Exact Analysis of Adaptive Variable-Rate Tasks under Fixed-Priority Scheduling

Alessandro Biondi, Alessandra Melani, Mauro Marinoni, Marco Di Natale, Giorgio Buttazzo
Scuola Superiore Sant'Anna, Pisa, Italy

Abstract

Engine control applications require the execution of tasks activated in relation to specific system variables, as the crankshaft rotation angle. To prevent possible overload conditions at high rotation speeds, such tasks are designed to vary their functionality (hence their computational requirements) for different speed ranges. Modeling and analyzing such a type of tasks poses new research challenges in the schedulability analysis that are now being addressed in the real-time literature. This paper advances the state of the art by presenting a method for computing the exact worst-case interference of such adaptive variable-rate tasks under fixed priority scheduling, enabling a precise analysis and design of engine control applications.

1. Introduction

A large variety of real-time task models have been proposed in the literature to analyze the schedulability of different types of embedded systems. The classical Liu and Layland periodic task model [12] captures the typical structure of control loops providing an implementation for discrete-time controllers. The sporadic task model introduced by Mok [13] captures the intrinsic irregular arrival sequence of external events, while providing a bound on the worst-case arrival rate necessary to derive an off-line schedulability analysis.

A rate-based execution abstraction was introduced by Jeffay et al. [9], [10] to generalize the classical periodic and sporadic scheme. According to such a model, a task specifies its expected rate as the maximum number x of executions expected to be requested in any interval of length y, however the maximum computation time required for any job of the task is fixed, while the actual distribution of events in time is arbitrary.

The multi-frame model proposed by Mok and Chen [14] provides the additional expressivity to capture conditional executions and execution patterns. In this model, tasks are activated periodically, but the execution time of each job varies according to a predefined pattern. Such a model has been later generalized by Baruah et al. [2] to allow jobs to be separated by a varying interarrival time. However, in both cases the activation pattern is known a priori and does not depend on any state variable.

An elastic model has been presented by Buttazzo et al. [3], [5] to tolerate and handle permanent overload conditions in periodic real-time systems. According to this model, a task has a fixed computation time, but a variable period, which can be varied in a given range. An overload condition is then handled by properly compressing task utilizations as they were elastic springs with given elastic coefficients, expressing the availability of each task to change its period.

More recently, the consideration of a fuel injection systems, as representatives of a possibly larger class of applications, has highlighted the limitations of the existing approaches and the need for a new type of task model and analysis.

The general goal of a fuel injection system is to determine the point(s) in time and the quantity of fuel to be injected in the cylinders of an engine, relative to the position of each piston, which is in turn a function of the angular position of the crankshaft. In a reciprocating engine, the dead centre is the position of a piston in which it is farthest from, or nearest to, the crankshaft. The former is the *top dead centre* (TDC) while the latter is the *bottom dead centre* (BDC), as illustrated in Figure 1. In a four-cylinder engine, the pistons are paired in phase opposition, so that, when two of them are in a TDC, the others are in a BDC. The TDC is the typical reference point, in the controller activities, for the functions and actions that need to take place within the rotation. These action include (among others) computing the phase (time relative to the TDC) of the injection and the quantity of fuel to be injected, but also checking whether the combustion occurred properly. Depending on the structure of the engine control application, these functions are implemented in tasks that are activated at each TDC, that is twice every crankshaft rotation (pseudo-cycle) or even more frequently (half-TDC).

The problem with this type of tasks is that the time between two activations (at the TDC) is not constant, nor arbitrary, but depends on the rotation speed of the engine, which can vary within given ranges with a certain maximum acceleration. Thus, the acceleration bounds define a space of possible activation times which is not easy to capture and analyze without incurring in excessive pessimism (as it would, for example, if using the sporadic model).

To further complicate the treatment, the (worst-case) execution time of the functions executed by such tasks is typically not constant. At low revolution rates, the time interval between two reference points (the TDC for a set of cylinders) is large and allows the execution of sophisticated controls (and possibly multiple fuel injections). The same algorithm cannot be executed at higher revolution rates, because it would lead to an overload, generating several deadline misses. Therefore, the implementation

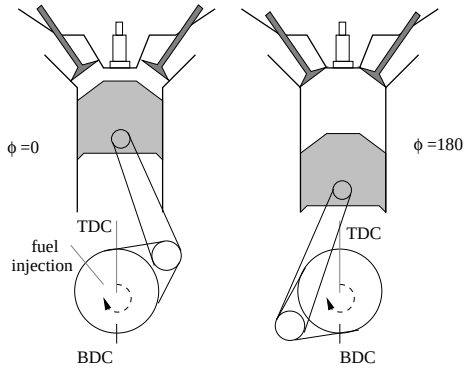

Figure 1: Relationship among engine phases and reference points in the crankshaft rotation period. In a 4-cylinder engine, cylinder pairs are in phase opposition.

is adapted using a simplified algorithm that reduces the WCET (i.e., the functions to be executed) when the rotation speed falls within pre-defined ranges. For most cars, the rotation speed typically varies between 600 and 6000 revolutions per minute (rpm), which maps to activation intervals between 100 and 10 ms, for a complete revolution. A simplified representation of such an adaptive behavior is shown in Figure 2.

```
#define omega1 1000
#define omega2 2000
#define omega3 4000
#define omega4 6000

task  sample_task {
      omega = read_rotation_speed();
      f0();
      if (omega ≤ omega4) f1();
      if (omega ≤ omega3) f2();
      if (omega ≤ omega2) f3();
      if (omega ≤ omega1) f4();
}
```

Figure 2: Typical implementation of a task with a functionality variable with the rotation speed of the crankshaft.

In this paper, the model proposed to describe such a type of engine control tasks is referred to as *Adaptive Variable Rate* task model, or AVR-model. Overall, a subset of the system tasks can be characterized as AVR tasks, typically executing together with classical periodic tasks under the control of a fixed-priority scheduler, as established in the automotive AUTOSAR standard.

Adaptivity of execution times can also be captured by considering that tasks may exhibit different execution modes. Note however, that mode change analysis [16]–[18] is not suited for analyzing AVR tasks, since their activation period is changing continuously, thus an infinite number of modes would be required to describe all possible situations.

1.1. Related work

In the real-time community, the problem related to AVR tasks was first presented by Buttle [6], as a common practice adopted in automotive applications to adapt the functionality and the computational requirements of engine control tasks for different rotation speeds.

A suitable model for AVR tasks with activation rates and execution times depending on the angular velocity of the engine has been proposed for the first time by Kim, Lakshmanan, and Rajkumar [11], who derived preliminary schedulability results under very simple assumptions. In particular, their analysis applies to a *single* AVR task with a period always smaller than the periods of the other tasks, and running at the highest priority level. In addition, all relative deadlines are assumed to be equal to periods and priorities are assigned based on the rate-monotonic algorithm.

Pollex et al. [15] also presented a sufficient schedulability analysis under fixed priorities, but they assumed that all the tasks with a variable rate depend on the same angular velocity, which can be arbitrary, but fixed. Moreover, the analysis is formulated using continuous intervals, hence it cannot be immediately translated into a practical schedulability test, whose complexity has not been evaluated.

The schedulability analysis of a generic set of AVR tasks under steady-state and dynamic conditions (considering a maximum acceleration of the engine) has been addressed by Buttazzo, Bini, and Buttle [4] under Earliest Deadline First (EDF) scheduling. They also provided a design method that allows computing the set of switching rotation speeds that keep the overall utilization below a desired bound.

The dynamic analysis of AVR tasks under fixed-priority scheduling has been considered by Davis et al. [7]. After discussing the complexity of the problem, they presented a sufficient test based on an Integer Linear Programming (ILP) formulation. Besides of being only sufficient, their approach is based on a quantization of the instantaneous crankshaft rotation speed, which may introduce additional pessimism in the analysis to guarantee the safety of the test.

1.2. Contributions and structure

This paper provides the following novel contributions:

1) it presents an exact analysis of the worst-case interference generated by an AVR task in dynamic situations, under fixed priority systems and arbitrary deadlines, assuming realistic acceleration bounds derived from the automotive industry;
2) it discusses an efficient method for reducing the worst-case complexity of the exact analysis by identifying a set of cases that dominate the others, thus avoiding the need of a quantization of the engine state variable;
3) it presents a set of simulation experiments to compare the proposed analysis with the ILP-based test proposed by Davis et al. [7], showing that the interference computed by the ILP-based approach is always greater or equal to the one computed by our method;

4) finally, it shows that the quantization process used in the ILP-based test makes the schedulability analysis unsafe, since for some intermediate values of the state variable, the interference computed by the ILP approach can be lower than the worst-case one.

The rest of the paper is organized as follows: Section 2 introduces the system model; Section 3 defines the interference of an AVR task and presents a brute force approach to compute it; Section 4 illustrates an efficient method for reducing the worst-case complexity for computing the exact interference by taking advantage of a pruning rule; Section 6 evaluates the performance of the proposed approach against the ILP-based test proposed by Davis et al. [7]; Section 7 states our conclusion and future work.

2. System model and notation

This section presents the model for the task set, for the engine dynamics, and the functions that are used to estimate the future activation and the execution mode of the next job instance.

2.1. Task model

This paper considers a computing system running a set of N preemptive real-time tasks $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_N\}$ under fixed priorities and arbitrary deadlines. Each task τ_i generates an infinite sequence of jobs, $J_{i,1}, J_{i,2}, \ldots$ and can be either a periodic task, characterized by a fixed worst-case execution time (WCET) C_i, period T_i, and relative deadline D_i, or an AVR task, where both C_i, T_i, and D_i are variable. For the sake of clarity, whenever needed, a rate-adaptive task may also be denoted as τ_i^*.

The peculiarity of an AVR task τ_i^* is that its activation pattern and functionality are determined by the physical evolution of the engine. In particular, a generic job $J_{i,k}$ of an AVR task is activated when the crankshaft reaches predefined angular positions, thus the interarrival time between $J_{i,k}$ and $J_{i,k+1}$, denoted as *period* $T_{i,k}$, is a function of the crankshaft rotation speed. The sequence of activation times of the jobs of τ_i is denoted as $t_{i,1}, t_{i,2}, \ldots$, and we assume that activations are triggered at angular intervals of $\Delta\theta_i$. The relative deadline $D_{i,k}$ of job $J_{i,k}$ can be set arbitrarily as a non-decreasing function of $T_{i,k}$.

To cope with such a high variability in the release times, an AVR task τ_i^* is typically implemented as a set $\mathcal{M}_i$ of M_i different modes,

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

each characterized by a certain functionality and WCET, kept constant when job activation periods fall in the range $[T_i^m, T_i^{m+1})$, where $T^{M_{i+1}} = T_i^{max}$ represents the maximum activation period allowed by the system. Hence, the computation time of a generic AVR job $J_{i,k}$ can be expressed as a non-decreasing step function $\mathcal{C}$ of the current job period $T_{i,k}$, that is,

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

An example of $\mathcal{C}$ function is illustrated in Figure 3.

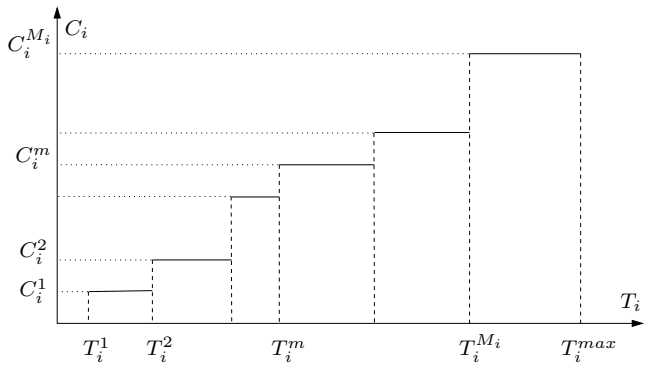

Figure 3: Computation time of an AVR task as a function of the period.

When a job $J_{i,k}$ is activated at time $t_{i,k}$, the release time of the next job ($t_{i,k+1}$) is not known, so the exact value of $T_{i,k}$ (necessary to select the corresponding mode) cannot be computed, but can only be estimated by a proper function.

In the following sections, we are interested in computing the contribution to the interference of each individual AVR task (for periodic tasks the computation is trivial). To simplify the notation, the task index will be omitted from the task parameters whenever we refer to a single AVR task.

2.2. Engine dynamics and task activations

To perform schedulability analysis in the presence of AVR tasks, it is important to characterize the relation between the engine dynamics and the task parameters. The engine speed at time t is denoted as $\omega(t)$ and, as a notation shortcut, the speed at the activation time of the generic k-th job is indicated by $\omega_k = \omega(t_k)$. The speed is bounded in the interval $[\omega_{min}, \omega_{max}]$, where the minimum speed ω_{min} defines the longest task period $T^{max} = \Delta\theta/\omega_{min}$, while the maximum speed ω_{max} defines the smallest period $T^1 = \Delta\theta/\omega_{max}$ related to the first mode. The engine acceleration at time t, denoted by $\alpha(t)$, is assumed to be bounded between a maximum deceleration α^- and a maximum acceleration α^+, so that $\alpha(t) \in [\alpha^-, \alpha^+]$.

Given the current engine state (ω_k, α_k) at time t_k, the time to the next job release is modeled (assuming a constant acceleration α_k) by the following function [4]:

$$T(\omega_k, \alpha_k) = \frac{\sqrt{\omega_k^2 + 2\alpha_k\Delta\theta} - \omega_k}{\alpha_k}, \quad (2)$$

where $\Delta\theta$ is the angular displacement that determines two consecutive job activations. Similarly, the instantaneous engine speed at the next job release is modeled (under the same assumption of constant acceleration) as:

$$\Omega(\omega_k, \alpha_k) = \sqrt{\omega_k^2 + 2\alpha_k\Delta\theta}. \quad (3)$$

In this paper we use an estimator typically adopted by the industry, which assumes zero acceleration in $[t_k, t_{k+1}]$, meaning that the period is computed as $T_k = t_k - t_{k-1} = T(\omega_{k-1}, \alpha_{k-1})$.

In addition, the following notation is used throughout the paper:

- Ω^n denotes the engine speed after n job releases, computed as $\Omega^n(\omega_k, \alpha_k) = \Omega(\Omega^{n-1}(\omega_k, \alpha_k), \alpha_k)$, where $\Omega^0(\omega_k, \alpha_k) = \omega_k$.
- T^n denotes the period estimate of the n-th following job, assuming constant acceleration, computed as $T^n(\omega_k, \alpha_k) = T(\Omega^{n-1}(\omega_k, \alpha_k), \alpha_k)$.
- $\Omega^{-1}(\omega_k, \alpha)$ denotes the inverse function of Ω, and, given a speed ω_k at time t_k, returns the speed ω_{k-1} at time t_{k-1}, assuming a constant acceleration α in $[t_{k-1}, t_k]$.
- $T^{-1}(T_k, \alpha)$ denotes the inverse function of T, and, given the current estimated period $T_k = t_k - t_{k-1}$, returns the speed ω_{k-1} at time t_{k-1}, assuming a constant acceleration α in $[t_{k-1}, t_k]$.

Figure 4 illustrates a range of possible scenarios that may determine the next activation for different speed and accelerations.

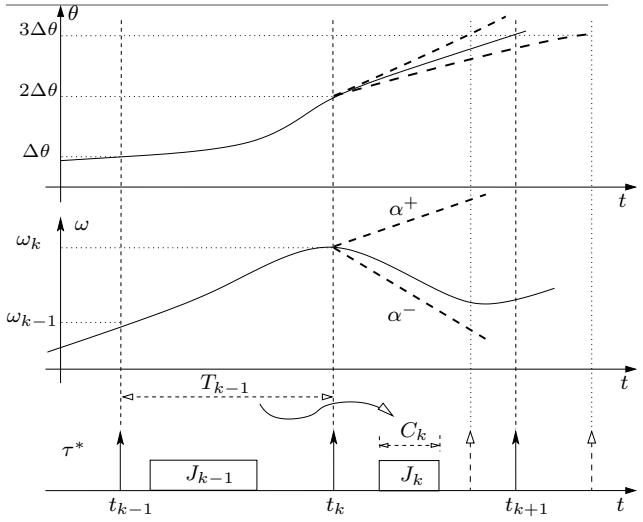

Figure 4: Dependence of task parameters from the system state.

A crucial aspect that must always be considered in the analysis presented in this paper is that the use of an estimator prevents using the relation $\omega_k = \Delta\theta / T_k$ (except for the limit values), because it would not consider the real task mode-change behavior.

Note that, if at time t_k we only know the instantaneous engine speed ω_k and nothing is known about the previous state of the system, the greatest possible estimation $\widetilde{T_k}$ of the period can be computed (assuming the maximum acceleration of the engine) as

$$\widetilde{T}(\omega_k) = T(\Omega^{-1}(\omega_k, \alpha^+), \alpha^+). \quad (4)$$

Likewise, given an estimate T_k of the activation period, the largest possible speed $\widetilde{\omega_k}$ at time t_k can be computed (assuming the maximum acceleration of the engine) as

$$\widetilde{\Omega}(T_k) = \Omega(T^{-1}(T_k, \alpha^+), \alpha^+). \quad (5)$$

Using Equation (5), a mode transition can be expressed in terms of speed as

$$\omega^m = \widetilde{\Omega}(T_m). \quad (6)$$

3. Characterizing the interference

The interference caused by an AVR task τ_h depends on the speed of the engine when the critical instant occurs. Hence, the interference is a function of the dynamics of the engine and should be computed for any initial speed ω_0.

$I_{\omega_0}(t)$ denotes the worst-case interference generated by an AVR task in the interval $[0, t]$, assuming that the critical instant occurs at time 0, when the speed of the engine is ω_0.

For each initial engine speed ω_0, we determine $\Omega(\omega_0, \alpha)$ and $T(\omega_0, \alpha)$, considering a generic acceleration α. A brute-force approach requires considering all possible values of $\alpha \in [\alpha^-, \alpha^+]$ to determine all possible subsequent job releases occurring in the interference window. Let us consider a job J_0 released at t_0, when the instantaneous speed is ω_0. Such an activation gives rise to a family of possible instances of J_1, with period in the range $[T(\omega_0, \alpha^+), T(\omega_0, \alpha^-)]$ and corresponding engine speed ω_1 in the range $[\Omega(\omega_0, \alpha^-), \Omega(\omega_0, \alpha^+)]$, according to the model described in Section 2.

Similarly, the next activation gives rise to a set of possible job instances with period in $[T(\omega_1, \alpha^+), T(\omega_1, \alpha^-)]$ and speed $\omega_2 \in [\Omega(\omega_1, \alpha^-), \Omega(\omega_1, \alpha^+)]$. This reasoning applies recursively for each subsequent activation, until the end of the interference window, leading to a *tree* of possible job releases, as depicted in Figure 5.

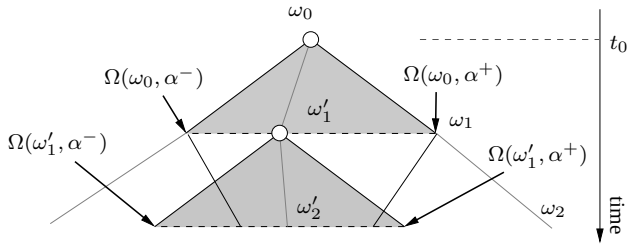

Figure 5: Tree of possible job releases of an AVR task.

Assuming the acceleration to vary continuously, the number of branches at each recursive step is theoretically infinite, but can be bounded by making the acceleration range discrete with a given granularity. Of course, any quantization of the acceleration domain makes the analysis approximate, and a large granularity, while reducing the search complexity, increases the pessimism.

The pseudocode in Figure 6 summarizes the recursive procedure for computing the maximum interference following a brute-force approach. The procedure is called by passing the initial engine speed ω_0, the longest computation time C_0 of any mode reachable from ω_0, and the current time instant.

In particular, since at the first activation the previous job release is unknown, C_0 is computed as a function of the largest period leading to the initial speed ω_0, that is: $\mathcal{C}(\widetilde{T}(\omega_0))$.

At each recursive step, until the maximum deadline of any task (i.e. the largest length of the interference window that needs to be considered by the analysis, also denoted as MAXTIME, as in lines 2-3), we must:

```
1: procedure INTERFERENCE(ω, C, t)
2:     if t > MAXTIME then return ;
3:     end if
4:     UPDATEINTERFERENCEFN(C, t);
5:     for α = α⁻ to α⁺ step Δα do
6:         ω^next ← Ω(ω, α);
7:         T^next ← T(ω, α);
8:         C^next ← C + C(T^next);
9:         INTERFERENCE(ω^next, C^next, t + T^next);
10:    end for
11: end procedure
```

Figure 6: Procedure for computing the interference of an AVR performing an exhaustive tree-search.

- update the worst case interference function for a given value of t (line 4);
- for each acceleration:
  - compute the speed and the period related to the next job (lines 6-7);
  - accumulate the overall computational request (line 8);
  - recursively call the function INTERFERENCE (line 9).

The maximum interference function is a stepwise function storing the maximum interference time for each possible value t. The procedure UPDATEINTERFERENCEFN at line 4 simply saves the maximum cumulative value of the computational demand for the time instant passed as argument.

In summary, for a given ω_0, the procedure generates a *tree* of job releases. Hence, computing the interference $I_{\omega_0}(t)$ is a search problem in the speed domain, and requires a complete visit of the tree. The interference $I_{\omega_0}(t)$ corresponds to the maximum among the interferences generated from all possible job sequences starting with speed ω_0. This procedure is very expensive in terms of computational complexity, and intractable for most practical uses. The next section determines a pruning rule that cuts a significant number of branches, while still guaranteeing an exact interference analysis.

4. Computing the exact interference

To cut redundant branches in the search tree, we note that for each job release after the critical instant, only a finite set of critical job releases must be taken into account to derive the maximum interference. We explain how to compute such critical job releases, and then derive a pruning rule for the search problem presented in the previous section.

First of all, we construct the interference generated by an AVR task as the sum of the possible contributions of its individual jobs. To determine the potential interference generated by a single job J_a, it is necessary to consider all possible releases of J_{a+1} compatible with the given acceleration $\alpha \in [\alpha^-, \alpha^+]$.

4.1. Potential-job interference

Definition 1: Given a job J_a of an AVR task activated in mode m at time t_a, the *potential-job interference* $i_{\omega_a,m}(\delta)$ of J_a is the maximum computational request generated by J_a and J_{a+1}, in the interval [0,δ], for all possible releases of J_{a+1} at $t_{a+1} = t_a + \delta$.

As explained in Section 3, job releases depend on 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, $i_{\omega_a,m}(0) = C^m$, since J_a is released in mode m. Considering the maximum acceleration α^+, it is easy to see that no job release can occur in the interval $0 < \delta < T(\omega_a, \alpha^+)$; therefore, in this interval $i_{\omega_a,m}(\delta) = C^m$.

For $T(\omega_a, \alpha^+) \leq \delta \leq T(\omega_a, \alpha^-)$, an additional job release must be considered: the earliest considering the maximum acceleration α^+, the latest considering the maximum deceleration α^-. Depending on the engine dynamics, the released job can belong to a number of different modes. The larger the acceleration/deceleration range, the greater the number of possible modes. Since we are modeling the worst-case interference of an AVR task, we have to take into account all possible job releases for each possible mode m' with $T^{m'} \in [T(\omega_a, \alpha^+), T(\omega_a, \alpha^-)]$ (T^m is included in the range). Finally, for $\delta > T(\omega_a, \alpha^-)$, no release of J_{a+1} can occur; hence, $i_{\omega_a,m}(\delta) = \mathcal{C}(T(\omega_a, \alpha^-))$ is the computational request of the latest possible job release time. Overall, $i_{\omega_a,m}(\delta)$ is a non-decreasing stepwise function, where each step represents the release of a different mode m'.

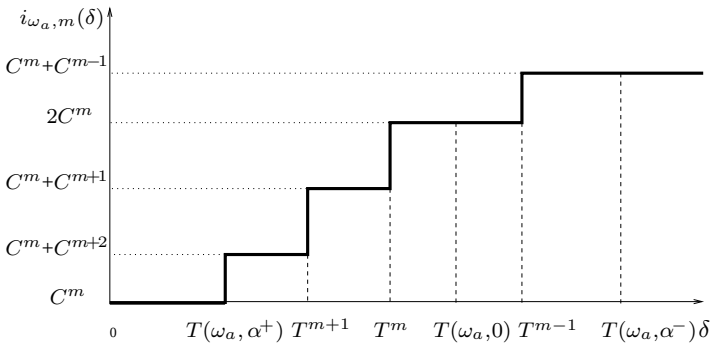

Figure 7: Potential job interference of a job activated at time t_a in mode m.

Figure 7 shows an example of potential job interference assuming that the engine dynamics allows to release jobs of two different modes in acceleration, and one in deceleration. As explained is Section 3, the interference $I_{\omega_0}(t)$ of an AVR task is the maximum sum of all its possible job requests. Therefore, it is possible to express $I_{\omega_0}(t)$ as the sum of time-shifted potential job interferences.

The following theorem allows identifying the job instances for which the interference dominates the one for job instances at other release times and therefore it can be used as a pruning rule to reduce the search space.

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 $\mathcal{C}(T(\omega_a, \alpha^-)) = \mathcal{C}(T(\omega_b, \alpha^-))$, then $\forall \delta \geq 0 \;\; i_{\omega_a,m}(\delta) \geq i_{\omega_b,m}(\delta)$.

Proof: The proof is trivial for $\omega_a = \omega_b$, therefore in the following we assume $\omega_a > \omega_b$. Since, for a given α, $T(\omega, \alpha^+)$ and $T(\omega, \alpha^-)$ are both 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 $i_{\omega_a,m}(\delta) = i_{\omega_b,m}(\delta) = C^m$ for $\delta < T(\omega_a, \alpha^+)$. For $T(\omega_a, \alpha^+) \leq \delta < T(\omega_b, \alpha^+)$ we have $i_{\omega_b,m}(\delta) = C^m$ (job releases after J_b cannot occur before $T(\omega_b, \alpha^+)$), while $i_{\omega_a,m}(\delta)$ can be 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 $i_{\omega_a,m}(\delta) > i_{\omega_b,m}(\delta)$.

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

- $T(\omega_b, \alpha^+) \leq T(\omega_a, \alpha^-)$, i.e., the two single-job interferences are overlapped in time. In this case, for $T(\omega_b, \alpha^+) \leq \delta \leq T(\omega_a, \alpha^-)$, we have $i_{\omega_a,m}(\delta) = i_{\omega_b,m}(\delta)$. In this range, the single-job interferences are considering the same activation periods, therefore they model the same computational demand;
- $T(\omega_b, \alpha^+) > T(\omega_a, \alpha^-)$, i.e., the two single-job interferences are non-overlapped in time. In this case, for $T(\omega_b, \alpha^+) \leq \delta \leq T(\omega_a, \alpha^-)$, we have $i_{\omega_a,m}(\delta) > i_{\omega_b,m}(\delta)$ since $i_{\omega_b,m}(\delta) = C^m$.

In both cases, for $\delta > T(\omega_a, \alpha^-)$, we have $i_{\omega_a,m}(\delta) = i_{\omega_b,m}(\delta)$. This follows from (ii) and the hypothesis $\mathcal{C}(T(\omega_a, \alpha^-)) = \mathcal{C}(T(\omega_b, \alpha^-))$, since $i_{\omega,m}(\delta)$ is non-decreasing.

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

Figure 8 shows a typical scenario in which Theorem 1 holds, related to the case $T(\omega_b, \alpha^+) \leq T(\omega_a, \alpha^-)$.

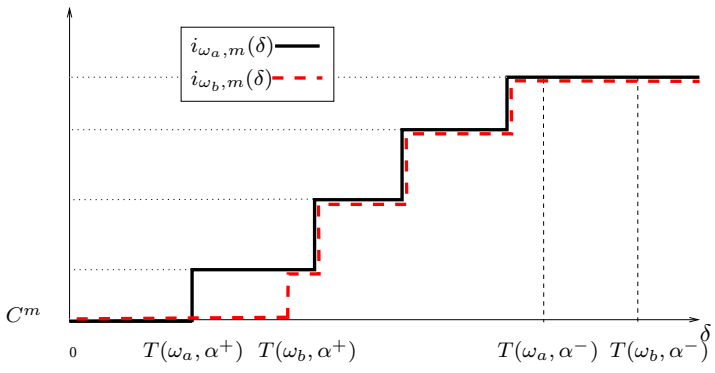

Figure 8: Example of a scenario for applying Theorem 1.

4.2. Pruning rule

In this section, Theorem 1 is used to build a pruning rule for reducing the search space. The complexity is reduced by finding a subset of potential job interferences that contribute to the exact interference $I_{\omega_0}(t)$.

Pruning rule properties. Consider a generic job J_{i-1}, and its immediate follower J_i. The release time of J_i is variable and depends on the engine acceleration. Let $J_{i,(t)}$ be the instance of J_i released at time t (t is one of the activation times allowed by the acceleration range of the engine). The objective of the pruning rule is to identify a limited set $\mathcal{P}$ of *critical job instances* satisfying the following properties:

(i) for each $J_{i,(s)} \notin \mathcal{P}$, there must exists a $J_{i,(t)} \in \mathcal{P}$ for which the potential job interference of $J_{i,(t)}$ dominates the one of $J_{i,(s)}$;
(ii) for each $J_{i,(s)} \notin \mathcal{P}$, there must exists a $J_{i,(t)} \in \mathcal{P}$ such that the interference generated by *all* job releases following $J_{i,(s)}$ is dominated by the interference of *at least one* set of possible job instances following $J_{i,(t)}$.

Property (i) allows to eliminate all the job instances $J_{i,(s)} \notin \mathcal{P}$, while property (ii) allows to discard the entire sub-tree of job instances released after $J_{i,(s)}$.

Job instances for property (i). Suppose that the critical instant occurs when the instantaneous engine speed is ω_0 and the AVR task is in mode m. The potential job interference $i_{\omega_0,m}(t)$ is the initial value of $I_{\omega_0}(t)$. Using Theorem 1, it is possible to identify a set $\mathcal{P}^{(i)} \subset \mathcal{P}$ of job releases for which their potential job interference dominates the others.

Theorem 1 only applies to jobs activated in the same mode. Hence, the set $\mathcal{P}^{(i)}$ includes at least one job instance for each mode m such that $T_m \in [T(\omega_0, \alpha^+), T(\omega_0, \alpha^-)]$. For each mode, we need to consider the *earliest* job release, in order to satisfy the theorem hypothesis $\omega_a \geq \omega_b$. Formally, we refer to such job releases as the *earliest distinct mode changes* (EDMCs). EDMCs can also be defined as the set of the job instances $J_{i,(t)}$ for which their current period T_i belongs to the set T_{EDMC}, which includes the period corresponding to the earliest possible arrival of a job instance (max acceleration) and all the periods that correspond to mode changes, that is

$$T_{\text{EDMC}} = \{T(\omega_0, \alpha^+)\} \cup \{T^m \mid T^m \in [T(\omega_0, \alpha^+), T(\omega_0, \alpha^-)]\}.$$

In the example of Figure 7, the set T_{EDMC} is given by the time instants on the x-axis corresponding to the steps of $i_{\omega_0,m}(t)$.

However, the set $\mathcal{P}^{(i)}$ needs to include other job instances besides those in EDMCs, since there can be job instances $J_{i,(q)}$ that are not dominated by any of the instances in EDMC. This can happen because the instances in EDMC do not necessarily guarantee the last hypothesis of Theorem 1. For these job releases $J_{i,(q)}$, there is no $J_{i,(t)}$ having period $T_{i,(t)} \in T_{\text{EDMC}}$ with $\mathcal{C}(T(\omega_{i,(t)}, \alpha^-)) = \mathcal{C}(T(\omega_{i,(q)}, \alpha^-))$.

The set EDMC needs to be extended to include other instances. Hence, the set EDMC = $\{J_{i,(t_1)}, J_{i,(t_2)}, \ldots J_{i,(t_z)}\}$ is sorted by increasing arrival times. $J_{i,(t_1)}$ is the earliest possible instance. Then, for every interval $[J_{i,(t_m)}, J_{i,(t_{m+1})}]$ we need to look for an intermediate time point (arrival time) tq that corresponds to an instance not dominated by the endpoint $J_{i,(t_m)}$ because $\mathcal{C}(T(\omega_{i,(t_m)}, \alpha^-)) \neq \mathcal{C}(T(\omega_{i,(tq)}, \alpha^-))$.

The candidate time instants tq can be computed by considering that they can only belong to the set of points for which $T(\omega_{i,(tq)}, \alpha^-) = T^m$ for one of the modes m.

If such tq exists, it is added to the set and the test continues in the interval $[J_{i,(tq)}, J_{i,(tm+1)}]$. When all points in $[J_{i,(tm)}, J_{i,(tm+1)}]$ are checked, the algorithm moves to the next time interval in the original set EDMC until all the dominant job instances are found.

Job instances for property (ii). Unfortunately, Theorem 1 is not sufficient to discard *all* the interferences generated by the jobs following $J_{i,(s)}$. For example, consider a generic job instance $J_{i,(s)}$ for which the potential job interference is dominated by $J_{i,(t)} \in \mathcal{P}^{(i)}$. Since by hypothesis we have $\omega_{i,(s)} \leq \omega_{i,(t)}$, a job $J_{i+1,(s')}$ immediately following $J_{i,(s)}$ could have a higher period than *all* the possible jobs instances $J_{i+1,(t')}$ immediately following $J_{i,(t)}$ (formally, the maximum periods for $J_{i+1,(t')}$ and $J_{i+1,(s')}$ are respectively $T_{i+1,(t')} = T(\Omega(\omega_{i,(t)}, \alpha^-), \alpha^-)$ and $T_{i+1,(s')} = T(\Omega(\omega_{i,(s)}, \alpha^-), \alpha^-) > T_{a+1}$). Hence, $J_{i,(s)}$ cannot be pruned, since it could be that $\mathcal{C}(T_{i+1,(s')}) > \mathcal{C}(T_{i+1,(t')})$.

The following theorem addresses this issue. To compact the readability of the theorem, we define the following set:

$$\overline{\mathbb{N}} = \{k \in \mathbb{N}^+ | \max(T^k(\omega_a, \alpha^-), T^k(\omega_b, \alpha^-)) \leq T^{max}\}.$$

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 $\forall n \in \overline{\mathbb{N}}$ $\mathcal{C}(T^n(\omega_a, \alpha^-)) = \mathcal{C}(T^n(\omega_b, \alpha^-))$, 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 $\omega_a > \omega_b$. We must show that, for each possible job J_{b+n} following J_b there exists a job J_{a+n} following J_a such that the potential job interference of J_{b+1} is dominated by that of J_{a+1}. To do this, we apply Theorem 1 by induction.

Since $\Omega^n(\omega, \alpha)$ is a monotonic increasing function, we have

$$\forall n \in \overline{\mathbb{N}} \quad \Omega^n(\omega_a, \alpha^-) \geq \Omega^n(\omega_b, \alpha^-);$$
$$\forall n \in \overline{\mathbb{N}} \quad \Omega^n(\omega_a, \alpha^+) \geq \Omega^n(\omega_b, \alpha^+).$$

Similarly, since $T^n(\omega, \alpha)$ is a monotonic decreasing function, we have

$$\forall n \in \overline{\mathbb{N}} \quad T^n(\omega_a, \alpha^-) \leq T^n(\omega_b, \alpha^-);$$
$$\forall n \in \overline{\mathbb{N}} \quad T^n(\omega_a, \alpha^+) \leq T^n(\omega_b, \alpha^+).$$

For each n, two scenarios are possible as shown in Figure 9:

a) $\Omega^n(\omega_a, \alpha^-) > \Omega^n(\omega_b, \alpha^+)$; in this case, to apply Theorem 1 on each J_{b+n}, it is sufficient to take J_{a+n} having $\omega_{a+n} = \Omega^n(\omega_a, \alpha^-)$. Let then $\omega_{b+n} \in [\Omega^n(\omega_b, \alpha^-), \Omega^n(\omega_b, \alpha^+)]$. This choice makes the three hypotheses satisfied, since:
   - $\omega_{a+n} \geq \omega_{b+n}$ follows from $\Omega^n(\omega_a, \alpha^-) > \Omega^n(\omega_b, \alpha^+)$;
   - $\mathcal{C}(T(\omega_{a+n}, \alpha^-)) = \mathcal{C}(T(\omega_{b+n}, \alpha^-))$ follows from the hypothesis $\mathcal{C}(T^n(\omega_a, \alpha^-)) = \mathcal{C}(T^n(\omega_b, \alpha^-))$ $\forall n$. This can be shown by replacing the definition of the function $T^{n+1}(\omega, \alpha)$ obtaining $\mathcal{C}(T(\Omega^n(\omega_{a+n}, \alpha^-), \alpha^-)) = \mathcal{C}(T(\Omega^n(\omega_{b+n}, \alpha^-), \alpha^-))$. Since $\mathcal{C}()$ is a monotonic non-decreasing function, the hypothesis is verified for all ω_{b+n}.
   - J_{a+n} has the same mode as J_{b+n}: again this is ensured by considering the hypothesis $\mathcal{C}(T^n(\omega_a, \alpha^-)) = \mathcal{C}(T^n(\omega_b, \alpha^-))$ $\forall n$.

b) $\Omega^n(\omega_a, \alpha^-) \leq \Omega^n(\omega_b, \alpha^+)$; in this case, for each possible job J_{b+n} having $\omega_{b+n} \in [\Omega^n(\omega_a, \alpha^-), \Omega^n(\omega_b, \alpha^+)]$ there exists a job J_{a+n} having $\omega_{a+n} \in [\Omega^n(\omega_a, \alpha^-), \Omega^n(\omega_a, \alpha^+)]$, with $\omega_{a+n} = \omega_{b+n}$. This result implies directly $\mathcal{C}(T(\omega_{a+n}, \alpha^-)) = \mathcal{C}(T(\omega_{b+n}, \alpha^-))$. Since in this interval also the periods for J_{b+n} and J_{a+n} are overlapped, each J_{b+n} can be mapped to J_{a+n} having the same mode. The application of Theorem 1 is then straightforward.
   On the other hand, for each possible job J_{b+n} having $\omega_{b+n} \in [\Omega^n(\omega_b, \alpha^-), \Omega^n(\omega_a, \alpha^-)]$, the same considerations made for the upper case hold. Hence, it is sufficient to take J_{a+n} having $\omega_{a+n} = \Omega^n(\omega_a, \alpha^-)$.

Hence the theorem follows. □

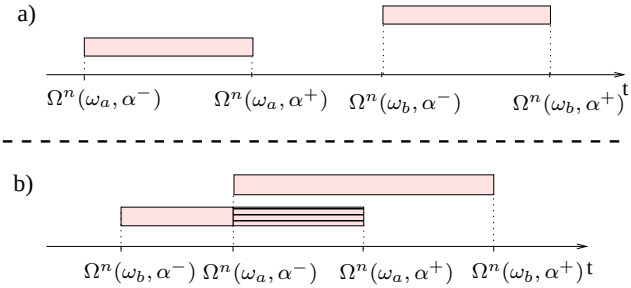

Figure 9: Possible scenarios for Theorem 2.

By using Theorem 2, it is possible to obtain the complete set $\mathcal{P}$ required by the pruning rule. To identify critical job releases belonging to $\mathcal{P}$ we proceed in the same way as for $\mathcal{P}^{(i)}$: the only difference is that Theorem 2 is applied in place of Theorem 1.

4.3. Dominant initial velocities

Theorem 2 can also be exploited to find a reduced set of initial instantaneous speeds that must be considered for the interference computation. Let ω_a and ω_b be two instantaneous speeds at which the critical instant may occur. If the hypotheses of Theorem 2 are verified, then we can conclude that $\forall t$ $I_{\omega_a}(t) \geq I_{\omega_b}(t)$, that is, the interference for ω_b is entirely dominated by the one for ω_a.

It follows that, given a range of possible engine speeds, we can compute a set of dominant speeds in this range in the very same way as we identified critical job releases applying the pruning rule in the range of speeds determined by α^- and α^+.

In the following,

- $\Lambda_i(\omega_1, \omega_2)$ denotes the set of dominant speeds for an AVR task τ_i^* when the engine speed can vary in the range $(\omega_1, \omega_2]$.
- $\Lambda_i^* = \Lambda_i(\omega_i^{min}, \omega_i^{max})$ denotes the set of dominant speeds of task τ_i^* for all possible speeds.
- Λ_i^m denotes the set of dominant speeds of task τ_i^* when the engine speed can vary in the range compatible with mode m; that is,

$$\Lambda_i^m = \Lambda_i(\widetilde{\Omega}(T^{m+1}), \widetilde{\Omega}(T^m)).$$

This is a key point to simplify the analysis with respect to the brute-force approach described in Section 3, and also to the ILP formulation proposed by Davis et al. [7], since both approaches require to consider the complete range of possible initial speeds.

5. Exact schedulability analysis

In this section we show how to use the interference of an AVR task τ_H^* (I_{H,ω_0}) to perform a response time analysis [1] of a generic task set consisting of AVR and periodic/sporadic tasks. We consider two cases to compute:

- the interference of an AVR task τ_H^* on a periodic/sporadic task τ_L with lower priority;
- the interference of an AVR task τ_H^* on another AVR task τ_L^* with lower priority.

When computing the interference of τ_H^* on a periodic/sporadic task or an AVR task with an independent source of activation events (independent ω_0), the worst-case combination for any initial speed of τ_H^* needs to be considered. To do so, the *envelope* $I_H(t)$ has to be computed as the maximum among the interference functions $I_{H,\omega_0}(t)$ for each initial speed. Using Theorem 2, it is sufficient to consider only the dominant speeds in Λ_H^*, hence

$$I_H(t) = \max_{\omega_0 \in \Lambda_H^*} \{I_{H,\omega_0}(t)\}. \tag{7}$$

To simplify the notation of the test, we also define the (finite) set $\mathcal{I}_H(t)$ of time points less than or equal to the argument t, in which the step function $I_H()$ changes its value.

5.1. AVR Interference on a periodic task

To analyze the interference of an AVR task τ_H^* on a lower priority periodic/sporadic task τ_L, every possible initial speed ω_0 of τ_H^* has to be considered at the beginning of the critical instant of τ_L. Therefore, the feasibility condition for τ_L becomes:

$$\forall \omega_0 \quad C_L + I_{H,\omega_0}(D_L) \leq D_L,$$

which is the same as

$$C_L + \max_{\omega_0}\{I_{H,\omega_0}(D_L)\} \leq D_L.$$

which, using Theorem 2 and Equation (7), reduces to

$$C_L + I_H(D_L) \leq D_L. \tag{8}$$

5.2. AVR Interference on an AVR task

When analyzing the interference of an AVR-task τ_H^* over another lower priority AVR task τ_L^*, two cases can be distinguished. Let ω_H and ω_L be the variables describing the speeds of the activation sources for τ_H^* and τ_L^*, respectively. As a first case we consider the two speeds to be independent, while in the second case we consider them related.

When rotation speeds are independent, the low-priority task τ_L^* is schedulable if and only if

$$\forall \omega_H \omega_L \quad \mathcal{C}_L(\widetilde{T}(\omega_L)) + I_{H,\omega_H}(D_L(\omega_L)) \leq D_L(\omega_L). \tag{9}$$

Since the WCET of τ_L^* depends on its execution mode, Equation (9) must be true for all modes. In addition, the full set of speeds of τ_H^* can be considered using Equation (7), thus τ_L^* is feasible if and only if $\forall m = 1, \ldots, M_L$ and $\forall \omega_L \in (\omega_L^{m+1}, \omega_L^m]$

$$C_L^m + I_H(D_L(\omega_L)) \leq D_L(\omega_L).$$

The previous formula should be evaluated for the full set of deadlines for mode m, denoted as $\mathcal{D}_L^m$:

$$\mathcal{D}_L^m = \{t \mid t \in (D_L(\omega_L^{m-1}), D_L(\omega_L^m)] \ \}.$$

Note however, that we do not need to consider an infinite number of points in $\mathcal{D}_L^m$, since $I_H(t)$ only changes its value in the finite set of time points $\mathcal{I}_H(\max\{\mathcal{D}_L^m\})$ (up to the maximum deadline for the mode). Therefore, τ_L^* can be feasibly scheduled if and only if $\forall m = 1, \ldots, M_L$ and $\forall t \in \mathcal{D}_L^m \cap \mathcal{I}_H(\max\{\mathcal{D}_L^m\})$

$$C_L^m + I_H(t) \leq t. \tag{10}$$

When ω_H and ω_L are related to a common speed ω_0, the previous analysis is pessimistic and needs to be refined. The response time analysis is a function of ω_0 and τ_L^* can be feasibly scheduled if and only if

$$\forall \omega_0 \quad \mathcal{C}_L(\widetilde{T_L}(\omega_0)) + I_{H,\omega_0}(D_L(\omega_0)) \leq D_L(\omega_0). \tag{11}$$

Considering each mode separately we have:

$$\forall m = 1, \ldots, M_L \quad \forall \omega_0 \in (\omega_L^{m+1}, \omega_L^m]$$
$$C_L^m + I_{H,\omega_0}(D_L(\omega_0)) \leq D_L(\omega_0).$$

To compute the interference of the high priority task, we consider the set of its dominant speeds within the speed range of every mode for τ_L^*. The range of speeds for each mode is partitioned in the intervals defined by the dominant speeds of τ_H^*. Let $(\omega^{d_{i-1}}, \omega^{d_i}]$ be the generic interval between two of such dominant speeds ($\omega^{d_j} \in \Lambda_H^*$), and let p_m be the number of such intervals for mode m. For each interval $(\omega^{d_{i-1}}, \omega^{d_i}]$, since the deadline is a non-increasing function of the speed, the shortest deadline for τ_L^* (replacing the term $D_L(\omega_0)$ in the formula) corresponds to the dominant ω^{d_i} at the highest end of the interval. ω^{d_i} also allows to upper bound the interference term $I_{H,\omega^{d_i}}(D_L(\omega^{d_i})) \geq I_{H,\omega_0}(D_L(\omega_0))$ for every $\omega_0 \in (\omega^{d_{i-1}}, \omega^{d_i}]$. Hence, the schedulability of τ_L^* can only be tested in p_m points:

$$\forall m = 1 \ldots M_L, \ \ i = 1 \ldots p_m$$
$$C_L^m + I_{H,\omega^{d_i}}(D_L(\omega^{d_i})) \leq D_L(\omega^{d_i}). \tag{12}$$

6. Evaluation

This section presents an evaluation of the proposed analysis method carried out on task data that are representative for an engine control system. The application was provided in the context of the INTERESTED EU project [8] and consists of a set of periodic and AVR tasks. One of these tasks is activated at the TDC mark and is characterized by 6 execution modes, reported in Table 1, and by a maximum period $T_{max} = 120$ ms (corresponding to 500 rpm).

mode	1	2	3	4	5	6
rpm	6500	5500	4500	3500	2500	1500
T^m (ms)	9.23	10.91	13.33	17.15	24	40
C^m (μs)	246	277	343	424	576	965

Table 1: Task parameters used in the evaluation.

Such a task is used to compare the accuracy and performance of the proposed approach with respect to the sufficient ILP-based method proposed by Davis et al. [7].

Before describing the results of the experiments, a few considerations are necessary to explain the terms of the comparison. The interference analysis is a function of two variables: the initial engine speed ω_0 and the length t of the interference window. The ILP method can only compute the interference for a set of discrete values of t and ω_0. Our analysis, thanks to the derivation of the dominant speeds, can actually compute the exact interference function for all the values t and ω_0 in their continuous ranges.

To run the experiments, the ILP method has been implemented on a CPLEX solver running on an 8-core Intel Xeon at 2.8 GHz, while our algorithm was implemented as MATLAB code on a dual-core laptop Intel i7 at 2.5 GHz. The run time of the two algorithms resulted to be in the order of few seconds to compute the maximum interference with the ILP algorithm, and about one minute to characterize the interference in the whole time interval with our algorithm.

Three experiments have been carried out. The first experiment is meant to compare the quality of the analysis in the domain of the initial speed ω_0. Figure 10 plots the interference $I_{\omega_0}(t)$ generated by the AVR task in a time interval of 100 ms for a set of initial engine speeds ω_0 between 1500 rpm and 6500 rpm and a maximum acceleration of 1.6210^{-4} rev/msec^2. As suggested by Davis at al. [7], a quantization step of 100 rpm was used to define the set of discrete values for ω_0. As shown in the figure, our analysis is able to achieve an improvement of 20 percent, or higher, for specific speeds, exhibiting an average improvement of about 10 percent.

In the ILP formulation, we used the set of linear constraints presented in [7], which does not include any bound on the minimum engine speed but only on the maximum one and the acceleration values. This allows the optimization engine to compute (incorrect) arbitrarily low values for the engine speed for low values of ω_0 and is the cause for the anomalous behavior shown in the figure for ω_0 below 1900rpm, where the interference computed using the ILP formulation falls below the exact analysis. Unfortunately, the addition of a lower speed bound requires updating several constraints in a non-trivial way.

In a second experiment, we evaluated the pessimism of the ILP method in the dimension of time, for an initial engine speed ω_0 = 5600 rpm. Figure 11 illustrates the interference function computed by the two algorithms when the interference interval is varied from 30 ms to 75 ms. In this time range, the pessimism of the ILP method remains of approximately 10 percent, and tends to increase for larger values of t.

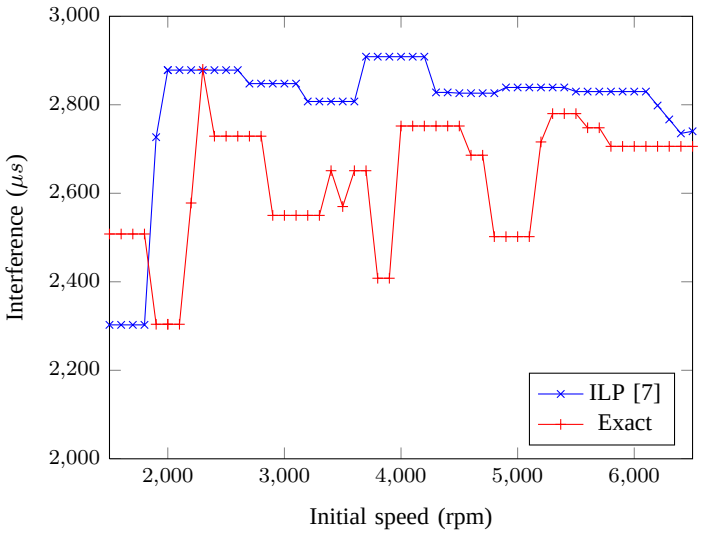

Figure 10: AVR Interference for different initial engine speeds, in a time interval of 100 ms.

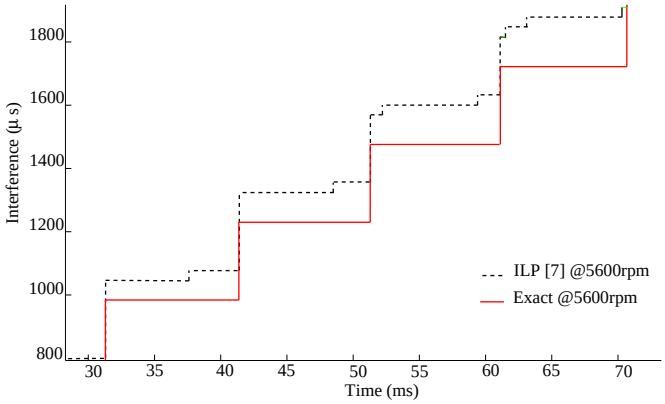

Figure 11: AVR Interference for different time intervals, with ω_0 = 5600 rpm.

A third experiment has been carried out to evaluate the possible consequences of the quantization process required by the ILP method for the values of ω_0. To do so, the ILP method was executed to compute the interference function using for two discrete values of ω_0, equal to 5500 rpm and 5600 rpm.

Figure 12 shows the plots of the interference derived by the ILP method together with the values computed by our algorithm for the intermediate initial speed of 5550 rpm. As highlighted in the figure, the interference computed by our exact algorithm at 5550 rpm is not dominated by the ILP interference function computed at 5500 rpm, nor by the one at 5600 rpm, despite the pessimism of the ILP approach.

This example makes an even stronger cautionary case against the use of any analysis based on discrete values, emphasizing the use of an accurate test that operates in a continuous domain.

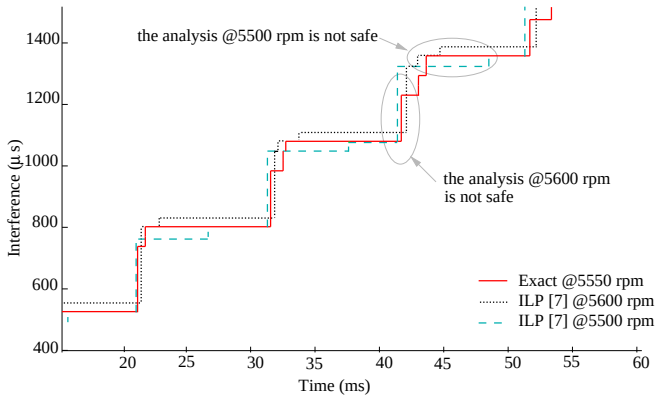

Figure 12: The analysis for a discrete set of ω_0 is not safe.

7. Conclusions

This paper presented an exact analysis of the worst-case interference generated by an adaptive variable-rate task under fixed-priority scheduling and arbitrary deadlines. The work advances the state of the art, since previous papers on fixed-priority AVR tasks either focused on special simplified cases [11] or proposed an approximate ILP test [7] based on a quantization of the state variable (i.e., the crankshaft rotation speed in engine control applications) determining the task activation rate.

The second contribution of this work has been the identification of a finite subset of dominant speeds that determine the exact worst-case interference of an AVR task. Thanks to this result, the interference analysis does not need to be performed for all possible values of the engine speed, with a given quantization, but can be determined by considering only the set of dominant speeds.

It is worth observing that the set of dominant speeds is not only important for reducing the complexity of the search, but also because it allows getting rid of the quantization of the state variable, which can make the schedulability analysis unsafe. In fact, by quantizing the state variable, the schedulability test is performed only in a subset of all possible values, so there can be points (not considered in the test) in which the schedulability test is not satisfied.

The simulation results reported in Section 6 confirm that the interference computed by the ILP approximated method in the discrete values of the initial speed ω_0 is always higher or equal to the interference computed by the approach proposed in this paper, in the same points. The results also indicate that, for intermediate values of the state variable, not considered in the ILP test, the worst-case interference can be higher than that computed by the ILP approach for both the lower and upper discrete value of ω_0, showing that no single interference curve computed by the ILP method is safe for analyzing intermediate points.

As a future work, we plan to extend the task model to consider more realistic design solutions considered in the automotive application domain, like AVR tasks with an initial phase and mode transitions with hysteresis.

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] S. K. Baruah, D. Chen, S. Gorinsky, and A. K. Mok. Generalized multiframe tasks. *Real-Time Systems*, 17(1):5–22, July 1999.

[3] G. Buttazzo, L. Abeni, and G. Lipari. Elastic task model for adaptive rate control. In *IEEE Real Time System Symposium*, Madrid, Spain, December 1998.

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

[5] G. Buttazzo, G. Lipari, M. Caccamo, and L. Abeni. Elastic scheduling for flexible workload management. *IEEE Transactions on Computers*, 51(3):289–302, March 2002.

[6] D. Buttle. Real-time in the prime-time. Keynote speech given at the 24th Euromicro Conference on Real-Time Systems (ECRTS 2012), Pisa, Italy, July 12th, 2012.

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

[8] INTERESTED project partners. Interested, european project, cordis project description. URL: http://cordis.europa.eu/fp7/ict/embedded-systems-engineering/factsheets/interested.pdf, 2008.

[9] K. Jeffay and D. Bennett. A rate-based execution abstraction for multimedia computing. In *Proceedings of the Fifth International Workshop on Network and Operating System Support for Digital Audio and Video, Lecture Notes in Computer Science, Springer-Verlag, Vol. 1018*, pages 67–78, Durham, NH, April 1995.

[10] K. Jeffay and S. Goddard. A theory of rate-based execution. In *Proceedings of the 20th IEEE Real-Time Systems Symposium*, pages 304–314, Phoenix, AZ, December 1999.

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

[12] C. Liu and J. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the Association for Computing Machinery*, 20(1):46–61, January 1973.

[13] A. Mok. *Fundamental Design Problems of Distributed Systems for the Hard Real-Time Environment*. PhD thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, Cambridge, MA, 1983.

[14] A. K. Mok and D. Chen. A multiframe model for real-time tasks. *IEEE Transactions on Software Engineering*, 23(10):635–645, October 1997.

[15] 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.

[16] J. Real and A. Crespo. Mode change protocols for real-time systems: A survey and a new proposal. *Real-Time Systems*, 26(2):161–197, March 2004.

[17] L. Sha, R. Rajkumar, J. P. Lehoczky, and K. Ramamritham. Mode change protocols for priority-driven preemptive scheduling. *Real-Time Systems*, 1(3):243–264, December 1989.

[18] N. Stoimenov, S. Perathoner, and L. Thiele. Reliable mode changes in real-time systems with fixed priority or EDF scheduling. In *Proceedings of the Design, Automation and Test Conference in Europe (DATE 2009)*, Nice, France, April 20-24, 2009.