum interference, for every pair of jobs J_a and J_b that satisfies the conditions of Theorem 2, job J_b can be discarded in favor of J_a. By extensively applying this reasoning, it can be concluded that the maximum interference generated by an AVR task can be computed by only taking into account a *limited* set of engine speeds, which will be referred to as *dominant speeds*. The notion of dominant speeds allows computing the exact interference with a contained complexity and avoiding quantization.

Several approaches can be adopted to compute such dominant speeds: indeed, a super set of dominant speeds can easily be computed, e.g., by manually checking the conditions of Theorem 2 with a binary search. A more accurate technique for computing dominant speeds is presented in Section 4.4.

Using the notion of dominant speeds, it is also possible to define a *critical* job sequence for an AVR task.

Definition 5. A *critical* job sequence for an AVR task τ^* is a job sequence where each job is released at a dominant speed.

The main property of the critical job sequences is that for each non-critical job sequence s' there exists a critical job sequence s whose interference dominates the one of s'. Formally, if $\mathcal{CS}(t)$ is the set of the possible critical job sequences in the time window $[0, t]$, and $S(t)$ is the set of all possible valid job sequences in the same internval, then

$$\begin{aligned}&\forall s' \in \{\mathcal{S}(t) \setminus \mathcal{CS}(t)\} \\ &\qquad \exists s \in \mathcal{CS}(t) \mid \quad \forall t' \in [0, t]\ I^{(s)}(t') \geq I^{(s')}(t'). \qquad (8)\end{aligned}$$

Based on this result, the worst-case interference caused by an AVR task τ^* is generated from a sequence of jobs belonging to one of the critical sequences.

4.4 Computing the dominant speeds

The dominant speeds in a given range $[\omega_b, \omega_a]$ can be determined by exploiting 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 interference envelopes $I_\omega(t)$ for speeds $\omega \in (\omega^*, \omega_a]$ are dominated by $I_{\omega_a}(t)$.

Note that speed ω^* can be readily 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 $\mathcal{C}(\omega_a^{(n)}) \neq \mathcal{C}(\omega^{*,(n)})$, corresponding to the speed for which the theorem hypothesis are violated. Since function $\mathcal{C}(\omega)$ is non-increasing, such a speed $\omega^{*,(n)}$ can always be found[1]. Also speed $\omega^{*,(n)}$ must 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 $\omega_C^{(n)}$. Speed $\omega_C^{(n)}$ can then be easily computed by using the inverted physical equation of $\Omega^n(\omega, \alpha^-)$, that is $\omega_C^{(n)} = \Omega^{-n}(\omega^{*,(n)}, \alpha^-)$. Finally, since the conditions of Theorem 2 must hold $\forall n \in \mathbb{N}_{\geq 0}$, speed ω^* is computed as the maximum of all such candidates, that is, $\omega^* = \max_{n\in\mathbb{N}_{\geq 0}}\{\omega_C^{(n)}\}$. 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 $[\omega^-, \omega^+]$, the maximum value for n is bounded to $\max\{n \in \mathbb{N}_{\geq 0} \mid \Omega^n(\omega_a, \alpha^-) \geq \omega^-\}$.

The technique for computing the dominant speeds is summarized in the algorithm reported in Figure 8.

```
1:  procedure GETDOMINANTS($\omega_b, \omega_a$)
2:      $\omega^* \leftarrow \omega_a$;
3:      while $\omega^* > \omega_b$ do
4:          DOMINANTS.ADD($\omega^*$);
5:          maxN $\leftarrow \max\{n \in \mathbb{N}_{\geq 0} \mid \Omega^n(\omega^*, \alpha^-) \geq \omega^-\}$;
6:          for $i = 0$ to maxN do
7:              $\omega^{(i)} = \Omega^i(\omega^*, \alpha^-)$;
8:              $\omega^{*,(i)} = \max_{m=1,\dots,M}\{\omega^m < \omega^i\}$;
9:              $\omega_C^{(i)} = \Omega^{-i}(\omega^{*,(i)}, \alpha^-)$;
10:         end for
11:         $\omega^* = \max_{i=0,\dots,maxN}\{\omega_C^{(i)}\}$;
12:     end while
13:     return DOMINANTS;
14: end procedure
```

Figure 8. Algorithm for computing the dominant speeds in a generic speed range $[\omega_b, \omega_a]$.

An example of application of the algorithm GETDOMINANTS is illustrated in Figure 9.

4.5 Additional pruning

By leveraging another simple observation, it is possible to further reduce the jobs sequences that have to be explored to compute the maximum interference.

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 $\mathcal{C}(\omega_a^{(n)}) = \mathcal{C}\left(\omega^{*,(n)}\right)$ of Theorem 2. In other words, it is not possible to have a mode change decelerating from speed $\omega_a^{(n)}$.

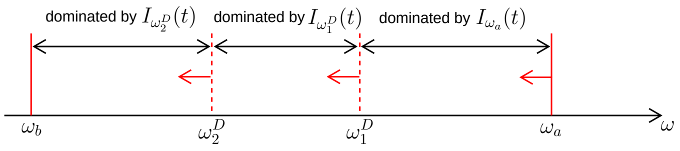

dominated by $I_{\omega_2^D}(t)$
dominated by $I_{\omega_1^D}(t)$
dominated by $I_{\omega_a}(t)$
ω_b
ω_2^D
ω_1^D
ω_a
ω

Figure 9. Example of the result produced by algorithm GETDOMINANTS when applied to a range of speeds $[\omega_b, \omega_a]$. The algorithm produces three dominant speeds: ω_a, ω_1^D and ω_2^D. The interference envelope $I_{\omega_a}(t)$ dominates all the interference envelopes $I_\omega(t)$ for speeds $\omega \in (\omega_1^D, \omega_a]$. Similarly, $I_{\omega_1^D}(t)$ dominates the ones for speeds $\omega \in (\omega_2^D, \omega_1^D]$ while $I_{\omega_2^D}(t)$ dominates the ones for speeds $\omega \in [\omega_b, \omega_2^D]$.

Consider two jobs J_a and J_b simultaneously released in the same mode m and at *dominant* speeds ω_a and ω_b, respectively, with $\omega_a \geq \omega_b$. If $\Omega(\omega_a, \alpha^-) < \Omega(\omega_b, \alpha^+)$, i.e., the interval of possible speeds at the release of the next jobs J_{a+1} and J_{b+1} are overlapped, then it may exist a dominant speed $\omega^* \in [\Omega(\omega_a, \alpha^-), \Omega(\omega_b, \alpha^+)]$ that is reachable by both ω_a and ω_b after *one* angular period of the AVR task. A procedure that computes the interference by only relying on dominant speeds would consider dominant speed ω^* at least two times, both as a follower of ω_a and ω_b.

However, the maximum interference can be computed (without loosing accuracy) by considering the dominant speed ω^* only as a follower of ω_a. Let J_{a+1} and J_{b+1} be the jobs following J_a and J_b, respectively, released at speed ω^*. Due to the physical nature of the inter-arrival times between jobs, being $\omega_a \geq \omega_b$, job J_{a+1} will be released *before* job J_{b+1}. Since the same system state (speed ω^*) can be reached *earlier* by J_{a+1}, and the computational demand of jobs J_a and J_b is the same (as they are both released in the same mode), the job sequences following J_{b+1} can be *discarded* in favor of the exploration of the job sequences following J_{a+1}.

In other words, any job sequence following J_{b+1} will never generate an interference higher than the maximum interference generated by the job sequences following J_{a+1}.

Since, in the considered job sequences, the dominant speed ω^* is reachable as an immediate follower of both ω_a and ω_b, and $\omega_a \geq \omega_b$, then it follows that $\omega^* \geq \Omega(\omega_a, \alpha^-)$. Therefore, this additional pruning condition follows directly from Lemma 1.

5 Exact Response-Time Analysis

This section derives the response time analysis for a set of AVR and periodic/sporadic tasks with constrained deadlines (both angular and temporal). The analysis first considers the case of a single AVR task interfering with a periodic task set and then addresses the dual case in which a periodic task set creates interference on a single AVR task. The extension to multiple AVR tasks activated by the same rotation source is considered in Section 5.1.2 and Section 5.3. This extension does not consider AVR tasks with different angular periods and phases: their analysis determines several complications and it is left as future work.

The proposed method builds upon standard response-time analysis [1] for fixed-priority scheduling, which aims at computing the length of the longest busy-period for a task τ_i (i.e., an interval of time without idle times where only τ_i and its higher-priority tasks execute).

5.1 Response time of a periodic task interfered by an AVR task

Let τ_i be a periodic task suffering interference from a set of regular periodic tasks and a single AVR task τ^*, all having higher priority than τ_i. In the following, the set of periodic tasks having higher priority than τ_i is denoted as $hp(i)$ and the set of critical job sequences in the interval $[0, D_i]$ is denoted as $\mathcal{CS}(D_i)$, or simply as $\mathcal{CS}$.

For a particular job sequence s of τ^*, the response time of τ_i, denoted as $R_i^{(s)}$, can be expressed as

$$R_i^{(s)} = \min_{t \geq 0} \left\{ C_i + I^{(s)}(t) + \sum_{\tau_j \in hp(\tau_i)} \left\lceil \frac{t}{T_j} \right\rceil C_j = t \right\}. \quad (9)$$

Note that the response time $R_i^{(s)}$ may tend to infinity under overload conditions. Here, to simplify the presentation, its computation is derived by assuming that the $R_i^{(s)}$ is implicitly bounded by $D_i + 1$ for any job sequence s. It is worth observing that, for the purpose of schedulability, this assumption does not impact the analysis.

The challenging part in the analysis is the computation of the interference $I^{(s)}(t)$ imposed by the AVR task for all the possible job sequences s. The following theorem formalizes that the response time of τ_i can be computed by only considering the critical job sequences in the interval $[0, D_i]$.

Theorem 3. *The response time R_i of a periodic task τ_i interfered by an AVR task τ^* is the maximum response time over all possible critical job sequences generated by τ^*, that is*

$$R_i = \max_{s \in \mathcal{CS}} R_i^{(s)}. \quad (10)$$

Proof. By contradiction, suppose that there exists a non-critical job sequence $s' \notin \mathcal{CS}$ such that $R_i^{(s')} > R_i$. Then, to avoid the completion of τ_i before (or at) time R_i, it must be that at time R_i, the sequence s' caused more interference to τ_i than when interfered by critical sequences, i.e., $\forall s \in \mathcal{CS}$, $I^{(s')}(R_i^{(s)}) > I^{(s)}(R_i^{(s)})$. Hence, for each critical job sequence s, there exists a time instant at which $I^{(s)}$ is dominated by the interference of the non-critical job sequence s'. This contradicts the main property of critical job sequences expressed by Equation (8), hence the theorem follows. $\square$

Before proceeding, it is worth noting that the use of the *maximum* interference $I(t) = \max_{\omega_0} I_{\omega_0}(t)$ that an AVR task can generate in a given time window of length t prevents the derivation of the exact response time, as highlighted by Stigge and Yi [21], [22] in the context of the digraph real-time task model. In fact, the speed sequences that generate the critical job sequences $s \in \mathcal{CS}$ (that contribute to $I(t)$) may be mutually exclusive. Thus, simply computing their maximum interference may lead to a sequence of speeds that cannot occur in practice. In other words, by computing the interference envelope we lose the information about the sequence of speeds that may generate the envelope.

To better clarify this point, which at a first look may appear counter-intuitive, consider the case where the response time of a periodic task τ_i is computed when it is interfered by an high-priority AVR task τ^*. Also, assume that the set $\mathcal{CS}$ is composed of only two sequences, i.e., $\mathcal{CS} = \{s_a, s_b\}$. Figure 10 shows the two interferences $I^{(s_a)}(t)$ and $I^{(s_b)}(t)$ originated by the two job sequences s_a and s_b, respectively, and the corresponding interference envelope $I(t)$, computed as the maximum between $I^{(s_a)}(t)$ and $I^{(s_b)}(t)$. As clearly visible from the plots, the envelope $I(t)$ leads to a response time $\overline{R}$ much higher than those resulting from the two concrete interferences (note that in the graph the response time is the time instant at which the computational demand matches the processor supply). In other words, the concrete job sequences of τ^* contributing to $I(t)$ (the *actual* ones that τ^* can generate), lead to a response time smaller than $\overline{R}$.

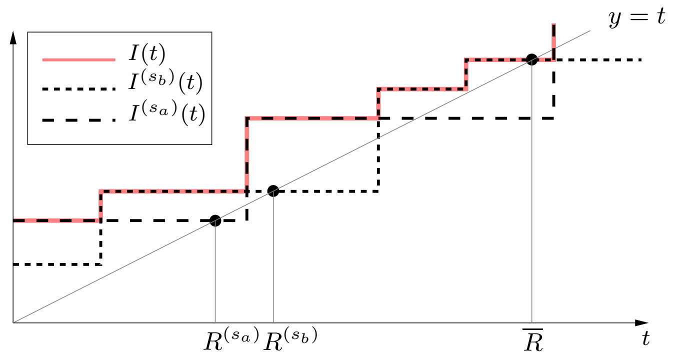


Figure 10. Example with two job sequences s_a and s_b in which the interference envelope (continuous line) leads to an over-estimated response time $\overline{R}$. The exact response time is $R^{(s_b)}$, which is given by job sequence s_b.

In general, since the determination of the time at which the processor is idle cannot be computed without a full knowledge of the tasks involved in the busy-period of τ_i, it is not possible to compute a priori the interference of an AVR task. In other words, the maximum response time R_i is originated from different sequences $s \in \mathcal{CS}$ depending on the interference of the higher priority tasks. Hence, contrary to classical periodic tasks, to characterize the exact response time it is not possible to abstract the interference of τ^* by using a single value of interference for each time instant t. To address this issue, the solution proposed in this paper consists of computing the interference *on the fly* while the response time of τ_i is computed, exploring the domain of the $\mathcal{CS}$ set.

5.1.1 *Algorithm for Computing the Response Time*

According to Theorem 3, to compute the response time R_i, it is necessary to identify all possible critical job sequences $s \in \mathcal{CS}$ and then compute the response time $R_i^{(s)}$ for each sequence s.

This section presents an algorithm to efficiently compute R_i by evaluating the critical job sequences *on-the-fly and only when needed*, providing additional pruning in the search of the speed space and significant speedup for the analysis. To implement the pruning, the algorithm leverages both Theorem 2, which serves to identify dominant speeds, and the additional pruning method discussed in Section 4.5, which allows discarding some dominant speeds in particular scenarios.

The proposed algorithm (reported in Figure 12) visits the speed tree with pruning using the concept of dominant speeds while discarding the job sequences that would result in an idle time (for priority level i) earlier than one of the candidate response times. The algorithm operates recursively for increasing time values. At any point in time, the main function of the algorithm, `RespTime`, computes the contribution to the interference of one additional job activation. `RespTime` is called by passing

- the priority index i of the task for which the response time is computed (used to evaluate the contribution to the interference from the set of periodic tasks);
- the current speed ω (at the time the job of τ^* is activated);
- the current time t;
- the execution time requests Π (the contribution to the interference of τ^*) accumulated up to time t; and
- a set $\mathcal{E}$ of dominant speeds that have *not* to be considered (according to the pruning conditions discussed in Section 4.5).

Note that the algorithm always terminates when the current time t exceeds the deadline of τ_i.

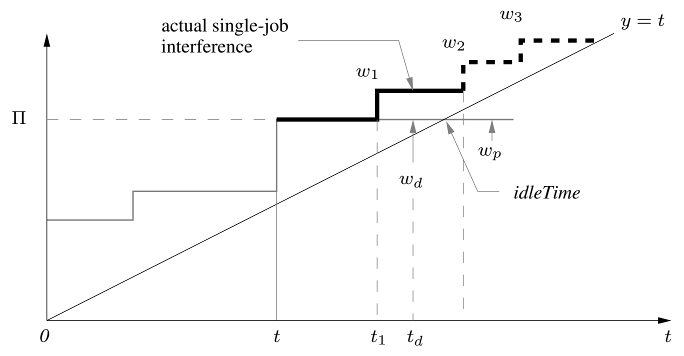


Figure 11. Pruning and branching in the algorithm for computing the response time. The figure considers a job J of an AVR task released at time t. The thick line indicates the single-job interference for J: the solid part considers jobs following J that are released *before* the idle time, while the dashed part refers to the jobs that would be released *after* the idle time, hence not contributing to the exact response time.

Each time `RespTime` is called, ω *is one of the dominant speeds* and t the candidate point in time for the activation of a job released at speed ω.

First, given t, ω and Π, the algorithm computes the next possible idle time (an example is illustrated in Figure 11), which corresponds to the tentative response time of τ_i as if no other jobs of the AVR task would be released. This can be computed via a standard fixed-point iteration starting from time t and accounting for Π units of interference generated by the AVR task. Such a tentative response time is also stored as candidate for the maximum response time of τ_i (line 7).

Subsequently, the procedure GETDOMINANTS is used for computing the dominant speeds in the range of possible instantaneous speed at the activation of the *next* job, that is $[\Omega(\omega, \alpha^-), \Omega(\omega, \alpha^+)]$ (line 9). Some of such dominants speeds can be discarded by looking at the set $\mathcal{E}$ (line 10). The overall set of dominant speeds that have to be considered is stored into $\mathcal{D}$.

Then, the algorithm proceeds by iterating over the dominant speeds into the set $\mathcal{D}$ in descending order (line 5), each of these representing a possible branch explored by the algorithm. For each dominant speed $\omega^{next} \in \mathcal{D}$, the algorithm computes the time T^{next} after which the next job can be activated at speed ω^{next} (line 17). If such a job can be activated *after* the idle time, then the corresponding branch is discarded (such as the one released at speed ω_p in Figure 11). Otherwise, if the job can be activated *before* the idle time (such as the one released at speed ω_d in Figure 11), then it is considered for the next recursive call.

To leverage the pruning conditions discussed in Section 4.5, the algorithm keeps track of the dominant speeds explored by the immediate following recursive branches. To this end, each recursive call of the algorithm returns the set of dominant speeds that have been considered; such speeds are then collected by the parent branch into the set $\mathcal{E}^{next}$ (lines 20-21). Since dominant speeds are explored in descending order, the conditions of Lemma 1 can only be violated when the next job is released in a mode different than the one previously considered for filling the set $\mathcal{E}^{next}$. In this case, the algorithm invalidates the set $\mathcal{E}^{next}$ (line 14), thus not enforcing the pruning conditions discussed in Section 4.5.

Figure 13 shows the algorithm to check the schedulability of τ_i using the procedure `RespTime`. Procedure `SchedulabilityTest` starts by computing the initial dominant speeds in the full range allowed for engine speeds, (i.e., $[\omega^-, \omega^+]$). Then, for each initial dominant speed ω_0, the response time candidates are computed. Since all the candidates represent response-time values related to possible job sequences starting from ω_0, the maximum R of such candidates is taken as response time for speed ω_0. If the response time R exceeds the deadline D_i, then τ_i is not schedulable; otherwise, if R results lower (or equal) than D_i for each initial dominant speeds, then τ_i is schedulable.

5.1.2 Interference from Multiple AVR Tasks

This section extends the analysis by considering the interference from multiple AVR tasks triggered by the same rotating source (which is a relevant case in engine-control applications [15]).

When computing the response time of a periodic task τ_i interfered by a set of AVR tasks $hp^A(\tau_i) = \{\tau_0^*, \tau_1^*, \dots \tau_p^*\}$ that have the same angular period and phase (assumed as 0 for convenience), the interference from the tasks in the AVR set is equivalent to the interference from a single task τ_k^*, as their release times are always implicitly synchronized.

Task τ_k^* is constructed as follows. Consider the union of the mode speeds of the AVR tasks and sort them from ω^+ to ω^-. The cardinality of the set gives the number of modes for τ_k^* (at most the sum of the number of modes for all the AVR tasks). Each mode m of τ_k^* is defined by the corresponding speed range $(\omega_k^{m+1}, \omega_k^m]$ and a worst case execution time $\mathcal{C}_k^m(\omega) = \sum_{\tau_j^* \in hp^A(\tau_i)} \mathcal{C}_j(\omega_k^m)$. At this point, the approach presented in the previous section can be applied.

Note that, in the presence of AVR tasks with different angular periods or different angular phases, their release times are not anymore synchronized, hence the proposed approach does not work. Nevertheless, their behavior is not totally independent, as they are still triggered by the same rotation source. The consideration of such task sets requires

```
global set RTCandidates
procedure RESPTIME($i, \omega, \Pi, t, \mathcal{E}$)
    if $t > D_i$ then return $\emptyset$;
    end if

    idleTime $\leftarrow$ GETIDLETIME($i, \omega, \Pi, t$);
    RTCandidates.ADD(idleTime);

    $\mathcal{D} \leftarrow$ GETDOMINANTS($\Omega(\omega, \alpha^-), \Omega(\omega, \alpha^+)$);
    $\mathcal{D} \leftarrow \mathcal{D} \setminus \mathcal{E}$;

    $C^{prev} = 0$;
    for each $\omega^{next} \in \mathcal{D}$ in descending order do
        if $\mathcal{C}(\omega^{next}) \neq C^{prev}$ then
            $\mathcal{E}^{next} = \emptyset$;
        end if
        $T^{next} \leftarrow \widetilde{T}(\omega, \omega^{next})$;
        if $t + T^{next} <$ idleTime then
            $\Pi^{next} \leftarrow \Pi + \mathcal{C}(\omega^{next})$;
            $\mathcal{D}^{next} \leftarrow$ RESPTIME($i, \omega^{next}, \Pi^{next}, t + T^{next}, \mathcal{E}^{next}$);

            $\mathcal{E}^{next} = \mathcal{E}^{next} \cup \mathcal{D}^{next}$;
            $C^{prev} = \mathcal{C}(\omega^{next})$;
        end if
    end for
    return $\mathcal{D}$;
end procedure
```

Figure 12. Procedure for computing the response time of a task τ_i interfered by an AVR task.

```
global set RTCandidates
procedure SCHEDULABILITYTEST($i$)
    dominants $\leftarrow$ GETDOMINANTS($\omega^-, \omega^+$);
    MAXTIME $\leftarrow D_i$;
    for $\omega_0$ in dominants do
        RTCandidates $\leftarrow \emptyset$;
        RESPTIME($i, \omega_0, 0, 0, \emptyset$);
        $R \leftarrow$ MAX(RTCandidates);

        if $R > D_i$ then
            return UNSCHEDULABLE;
        end if
    end for
    return SCHEDULABLE;
end procedure
```

Figure 13. Procedure describing the schedulability test for a task τ_i.

new theoretical foundations, as the response-time algorithm would have to take into account *multiple* search problems in the speed domain that are coupled by the same evolution of the engine speed over time. For this reason, this extension is left as future work.

5.2 Response time of an AVR task interfered by periodic tasks

Let us now consider the response time of an AVR task τ^* interfered by periodic tasks (assuming there are periodic tasks having higher priority than τ^*).

Since the response time of τ^* depends on the instantaneous speed ω_0 at which it is released, we have

$$R(\omega_0) = \min_{t \geq 0} \left\{ \mathcal{C}(\omega_0) + \sum_{\tau_i \in hp(\tau^*)} \left\lceil \frac{t}{T_i} \right\rceil C_i = t \right\},$$

where $hp(\tau^*)$ denotes the set of periodic tasks having higher priority than τ^*.

Note that the dependency on the speed ω_0 can be removed by considering each mode of τ^*, so obtaining a response-time value for each mode $m = 1, \ldots, M$, that is

$$R^m = \min_{t \geq 0} \left\{ C^m + \sum_{\tau_i \in hp(\tau^*)} \left\lceil \frac{t}{T_i} \right\rceil C_i = t \right\}.$$

Finally, the schedulability of τ^* can be checked by verifying that $R^m \leq D(\omega^m)$ for each mode $m = 1, \ldots, M$, where $D(\omega^m)$ is the shortest temporal deadline of τ^* in mode m, computed by Equation (2) in the special case of $\alpha_k = \alpha^+$ and $\Theta_i = \Delta_i$, so obtaining $D(\omega) = (\sqrt{\omega^2 + 2\Delta_i \alpha^+} - \omega)/\alpha^+$.

5.3 Response time of an AVR task interfered by other AVR tasks

This section addresses the schedulability of an AVR task τ_i^* interfered by both periodic tasks and multiple AVR tasks that have the same angular period of τ_i^*. When multiple AVR tasks with the same angular period are considered, only one job for each high-priority AVR task can produce interference, i.e., a single computation time must be accounted for. The set of AVR tasks having higher priority than τ_i^* is denoted as $hp^A(\tau_i^*)$.

As done in Section 5.2, the dependency from the speed $\omega_0 \in [\omega^-, \omega^+]$ can be removed by considering each mode m of τ_i^* and computing the reponse time for each m. Once the mode of τ_i^* is selected, each higher priority AVR task $\tau_j^* \in hp^A(\tau_i^*)$ may be in a finite set of modes $m_{j,k}, \ldots, m_{j,n}$ such that the intersection of the speed ranges for m and any of the $m_{j,p}$ with $k \leq p \leq n$ is not empty (as shown in Figure 14). The possible interference of each higher priority AVR task τ_j^* only changes at the boundary speeds of its modes. Hence, these are the only (finite) number of speeds that need to be considered.

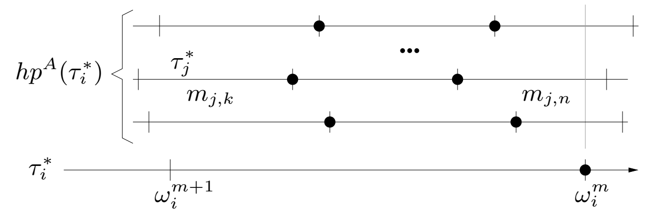


Figure 14. Identifying the contributions of the higher priority AVR task modes to τ_i^*. The figure considers a mode m for an AVR task τ_i^* and shows the mode of the high-priority AVR task (set $hp^A(\tau_i^*)$) that overlap with the corresponding speed range $(\omega_i^{m+1}, \omega_i^m]$. The black dots indicate the switching speeds.

Formally, to guarantee the schedulability of τ_i^*, the following conditions must be satisfied: for each mode m of τ_i^*, $\forall \tau_j^* \in hp^A(\tau_i^*) \cup \{\tau_i\}$

$$\forall \omega_j^{m'} \in (\omega_i^{m+1}, \omega_i^m] \quad R_i^m(\omega_j^{m'}) \leq D_i(\omega_j^{m'}),$$

where

$$R_i^m(\omega) = \min_{t \geq 0} \left\{ C_i^m + \sum_{\tau_j^* \in hp^A(\tau_i^*)} \mathcal{C}_j(\omega) + \sum_{\tau_j \in hp(\tau_i^*)} \left\lceil \frac{t}{T_j} \right\rceil C_j = t \right\}.$$

6 Experimental Results

This section presents a set of experimental results aimed at comparing the exact schedulability analysis presented in this paper with the sufficient ILP-based analysis for AVR tasks presented by Davis et al. in [11]. Both schedulability tests have been implemented and applied to synthetic workloads for comparison. The ILP-based formulation has been implemented using the IBM CPLEX solver, whereas the proposed algorithm has been implemented in C++. To ensure a broader comparison, the proposed schedulability test has also been compared against two other tests proposed in [11] and the utilization bound for EDF scheduling proposed in [3], [9].

Since the ILP-based analysis of Davis et al. requires a quantization on the speed domain, a step of 100 RPM was adopted, as suggested by the authors. Our analysis discriminates 1 RPM in the computation of the dominant speeds. It is also worth noting that the approach presented in [11] considers a slightly different task model in which mode-change is triggered as a function of an estimation of the instantaneous speed through the average speed in the previous inter-arrival time. The ILP formulation [11] leads to some inconsistencies in the computation of the interference for low speed values: the problem has been fixed by the authors in a technical report [12], taken as a reference for our comparison.

In the experiments, the rotation source is assumed to range from $\omega^- = 500$ RPM to $\omega^+ = 6500$ RPM, which are typical values for a production car engine. As done by Davis et al. [11], the values for the acceleration have been selected 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.

6.1 Workload generation

The experiments have been performed on a task set consisting of n periodic tasks and an AVR task. Given an overall target utilization U^P for the set of periodic tasks, each periodic task is generated as follows:

- The utilization U_i of each task τ_i is randomly generated using the UUniFast [2] algorithm such that $\sum_{i=1}^{n} U_i = U^P$. The minimum utilization of each periodic task is enforced to $U^{min} = 0.005$;
- Task periods T_i are randomly generated (with a uniform distribution) in the range $[3, 100]$ msec;
- Deadlines for periodic tasks are implicit, i.e., $D_i = T_i$;
- Execution times are computed as $C_i = U_i T_i$.

Observe that the case of multiple AVR tasks with a common activation source can be modeled as a single AVR task (also called *representative AVR*) as discussed in Section 5.1.2.

Given a target utilization U^* for the representative AVR task, its parameters are generated as follows:

- The angular period is $\Theta = 2\pi$ (causing a task activation for each engine revolution);
- The angular deadline is implicit, i.e., $\Delta = \Theta$;
- The number of modes M is randomly generated in the range $[M^{min}, M^{max}]$. The values defining the range are parameters for the definition of the experimental setup;
- A random mode m' is selected to have the maximum utilization $U^{m'} = U^*$. The utilization U^m of the other modes $m \neq m'$ is randomly generated in the range $[0.85U^*, U^*]$;
- 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^+. Once the boundary speeds for the mode transitions are generated, they are checked to ensure a minimum separation between any two values. If the minimum separation between any two speeds 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 defined as $C^m = U^m \Theta / w^m$. If the generated computation times are not monotonically increasing with respect to modes, then they are discarded, and a new set is generated.

The overall utilization of the set of periodic and AVR tasks is $U = U^P + U^*$. Task priorities are assigned according to the Rate Monotonic order (i.e., the lower the period, the higher the priority), where the period of the AVR task is considered as $T^* = \Theta/\omega^+$, that is, its lowest possible inter-arrival time.

The approaches compared in the experiments are denoted as:

- EXACT - The analysis presented in Section 5;
- ILP - The analysis proposed in [11] using the revised ILP constraints of [12].
- VRB-L2 - The VRB-L2 test proposed in [11] (Eq. (7)).
- SPORADIC - The standard response-time analysis where AVR tasks is converted to sporadic tasks taking the maximum execution time and the minimum inter-arrival time (denoted as RTA-SP in [11]).
- EDF-U-BOUND - The utilization-based test for EDF proposed in [3], [9] (note that, under the experimental setting considered here, the bound proposed in [3] is the same as the one of [9]).

Note that the analysis presented in this paper does not apply to EDF scheduling, therefore the results for the EDF-U-BOUND test do not enable a fair comparison; rather, they should be intended as representative for the performance that can be obtained with dynamic-priority scheduling.

6.2 Experiment 1

The first experiment was carried out to measure the schedulability ratio of the two approaches as a function of the overall utilization U of task set composed of $n = 5$ periodic tasks and an AVR task with $M^{min} = 4$ and $M^{max} = 8$. The utilization of the AVR task was computed as $U^* = \rho_u U$. For each value of the utilization, the two schedulability tests were executed over 500 randomly generated task sets.

Figure 15(a) shows the results of this experiment when the utilization U varies from 0.3 to 0.95, and for $\rho_u = 0.4$. Clearly, both tests tend to degrade as the system load increases. In the range $[0.7, 0.95]$, the EXACT analysis improves the schedulability with respect to the ILP test, being

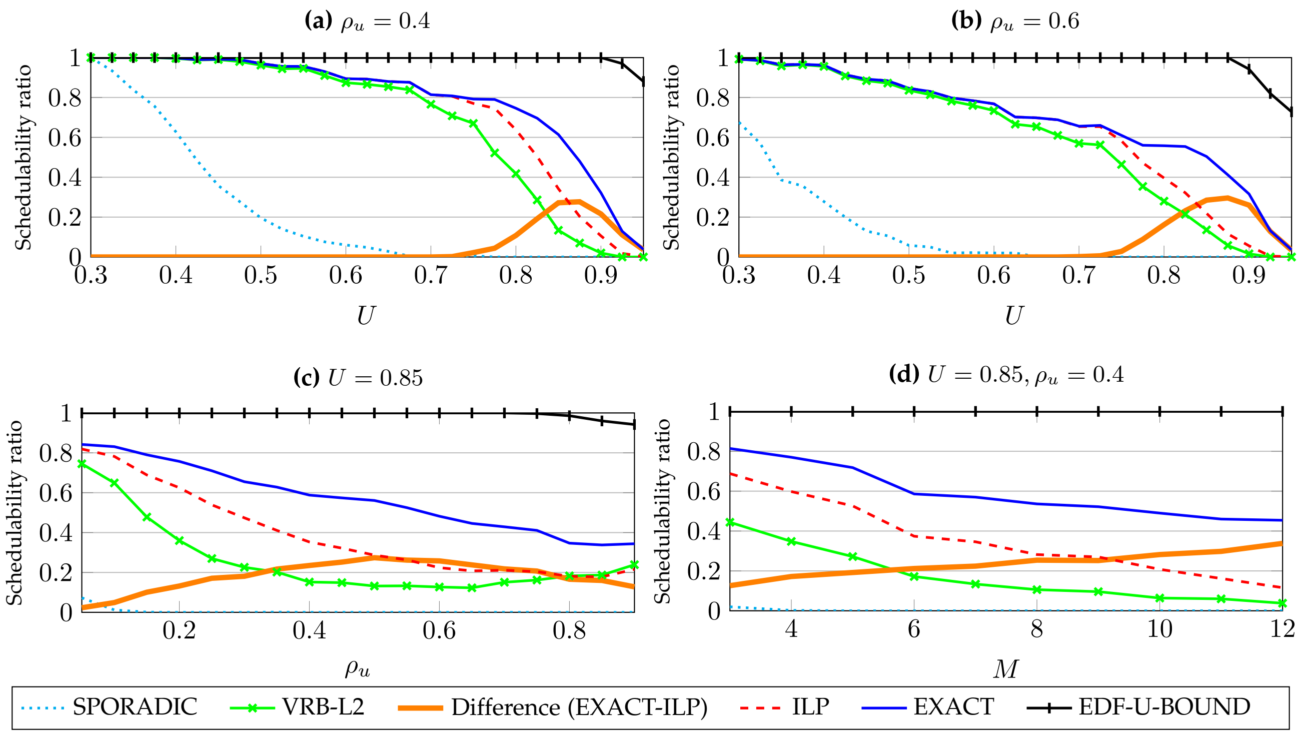


Figure 15. Schedulability ratio under four representative configurations as a function of the system utilization U (insets (a) and (b)), the relative utilization ρ_u of AVR tasks (inset (c)), and the number of modes M (inset (d)). The parameters of each configuration are reported in the captions above the graphs.

able to admit 6 times more task sets for $U = 0.9$. Figure 15(b) shows the results of a similar experiment carried out for $\rho_u = 0.6$, where the gain in schedulability of the EXACT test over ILP is 10 times more for $U = 0.9$. Both figures also report the difference of the two schedulability ratios to better appreciate the results.

Note that the achieved improvement of the proposed analysis exactly occurs in the workload range where these applications typical operate (80% utilization or higher). The performance gap with respect to the VBR-L2 and SPORADIC tests is not surprising and is in line with the results presented in [11]. The results also confirm the excellent performance of EDF scheduling, as it has been observed in [4].

6.3 Experiment 2

A second experiment was carried out to better evaluate the dependency of the schedulability ratio on the utilization of the AVR task by varying the factor $\rho_u = U^*/U$ from 0.05 to 0.9. For each value of ρ_u, the two schedulability tests (EXACT and ILP) were executed over 500 randomly generated task sets composed of $n = 5$ periodic tasks and an AVR task with $M^{min} = 4$ and $M^{max} = 8$.

The results of this experiment for $U = 0.85$ are reported in Figure 15(c). As visible from the plots, the gain in schedulability of the EXACT test increases with ρ_u, until reaching a significant improvement around $\rho_u = 0.5$. Notice that, for high values of the AVR utilization ($\rho_u \in [0.6, 0.85]$) the ILP analysis shows a saturation effect in the schedulability ratio, whereas the EXACT test is still able to admit 2 times more task sets than ILP. Finally, it is worth observing that, in the considered setting, the SPORADIC test is totally ineffective, while the performance gap between ILP and VRB-L2 tends to reduce as ρ_u increases.

6.4 Experiment 3

Another experiment has been done to measure the schedulability ratio of the two methods when the number M of modes of the AVR task varies from M^{min} to M^{max}. The overall utilization of the task set is fixed to $U = 0.85$, and the utilization of the AVR task is $U^* = 0.4U$. For each value of M, 500 task sets have been randomly generated including $n = 5$ periodic tasks.

Figure 15(d) shows the results of this experiment, when M is varied from 3 to 12. Note that both the tests decrease their performance as M increases, but the improvement of the EXACT test over ILP increases with the number of modes M or, equivalently, with the number of AVR tasks, as observed at the beginning of Section 6.1. Concerning the SPORADIC and VRB-L2 tests, the same observations made in the previous section also hold for this experiment.

6.5 Running times

Another set of experiments has been carried out to measure the running time of the schedulability test proposed in this paper. The implementation has been compiled with GCC 4.9.2 for Windows with all the optimizations enabled (-O3 flag). The tests have been executed on a machine equipped with a quad-core Intel i7 processor running at 3.5 GHz. The implementation is sequential (i.e., no parallelism has been exploited).

The results for four representative configurations are reported in Figure 16: the parameters of each configuration are reported in the captions above the graphs. The graphs

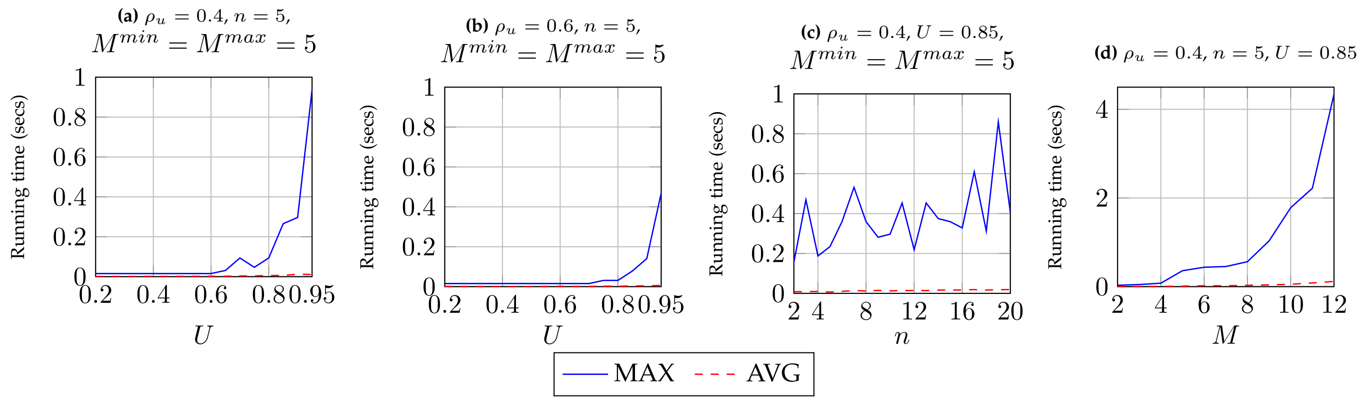


Figure 16. Average and maximum running times (in seconds) of the EXACT analysis for an entire task set. The results consider four representative configurations, which are reported in the captions above the graphs. Insets (a) and (b) refer to Experiment 1, inset (c) refers to Exper. 2 and inset (d) refers to Exper. 3.

show both maximum and average running times of the analysis (for an entire task set) as a function of the utilization U, the number of periodic tasks n and the number of modes M of the AVR task. Specifically, insets (a) and (b) refer to Experiment 1, inset (c) refers to Experiment 2, and inset (d) refers to Experiment 3. For each tested value of the parameter that has been varied, the schedulability test has been executed on 1000 randomly generated task sets.

As can be observed from the graphs, all the collected maximum running times are below 1 second for the configurations of insets (a), (b) and (c). The maximum running times increase up to 4 seconds only as a function of the number of modes (inset (d)): this happens because the number of dominant speeds increases with the number of modes, thus determining an increasing number of scenarios that have to be considered in the analysis. Finally, also note that the average running times are always in the order of a few milliseconds and far from the maximum values for all the four configurations.

Overall, this set of experiments clearly shows that the running time of the proposed analysis is perfectly compatible with the time-frame of off-line design activities. Further improvements in terms of speed-up can also be achieved with a carefully optimization of the implementation and/or by exploiting parallelism.

7 Generalization to Arbitrary Acceleration Functions

The assumption of constant acceleration, on which the model presented in Section 2.2 is based, can lead to the computation of optimistic (larger) inter-arrival times between two jobs of an AVR task, when using function $\widetilde{T}(\omega_k, \omega_{k+1})$. Given two jobs $J_{i,k}$ and $J_{i,k+1}$ of an AVR task τ_i^*, respectively released at speeds ω_k and ω_{k+1}, there can exist several *non-constant* acceleration profiles that, when undertaken by the rotation source, lead to lower inter-arrival times between $J_{i,k}$ and $J_{i,k+1}$ with respect to the one related to the case of constant acceleration.

To overcome this limitation, this section presents another model for the rotation source. Differently from the one presented in Section 2.2, this model does not rely on the assumption of constant acceleration, but it is based on limit-case acceleration profiles that allow deriving a conservative lower-bound on the inter-arrival time of AVR tasks. Such limit-case acceleration profiles are obtained by considering bounded acceleration but *unbounded jerk* (i.e., infinite rate of change for the acceleration), which allows coping with any possible acceleration profile that the rotation source can undertake.

Given an initial state $X_0 = [\theta = 0, \omega = \omega_a]$ and a final state $X_1 = [\theta = \Theta_i, \omega = \omega_b]$, it is known [8], [23] that, under bounded speed and bounded acceleration, the acceleration profile leading to the minimum time to reach state X_1 from state X_0 can be constructed as follows:

- accelerate with the maximum acceleration α^+ until reaching an intermediate speed ω_X, covering an angle Θ^+;
- decelerate with maximum deceleration α^- from ω_X, covering an angle $\Theta^- = \Theta_i - \Theta^+$.

The method to compute the value of ω_X is explained in the following. When the initial speed ω_a is close to the maximum speed ω^+, the desired value for ω_X may be larger than the maximum speed ($\omega_X > \omega^+$). In this case, the minimum time is obtained with a slightly different acceleration profile:

- accelerate with the maximum acceleration α^+ until reaching the maximum speed ω^+, covering an angle Θ_i^+;
- remain at the maximum speed ω^+ for an angle $\Theta^=$;
- decelerate with maximum deceleration α^- from speed ω^+, covering an angle $\Theta^- = \Theta_i - \Theta^+ - \Theta^=$.

The acceleration profiles in the two considered cases are illustrated in Figure 17(a) and Figure 17(b), respectively. The inter-arrival times under both cases will now be derived. For the sake of simplicity, the intermediate algebraic steps needed to obtain the results will be omitted.

Case (a). The acceleration profile in Figure 17(a) is defined by the following system of physical equations:

$$\begin{cases} \Theta_i = \Theta^+ + \Theta^- \\ \omega_X = \sqrt{\omega_a^2 + 2\Theta^+\alpha^+} \\ \omega_b = \sqrt{\omega_X^2 + 2\Theta^-\alpha^-} \end{cases}$$

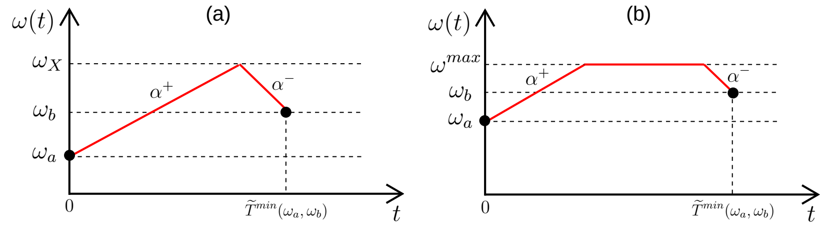


Figure 17. Limit-case acceleration profiles (within the angular period of an AVR task) that lead to minimum inter-arrival times. A job $J_{i,k}$ is released at speed ω_a (time $t=0$) and the following job $J_{i,k+1}$ is released at speed ω_b after $\widetilde{T}_i(\omega_a,\omega_b)$ time units.

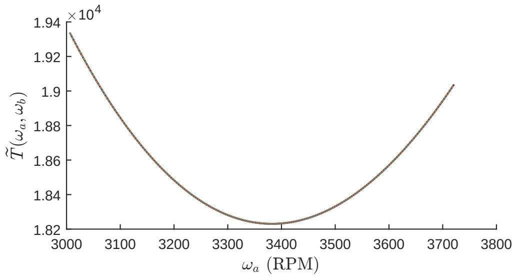


Figure 18. Example plot of $\widetilde{T}(\omega_a,\omega_b)$ as a function of ω_a for $\omega_b = 3000$ RPM.

Solving the above system of equations for ω_X gives

$$\omega_X = \sqrt{\frac{\omega_a^2\alpha^- - \omega_b^2\alpha^+ + 2\Theta_i\alpha^+\alpha^-}{\alpha^- - \alpha^+}}. \tag{11}$$

As a consequence, if two consecutive jobs $J_{i,k}$ and $J_{i,k+1}$ are respectively released at speeds ω_a and ω_b, a safe lower-bound $\widetilde{T}(\omega_a,\omega_b)$ on the inter-arrival time between the two jobs can be computed as

$$\widetilde{T}_i(\omega_a,\omega_b) = \frac{\omega_X - \omega_a}{\alpha^+} + \frac{\omega_b - \omega_X}{\alpha^-}. \tag{12}$$

Case (b). The value ω_X (computed according to Equation (11)) can be higher than the maximum speed ω^+. In this case, the acceleration profile in Figure 17(b) leads to the minimum time.

The first and the last part of the acceleration profile are regulated by the following two equations:

$$\begin{cases} \omega^+ = \sqrt{\omega_a^2 + 2\Theta^+\alpha^+} \\ \omega_b = \sqrt{(\omega^+)^2 + 2\Theta^-\alpha^-} \end{cases}.$$

By solving the two equations for Θ^+ and Θ^- we obtain $\Theta^+ = ((\omega^+)^2 - \omega_a^2)/(2\alpha^+)$ and $\Theta^- = (\omega_b^2 - (\omega^+)^2)/(2\alpha^-)$. Now, the angular distance $\Theta^=$ for which the rotation source remains at constant speed ω^+ can be computed as $\Theta^= = \Theta_i - (\Theta^+ + \Theta^-)$.

Once $\Theta^=$ is computed, if two consecutive jobs $J_{i,k}$ and $J_{i,k+1}$ are respectively released at speeds ω_a and ω_b, and $\omega_X > \omega^+$ (according to Equation (11)), a safe lower-bound $\widetilde{T}(\omega_a,\omega_b)$ on the inter-arrival time between the two jobs can be computed as

$$\widetilde{T}_i(\omega_a,\omega_b) = \frac{\omega^+ - \omega_a}{\alpha^+} + \frac{\Theta^=}{\omega^+} + \frac{\omega_b - \omega^+}{\alpha^-}. \tag{13}$$

As for the model in Section 2.2, considering a given job $J_{i,k}$ released at instantaneous speed ω_a, a particular case of Equation (12) and Equation (13) can be derived for computing the minimum inter-arrival time $\widetilde{T}_i^m(\omega_a)$ to have the next job $J_{i,k+1}$ released in mode m (if reachable with the acceleration bounds), that is $\widetilde{T}_i^m(\omega_a) = \widetilde{T}_i(\omega_a,\omega^m)$.

7.1 Monotonicity of inter-arrival times

This section shows that the same properties defined in Section 2.3 also hold for a model of the rotation source without constant acceleration. Unfortunately, due to the analytical complexity of Equations (12) and (13), it is not straightforward to show the monotonicity properties of interest. Below, each case is separately discussed.

Case (a). Given a speed ω_b, Equation (12) is not monotone with respect to speed ω_a. That is, given two speeds ω_a and $\omega_{a'}$, with $\omega_a > \omega_{a'}$, it may happen that $\widetilde{T}_i(\omega_a,\omega_b) > \widetilde{T}_i(\omega_{a'},\omega_b)$. This can be also noted by looking at the sample plot of Equation (12) shown in Figure 18.

However, a physical interpretation of Equation (12) leads to the conclusion that such cases are physically impossible, i.e., the values for ω_a for which Equation (12) is not monotone correspond to speeds for which is not possible to reach ω_b without violating the acceleration bounds.

This can be shown by studying Equation (12). The first derivative of Equation (12) with respect to ω_a is

$$\frac{\partial}{\partial\omega_a}\widetilde{T}_i(\omega_a,\omega_b) = \frac{\omega_a - \sqrt{\frac{2\Theta_i\alpha^-\alpha^+ + \alpha^-\omega_a^2 - \alpha^+\omega_b^2}{\alpha^- - \alpha^+}}}{\alpha^+\sqrt{\frac{2\Theta_i\alpha^-\alpha^+ + \alpha^-\omega_a^2 - \alpha^+\omega_b^2}{\alpha^- - \alpha^+}}}.$$

By equating the first derivative to zero, it is possible to find the stationary point of Equation (12):

$$\frac{\partial}{\partial\omega_a}\widetilde{T}_i(\omega_a,\omega_b) = 0 \;\Rightarrow\; \omega_a = \sqrt{\omega_b - 2\Theta_i\alpha^-} = \Omega_i^-(\omega_b,\alpha^-).$$

Such a point is a *minimum* and corresponds exactly to the maximum speed from which it is possible to reach ω_b without violating the acceleration bounds. Therefore, Equation (12) is monotonically decreasing with ω_a for speeds ω_a from which it is physically possible to reach ω_b.

In a similar manner, it is possible to show that Equation (12) is monotonically decreasing with ω_b, which allows concluding that the equation is also simultaneously decreasing in both variables. Hence, the same properties defined in Section 2.3 hold.

Case (b). Equation (13) is also not monotone with respect to ω_a. However, it is possible to show that it is not monotone only for speeds ω_a that are outside the validity range. The same approach used for case (a) applies.

The first derivative of Equation (13) with respect to ω_a is

$$\frac{\partial}{\partial\omega_a}\widetilde{T}_i(\omega_a,\omega_b) = \frac{\omega_a - \omega^+}{\alpha^+\omega^+}.$$

By equaling the first derivative to zero, it is possible to find the stationary point of Equation (13), that is

$$\frac{\partial}{\partial\omega_a}\widetilde{T}_i(\omega_a,\omega_b) = 0 \;\Rightarrow\; \omega_a = \omega^+.$$

Also in this case, such a point is a minimum. Clearly, there cannot exist valid speeds ω_a that are higher than the maximum speed ω^+, hence the monotonicity property holds for all valid speeds $\omega_a \leq \omega^+$. In a similar manner, the same can be shown for all valid speeds $\omega_b \leq \omega^+$.

To summarize, if the function $\widetilde{T}_i(\omega_a, \omega_b)$ is applied to pairs of speeds ω_a and ω_b that are compatible with the acceleration and speed bounds, in both the cases we have: **(i)** if $\omega_a > \omega'_a$, then $\widetilde{T}_i(\omega_a, \omega_b) < \widetilde{T}_i(\omega'_a, \omega_b)$; **(ii)** if $\omega_b > \omega'_b$, then $\widetilde{T}_i(\omega_a, \omega_b) < \widetilde{T}_i(\omega_a, \omega'_b)$; and **(iii)** if $\omega_a > \omega'_a \wedge \omega_b > \omega'_b$, then $\widetilde{T}_i(\omega_a, \omega_b) < \widetilde{T}_i(\omega'_a, \omega'_b)$.

8 Related work

To the best of our records, a suitable model for AVR tasks has been proposed for the first time in 2012 by Kim, Lakshmanan, and Rajkumar [16], who derived preliminary schedulability results under very simple assumptions. In particular, their analysis applies to a single engine-triggered task with a interarrival time always smaller than the periods of the other tasks, and running at the highest priority level. Negrean et al. [18] discussed the problem of analyzing the mode-changes of engine-triggered tasks by means of standard mode-change analysis techniques. The paper also addressed the case of multiprocessor systems under partitioned scheduling. However, no analysis was detailed in their work. In a keynote speech given at ECRTS 2012, Darren Buttle gave [10] discussed some timing-related issues in automotive software, presenting a common practice adopted in automotive applications to adapt the functionality and the computational requirements of engine-control tasks for different rotation speeds of the engine. Following Buttle's keynote, the real-time community started getting interested to the analysis of engine-triggered tasks, producing various solutions under different modeling approaches, assumptions, and scheduling policies.

Preliminary results concerning the analysis of engine-triggered tasks under *fixed-priority scheduling* have been presented by Pollex et al. [19], [20] in 2013. In [20], the authors presented a sufficient schedulability analysis under the assumption of arbitrary, but *fixed* engine speed, thus ignoring the potentially dangerous effect caused by mode-changes. Subsequently, in [19], the same authors proposed a simple analysis based on a transformation of engine-triggered tasks to sporadic tasks. The first relevant milestone is due to Davis et al. [11], [12], who in 2014 presented a sufficient ILP-based analysis for task sets including both periodic and engine-triggered tasks, where the latter are activated by the same rotation source. The physical constraints of the system have been considered for setting up an ILP formulation that computes an upper bound of the interference generated by engine-triggered tasks. The ILP formulation applies to a given speed range, and hence requires a quantization of the speed domain for being used in a schedulability test. Feld and Slomka [13] derived an analysis for variable rate tasks that with arbitrary angular phases, but their approach cannot be applied in the presence of other periodic tasks.

The present paper extends the results presented in [6] and [5] by including: **(i)** the generalization of the results to make them independent from the rotation source model and the corresponding consideration of arbitrary acceleration patterns; **(ii)** an extended and more formal presentation of the theoretical foundations for the derivation of dominant speeds; **(iii)** an additional pruning condition that allows speeding up the computation of response times; **(iv)** the algorithm for computing dominant speeds; **(v)** additional experimental results to evaluate the running time of the proposed analysis technique.

Other authors looked into the dynamic-priority scheduling of AVR tasks by adopting the *earliest deadline first* (EDF) algorithm. Most relevant to this paper are the works by Buttazzo et al. [9] and Biondi and Buttazzo [3], who proposed utilization-based schedulability tests, and Guo and Baruah [14], who proposed sufficient tests for constrained-deadline tasks and speedup factors. Still concerning EDF scheduling, Biondi et al. [4] presented an exact feasibility analysis for AVR tasks based on dominant speeds. Finally, Mohaqeqi et al. [17] provide an alternative analysis method by transforming the AVR task model in a task digraph and applying standard digraph analysis methods to the resulting model. On the design side, accurate heuristics for the selection of the transition speeds (and the task priority) have been presented by Biondi et al. [7], where the engine performance is optimized with respect to a general speed-dependent performance model.

9 Conclusion

This paper presented an exact response-time analysis for task sets consisting of periodic/sporadic tasks and AVR tasks with a common activation source, all managed under fixed-priority scheduling. The analysis is based on the notion of *dominant speeds*, which allow to drastically restrict the scenarios that have to be considered for computing the worst-case interference generated by AVR tasks. This result allows a designer to precisely analyze the behavior of engine control applications in the temporal domain, providing a method for predicting possible overload conditions that could jeopardize the system performance. Experimental results show that the proposed approach always dominates the previous sufficient tests, with significant improvements in terms of schedulability for high processor workloads (80% utilization or higher), which represent the typical operating conditions of engine control applications.

As a future work, we plan to extend the response time analysis to sets with multiple AVR tasks with different angular periods and phases and possibly different independent activation sources.

References

[1] N. Audsley, A. Burns, M. Richardson, K. Tindell, and A. Wellings. Applying new scheduling theory to static priority pre-emptive scheduling. *Software Engineering*, 8(5):284–292, Sept. 1993.

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

[3] 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)*, pages 525–530, Grenoble, France, March 9-13, 2015.

[4] A. Biondi, G. Buttazzo, and S. Simoncelli. Feasibility analysis of engine control tasks under EDF scheduling. In *Proc. of the 27th Euromicro Conference on Real-Time Systems (ECRTS 2015)*, Lund, Sweden, July 8-10, 2015.

[5] 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.

[6] 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.

[7] A. Biondi, M. D. Natale, and G. Buttazzo. Performance-driven design of engine control tasks. In *Proceedings of the 7th International Conference on Cyber-Physical Systems (ICCPS 2016)*, Vienna, Austria, April 11-14, 2016.

[8] J. E. Bobrow, S. Dubowsky, and J. S. Gibson. Time-optimal control of robotic manipulators along specified paths. *The International Journal of Robotics Research*, 4:3–17, 1985.

[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*, Dresden, Germany, March 24-28, 2014.

[10] 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.

[11] 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 2014.

[12] R. I. Davis, T. Feld, V. Pollex, and F. Slomka. Schedulability tests for tasks with variable rate-dependent behaviour under fixed priority scheduling. In *University of York, Department of Computer Science Technical Report, YCS-2014-488*, January 2014.

[13] T. Feld and F. Slomka. Sufficient response time analysis considering dependencies between rate-dependent tasks. In *Proc. of the International Conference on Design, Automation and Test in Europe (DATE 2015)*, pages 519–524, Grenoble, France, March 9-13, 2015.

[14] 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.

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

[16] 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)*, pages 28–38, Beijing, China, April 2012.

[17] M. Mohaqeqi, J. Abdullah, P. Ekberg, and W. Yi. Refinement of workload models for engine controllers by state-space partitioning. In *Proc. of the 29th Euromicro Conference on Real-Time Systems (ECRTS 2017)*, Dubrovnik, Croatia, June 2017.

[18] M. Negrean, R. Ernst, and S. Schliecker. Mastering timing challenges for the design of multi-mode applications on multi-core real-time embedded systems. In *Proc. of the Embedded Real Time Software and Systems 2012 (ERTS2)*, Toulouse, France, Feb. 1-3 2012.

[19] V. Pollex, T. Feld, F. Slomka, U. Margull, R. Mader, and G. Wirrer. Sufficient real-time analysis for an engine control unit. In *21st International Conference on Real-Time Networks and Systems*, Sophia Antipolis, France, Oct. 16-18 2013.

[20] 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.

[21] M. Stigge and W. Yi. Combinatorial abstraction refinement for feasibility analysis. In *Proceedings of the 34nd IEEE Real-Time Systems Symposium (RTSS 2013)*, 2013.

[22] M. Stigge and W. Yi. Refinement-based exact response-time analysis. In *Proc. 26th Euromicro Conference on Real-Time Systems (ECRTS'14)*, 2014.

[23] E. Velenis and P. Tsiotras. Minimum-time travel for a vehicle with acceleration limits: Theoretical analysis and receding-horizon implementation. *Journal of Optimization Theory and Applications*, 138(2):275–296, 2008.

Alessandro Biondi is post-doc researcher at the Real-Time Systems (ReTiS) Laboratory of the Scuola Superiore Sant'Anna. He graduated (cum laude) in Computer Engineering at the University of Pisa, Italy, within the excellence program, and received a Ph.D. in computer engineering at the Scuola Superiore Sant'Anna under the supervision of Prof. Giorgio Buttazzo and Prof. Marco Di Natale. In 2016, he has been visiting scholar at the Max Planck Institute for Software Systems (Germany). His research interests include design and implementation of real-time operating systems and hypervisors, schedulability analysis, cyber-physical systems, synchronization protocols, and component-based design for real-time multiprocessor systems. He was recipient of four Best Paper Awards, and an Outstanding Paper Award.

Marco Di Natale is an IEEE Senior member and Full Professor at the Scuola Superiore SantAnna. He received his PhD from Scuola Superiore SantAnna and was a visiting Researcher at the University of California, Berkeley in 2006 and 2008. He is currently visiting Fellow for the United Technologies corporation. He's been a researcher in the area of realtime and embedded systems for more than 20 years, being author or co-author of more than 200 scientific papers, winner of six best paper awards and one best presentation award.

Giorgio Buttazzo is full professor of computer engineering at the Scuola Superiore Sant'Anna of Pisa. He graduated in electronic engineering at the University of Pisa in 1985, received a M.S. degree in computer science at the University of Pennsylvania in 1987, and a Ph.D. in computer engineering at the Scuola Superiore Sant'Anna of Pisa in 1991. From 1987 to 1988, he worked on active perception and real-time control at the G.R.A.S.P. Laboratory of the University of Pennsylvania, Philadelphia. He has been Program Chair and General Chair of the major international conferences on real-time systems and Chair of the IEEE Technical Committee on Real-Time Systems. He is Editor-in-Chief of Real-Time Systems, Associate Editor of the ACM Transactions on Cyber-Physical Systems, and IEEE Fellow since 2012. He has authored 7 books on real-time systems and over 200 papers in the field of real-time systems, robotics, and neural networks.