Refinement of Workload Models for Engine Controllers by State Space Partitioning

Morteza Mohaqeqi[1], Jakaria Abdullah[2], Pontus Ekberg[3], and Wang Yi[4]

1 **Uppsala University, Uppsala, Sweden**
`morteza.mohaqeqi@it.uu.se`
2 **Uppsala University, Uppsala, Sweden**
`jakaria.abdullah@it.uu.se`
3 **Uppsala University, Uppsala, Sweden**
`pontus.ekberg@it.uu.se`
4 **Uppsala University, Uppsala, Sweden**
`yi@it.uu.se`

Abstract

We study an engine control application where the behavior of engine controllers depends on the engine's rotational speed. For efficient and precise timing analysis, we use the Digraph Real-Time (DRT) task model to specify the workload of control tasks where we employ optimal control theory to faithfully calculate the respective minimum inter-release times. We show how DRT models can be refined by finer grained partitioning of the state space of the engine up to a model which enables an exact timing analysis. Compared to previously proposed methods which are either unsafe or pessimistic, our work provides both abstract and tight characterizations of the corresponding workload.

1998 ACM Subject Classification C.3 Real-Time and Embedded Systems

Keywords and phrases engine control tasks, schedulability analysis, minimum-time problem, DRT task model

Digital Object Identifier 10.4230/LIPIcs.ECRTS.2017.11

1 Introduction

Cyber-physical systems are recognized by a tight integration of a physical process, which is subject to dynamic changes, and a computing system, which is responsible for handling the events initiated by the physical environment. From a real-time scheduling perspective, an event triggers a task to release a new job which must be completed within a specified time. Since the events are released according to the state of physical variables, and as the physical system is subject to dynamic variations, there may exist no regular or completely predictable release pattern for the jobs. In order to precisely specify the real-time workload in the mentioned systems, which is necessary for an accurate timing (and schedulability) analysis, one needs to (i) properly characterize the dynamic behavior of the physical system; and, (ii) construct reliable models (i.e., task systems) to capture the respective workload.

A key parameter in real-time task models is the minimum inter-release time between the jobs that are released by a task. In cyber-physical systems, such as an engine control application [11], this parameter can change at run time based on the physical system dynamics (i.e., engine speed in this case). Therefore, in order to accurately specify minimum inter-release times for each task, the relation between system dynamics and the release pattern of the corresponding events must be exploited in a rigorous way.

In this work, we employ a mathematical model of the physical system which is described through a set of differential equations. We show how the parameters of the corresponding real-time tasks, particularly minimum inter-release times, can be extracted in a faithful manner with a tunable accuracy. For this purpose, we formulate the problem of finding minimum inter-release times as an optimization problem. Then, we use the *calculus of variations* [13] from optimal control theory to calculate the minimum time required by the physical system to transfer from an initial state to a target state. The solution is used to derive the minimum inter-release time of the respective jobs (as the release of a job depends on the physical system's state). Based on the obtained values, a digraph real-time (DRT) [17] task is constructed as a representation of the engine control task. An important benefit of using the DRT task model is that, while it is expressive enough to capture the varying behavior of engine controllers, there exist efficient timing analysis methods for this task model for both static and dynamic priority scheduling policies [20, 22]. Also, using the theoretical framework of calculus of variations provides a "generic" solution which can be adopted for systems with more complicated dynamical behavior.

As it will be shown, construction of a DRT task for an engine controller requires partitioning of the problem state space. In this work, we also propose a method for partitioning the continuous range of the engine speed such that the resultant DRT task contains no pessimism with respect to schedulability analysis. As a result, our method provides a tight workload characterization of engine control tasks. To be consistent with the literature [5, 10, 4, 2], hereafter engine control tasks are referred to as AVR (adaptive variable-rate) tasks.

The rest of this paper is organized as follows. Previous work on timing analysis of AVR tasks is reviewed in the rest of this section. Section 2 introduces the system model considered in the current work, as well as a number of basic relations. Our method for constructing a DRT model for an AVR task is described in Section 3. Section 4 gives an illustrative example of how the proposed method works on a sample AVR task with realistic engine parameters. The faithfulness of the proposed method is formally established in Section 5. We present a method to construct the DRT task such that it provides an exact schedulability analysis of the AVR task in Section 6. The proposed method is evaluated in Section 7. A summary of the paper as well as some remarks for extending the work is presented in Section 8.

Related work: Real-time tasks with continually variable parameters were studied by Kim et al. [12] by introducing the *rhythmic* task model and proposing a respective response-time analysis method for rate monotonic scheduling. The method, however, is restricted to task sets with only one rhythmic task, which must be of the highest priority. Pollex et al. [16, 15] relaxed this constraint by assuming a task system containing an arbitrary number of so-called engine-triggered tasks. They proposed a sufficient schedulability test for such task sets in [16] under the assumption of an arbitrary but constant angular speed. Later, in [15], they removed the limitation to constant speeds, presenting a pessimistic analysis for the worst-case response time of engine-triggered tasks. The accuracy of the analysis was improved in [9] by taking the dependency of the tasks into account. The method also permits the execution time of the jobs to be specified as a *continuous* function of the rotational speed. In contrast to the analyses in [16, 15, 9], which present sufficient/pessimistic real-time analysis, our method provides both necessary and sufficient conditions for schedulability.

A major problem with characterizing the workload of an AVR task is infiniteness of the problem state space. This is because physical variables, such as the rotational speed, can vary continuously, and the fact that task parameters depend on such variables. Davis et al. [8] used state space quantization to address this issue, and then, presented a sufficient

schedulability test using Integer Linear Programming (ILP). However, the method only provides a sufficient test, containing additional pessimism due to the quantization.

Focusing on fixed priority schedulability of AVR tasks, Biondi et al. [5] overcame the infinite state space problem by recognizing a set of finite cases (for the speed at the instant of a job release) which *dominate* the whole state space. Based on this, schedulability analysis is reduced to a search problem in the space of all possible dominant cases. The same approach was used in [6] to derive the worst-case response time of a set of mixed periodic and AVR tasks under fixed priority scheduling. The approach was further employed to develop an EDF schedulability test for the respective tasks in [4]. In this case, dominant speeds are used to calculate the demand bound function (dbf) based on which feasibility analysis is accomplished. The analysis is proposed for those AVR tasks that depend on a single rotation source, and have a zero angular phase and the same period. In [4], it is also proposed to use a digraph structure, i.e., DRT graphs, for modeling and analysis of AVR tasks. A crucial restriction of the analyses presented in [5, 6, 4] is that they are valid only when the following assumption holds: between the release of two consecutive jobs of a task, the angular acceleration is constant. However, in the general case where the acceleration may vary between the release of two jobs, the proposed methods are not necessarily valid. We illustrate this issue in more details in Sections 4 and 7 by providing concrete counter examples.

A model for describing AVR tasks that are implemented with a hysteresis during mode switching has been proposed in [2]. Also, the dependency of the tasks on a common rotation source is taken into account. The work is more general than our method in terms of considering these realistic aspects. However, this generality comes at the cost of inaccuracy as they present only an upper bound on the system utilization, which is used for schedulability tests. The analysis is also restricted to the EDF scheduling algorithm. In a recent work, Biondi and Buttazzo [3] considered the speed estimation problem and its impact on schedulability analysis of AVR tasks. They argue that in practice, there is not a precise knowledge about the instantaneous speed of the engine. Based on this observation, they study the error that is introduced by state estimation into the analysis. Meanwhile, they do not focus on providing a new timing analysis method.

An initial idea of using the DRT task model for feasibility analysis of AVR tasks was also proposed by Guo and Baruah [10]. Compared to our work, the method in [10] considers a pessimistic assumption for extracting the inter-release times. More precisely, when switching from a mode to another one that corresponds to a higher speed range, the method assumes that the engine rotates with the maximum acceleration irrespective of whether this acceleration will lead to the designated speed range or not. Hence, the method provides a pessimistic lower bound on the minimum inter-release time which, as will be shown in Section 4, can be tightened. A similar assumption was used by Feld and Slomka [9] for response-time analysis of AVR tasks under fixed priority scheduling, where the maximum acceleration is used to compute the interfering workload of higher priority tasks in a given interval. As a result, the approach suffers from the same pessimism in timing analysis.

2 System model and preliminaries

This section presents the model used in the current study to describe the engine's dynamics. AVR and DRT task models are also briefly reviewed.

2.1 Dynamical system

In order to specify the state of the dynamical system and its evolution, three variables are defined: $\theta(t) \doteq$ angular position of the crankshaft, $\omega(t) \doteq$ angular velocity (rotational speed), and $\alpha(t) \doteq$ angular acceleration, where t denotes the physical time. These variables are related to each other through the following equations [9]:

$$\omega(t) = \dot{\theta} = \frac{d\theta}{dt}, \tag{1}$$

$$\alpha(t) = \dot{\omega} = \frac{d\omega}{dt}. \tag{2}$$

The minimum and maximum possible accelerations are denoted by α^- and α^+, respectively. Further, the rotational speed can vary in the range $[\omega_{min}, \omega_{max})$.

Under a fixed acceleration, θ and ω can be expressed as explicit functions of time. Towards this, assume a constant acceleration α during an interval of $[0, t_e]$. Then, from (1) and (2), the angular position and speed at any time instant $t \in [0, t_e]$ can be computed by

$$\theta(t) = \frac{1}{2}\alpha t^2 + \omega_0 t + \theta_0, \tag{3}$$

$$\omega(t) = \omega_0 + \alpha t, \tag{4}$$

where θ_0 and ω_0 denote the initial angle and the initial speed, respectively. From these equations, the relation between θ and ω under a constant acceleration α can be specified as

$$\theta(\omega) = \frac{1}{2\alpha}\left(\omega^2 - \omega_0^2\right) + \theta_0. \tag{5}$$

Assuming an initial speed ω_0 and a constant acceleration of α, the time it takes to have an angular change of $\Delta\theta$ can be calculated, as shown in [7], by

$$T(\omega_0, \alpha, \Delta\theta) = \frac{\sqrt{\omega_0^2 + 2\alpha\Delta\theta} - \omega_0}{\alpha}. \tag{6}$$

Also, the speed after an angular change of $\Delta\theta$, denoted by $\Omega(\omega_0, \alpha, \Delta\theta)$, is derived as, [5]:

$$\Omega(\omega_0, \alpha, \Delta\theta) = \sqrt{\omega_0^2 + 2\alpha\Delta\theta}. \tag{7}$$

According to the described relations, we point out two properties assuming the domain of non-negative values for ω_0 and $\Delta\theta$.

▶ **Property 1.** *Function $T(\omega_0, \alpha, \Delta\theta)$ defined in (6) is (strictly) decreasing in ω_0 and α.*

▶ **Property 2.** *Function $\Omega(\omega_0, \alpha, \Delta\theta)$ defined in (7) is (strictly) increasing in ω_0 and α.*

Throughout this work, we assume that $\Delta\theta$ specifies the crankshaft revolution in terms of "number of rotations". As a result, a full rotation is recognized by $\Delta\theta = 1$. Hence, according to (7), and assuming an initial speed of ω_0 and a constant acceleration of α, the speed after one complete revolution, denoted by $\Omega_1(\omega_0, \alpha)$, would be $\Omega_1(\omega_0, \alpha) = \sqrt{\omega_0^2 + 2\alpha}$. As a generalization, by $\Omega_i(\omega_0, \alpha)$ we denote the speed after exactly i rotations, for $i \in \mathbb{N}$, which can be calculated (under the same assumptions) as

$$\Omega_i(\omega_0, \alpha) = \sqrt{\omega_0^2 + 2i\alpha}. \tag{8}$$

2.2 AVR tasks

We consider the AVR task model that is commonly used in the literature (e.g., [5, 2, 4, 10]). Based on this model, an AVR task releases a new job whenever the crankshaft of the engine reaches a specific angle. Without loss of generality, we assume that the jobs are released whenever one full rotation is taken (i.e., $\Delta\theta = 1$).[1] The functionality of an AVR task is determined according to the instantaneous rotational speed. More specifically, a task comprises a set of M *modes*, each of which is related to a certain speed range. At run time, according to the instantaneous speed, a mode is selected and the corresponding functionality is executed. Based on this, each mode is specified by a pair $(C_m, [\omega_m, \omega_{m+1}))$ which associates a worst-case execution time (WCET) C_m and a speed range $[\omega_m, \omega_{m+1})$ to the mode. Consequently, the set of modes associated to a task is described as $\mathcal{M} = \{(C_m, [\omega_m, \omega_{m+1})) \mid m = 1, 2, \ldots, M\}$, where $\omega_1 = \omega_{min}$ and $\omega_{M+1} = \omega_{max}$.

2.3 The DRT task model

Our goal is to abstract the behavior of engine control applications such that, while having an accurate characterization, one can achieve efficient timing analysis. For this purpose, we use the DRT task model, which provides the facility of modeling real-time tasks with variable (mode switching) behavior, as described below.

A DRT task T is specified by a directed graph $G(T) = (V, E)$, where V and E denote the set of vertices and edges, respectively. Each vertex $v \in V$ corresponds to a specific *job type*, labeled by a pair of non-negative integers $\langle e(v), d(v)\rangle$, where $e(v)$ and $d(v)$ represent the WCET and relative deadline of the respective jobs. Each edge $(v, u) \in E$ is also labeled by a non-negative integer $p(v, u)$ which shows the minimum inter-release time between any job of v and any job of u. In such a graph, the existence of multiple outgoing edges from one vertex represents a notion of non-determinism in the type of the next job. Hence, each path in the graph is related to a possible sequence of jobs which can be released by the task. While this real-time task model provides a high degree of expressiveness, there exists quite efficient methods for feasibility [17], schedulability [19], and response-time [18] analysis of DRT task sets. In subsequent sections, we use the terms DRT task and DRT graph interchangeably.

3 AVR to DRT transformation

For modeling an AVR task as a DRT graph, two problems need to be addressed: (i) partitioning the speed range (i.e., $[\omega_{min}, \omega_{max})$) into an appropriate set of sub-intervals (each of which will be mapped to a DRT vertex); and (ii) constructing a DRT graph (and deriving its parameter values) based on these sub-intervals. For a better presentation, we first tackle the second one in this section, where we show how one can obtain a DRT graph for an AVR task assuming a *given* speed range partitioning. In Section 6, we deal with the first issue, and present a method to partition the speed range such that schedulability analysis of the obtained DRT graph provides an exact schedulability test for the original AVR task. We will refer to such a DRT graph as a tight model for the AVR task.

[1] To see how the general case can be considered in the system model, see, for example, [4].

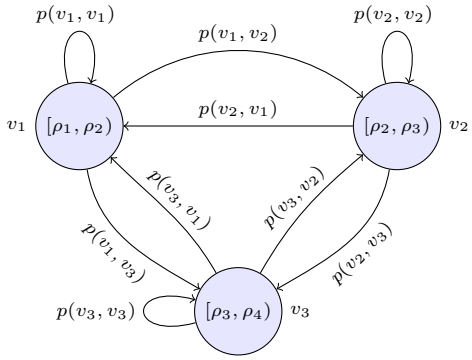


Figure 1 A sample DRT task with three vertices.

3.1 Transformation method

The idea is to capture the continuous state space of the engine speed, which determines the AVR task functionality, by a number of K vertices (job types) of a DRT graph. For this purpose, we assume a partition of the speed range $[\omega_{min}, \omega_{max})$ into a set of K sub-intervals $\{[\rho_1, \rho_2), \ldots, [\rho_K, \rho_{K+1})\}$, where $\rho_1 = \omega_{min}$ and $\rho_{K+1} = \omega_{max}$. Each sub-interval will be mapped a vertex of the DRT task. A trivial partitioning can be obtained by the speed ranges linked to the modes of the considered AVR task. However, any other partitioning can be used as well, leading to different levels of precision of the obtained DRT task, and thus, the respective timing analyses.

Figure 1 demonstrates a sample DRT task where each vertex is associated with a speed range. Based on this model, when the speed at the job release time is a value in the range of $[\rho_i, \rho_{i+1})$, for $i \in \{1, 2, 3\}$, a job of type v_i is released.

After the set of vertices are obtained, the label of each edge $e = (v, u)$, namely the minimum inter-release time, is determined based on the following observation. A job of u can be released whenever an entire rotation is taken by the crankshaft after the release of a job of type v. As a result, the minimum inter-release time is equal to the minimum time that it takes for the crankshaft to have a full rotation. The initial speed may be any value in the range associated to vertex v. In turn, the final speed (i.e., the speed exactly at the moment when the rotation completes) can be any value in the speed range of vertex u.

Based on these descriptions, we construct a DRT graph for a given AVR task. The procedure is shown in Algorithm 1. In Line 6 of Algorithm 1, the WCET of each vertex is determined by the maximum execution time of those modes whose speed range has an overlap with the speed range associated to that vertex. Next, the edge labels are specified (Lines 10 to 16). As seen, the procedure depends on calculating the minimum inter-release times by calling MinTime(.). The procedure MinTime(.) gets two speed ranges and uses a method from the optimal control theory to calculate the minimum possible time required for one rotation starting from a speed in the first range and ending in a speed in the second range. A formal description of the problem of finding the minimum time under these constraints and its solution are presented in the next subsection.

In order to determine the relative deadline of job types, we assume that each job needs to be finished before the release of the next job. As a result, for each job type, the minimum inter-release time among all outgoing edges from the respective vertex is considered as its relative deadline (Line 19 in Algorithm 1).

Algorithm 1 Deriving a DRT graph for an AVR task

input: A set $\mathcal{M} = \{(C_m, [\omega_m, \omega_{m+1})) \mid m = 1, \ldots, M\}$ of AVR task modes; Speed intervals $\{[\rho_1, \rho_2), \ldots, [\rho_K, \rho_{K+1})\}$.
output: A DRT graph.

procedure ConstructDRT
  $V \leftarrow \{\}, E \leftarrow \{\}$
  ▷ Creating the vertices:
  for $i \leftarrow 1$ **to** K **do**
    $v_i \leftarrow$ A new vertex
    $e(v_i) \leftarrow \max_{1 \le j \le M} \{C_j \mid [\rho_i, \rho_{i+1}) \cap [\omega_j, \omega_{j+1}) \neq \emptyset\}$
    $V \leftarrow V \cup \{v_i\}$
  end for
  ▷ Creating the edges:
  for all $v_i, v_j \in V$ **do**
    $p \leftarrow$ MinTime($[\rho_i, \rho_{i+1}), [\rho_j, \rho_{j+1})$)
    if $p \neq \infty$ **then**
      $p(v_i, v_j) \leftarrow p$
      $E \leftarrow E \cup \{(v_i, v_j)\}$
    end if
  end for
  ▷ Setting relative deadlines:
  for $i \leftarrow 1$ **to** K **do**
    $d(v_i) \leftarrow \min \{p(v_i, v_j) \mid (v_i, v_j) \in E\}$
  end for
  $G \leftarrow (V, E)$
  return G
end procedure

3.2 Computing minimum inter-release times

The goal is to calculate minimum inter-release times in the obtained DRT graph. For this, we first express the problem of finding minimum inter-release times as a minimization problem.

▶ **Problem 1** (Minimum-time problem). *Consider a dynamical system described by Eqs.* (1) *and* (2). *Also, assume that* $[\rho_i, \rho_{i+1})$ *and* $[\rho_f, \rho_{f+1})$ *are two given speed intervals. The problem is to find the minimum time* $t^* > 0$ *for which* $\theta(t^*) = 1$ *(which specifies one complete rotation), subject to* $\theta(0) = 0$, $\omega(0) \in [\rho_i, \rho_{i+1})$, $\omega(t^*) \in [\rho_f, \rho_{f+1})$, *and*

$$\omega_{min} \le \omega(t) \le \omega_{max}, \qquad \forall t \in [0, t^*], \tag{9a}$$

$$\alpha^- \le \alpha(t) \le \alpha^+, \qquad \forall t \in [0, t^*]. \tag{9b}$$

In words, the constraints express that we start at the angular position 0 *with a speed in interval* $[\rho_i, \rho_{i+1})$ *and want to end up in speed interval* $[\rho_f, \rho_{f+1})$ *after one full rotation.*

In the context of optimal control of dynamical systems, this problem is regarded as a *minimum-time* control problem [13]. Stating Problem 1 in that context, we can think of the acceleration function $\alpha(t)$ as the control input. Then, the problem is to find the optimal control (namely, the function $\alpha(t)$) which reveals the minimum time, while the initial and terminal states are constrained. The so-called *Pontryagin's minimum principle* [13] provides necessary conditions for an optimal solution to this problem. The principle is based on the notion of Hamiltonian function, reviewed below.

▶ **Definition 1** (Hamiltonian, [13]). Consider the dynamical system and the optimization problem specified in Problem 1. Further, let ψ_1 and ψ_2 denote two functions (of t). Then, the Hamiltonian is defined as[2]

$$H(\psi_1, \psi_2, \theta, \omega, \alpha) = \psi_1 \omega + \psi_2 \alpha + 1. \tag{10}$$

▶ **Theorem 2** (Pontryagin's minimum principle, [13]). *Let $\alpha^*(.)$ be an acceleration function which reveals the optimal solution of Problem 1. Then, functions ψ_1 and ψ_2 exist which satisfy*

$$\frac{d\psi_1}{dt} = \frac{\partial H}{\partial \theta}, \text{ and } \frac{d\psi_2}{dt} = \frac{\partial H}{\partial \omega}, \tag{11}$$

and

$$H(\psi_1, \psi_2, \theta, \omega, \alpha^*) \leq H(\psi_1, \psi_2, \theta, \omega, \alpha) \tag{12}$$

for all admissible controls $\alpha(.)$ and all times $t \in [t_0, t_f]$.

The theorem expresses that the optimal control α^* for the original problem must minimize the Hamiltonian as well. Thus, instead of solving Problem 1 directly, we focus on finding the control function that minimizes the Hamiltonian, which is typically an easier problem.

The necessary condition stated by the theorem provides a way to characterize the optimal control. Specifically, from (11) in Theorem 2, and by taking the derivative of the Hamiltonian, i.e., $H()$ defined in (10), with respect to θ and ω, we reach $\dot{\psi_1}(t) = 0$ and $\dot{\psi_2}(t) = -\psi_1$, which yield

$$\begin{aligned} \psi_1(t) &= c_1, \\ \psi_2(t) &= -c_1 t + c_2, \end{aligned} \tag{13}$$

for two (unknown) constants c_1 and c_2. By replacing $\psi_1(t)$ and $\psi_2(t)$ in (10) with their definition in (13), the Hamiltonian is obtained as

$$H(\psi_1, \psi_2, \theta, \omega, \alpha) = c_1 x_2 + (-c_1 t + c_2)\, \alpha + 1. \tag{14}$$

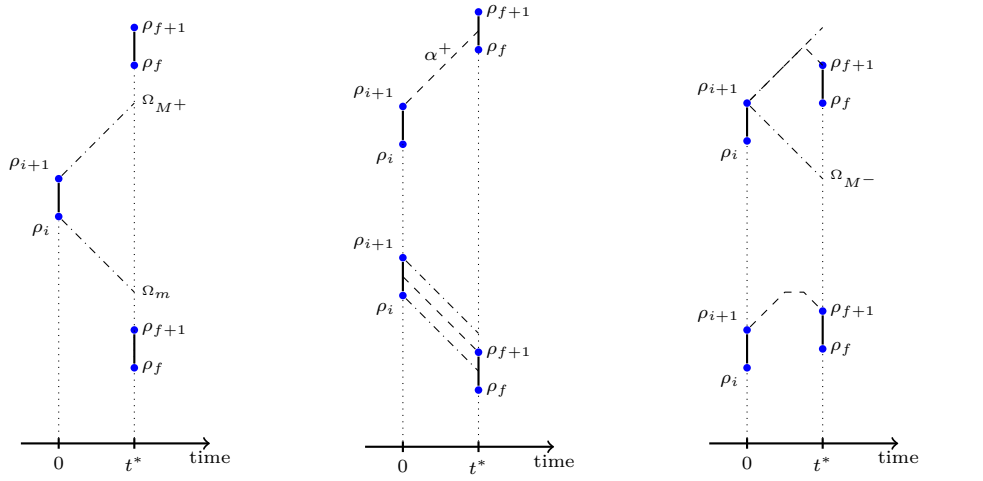
According to the Pontryagin's minimum principle (Theorem 2), a necessary condition for a control variable α to be an optimal solution to Problem 1 is that it minimizes the Hamiltonian. Based on (14), the acceleration function which minimizes the Hamiltonian is determined by

$$\alpha^*(t) = \begin{cases} \alpha^+, & \text{if } -c_1 t + c_2 < 0, \\ \alpha^-, & \text{if } -c_1 t + c_2 > 0. \end{cases} \tag{15}$$

Consequently, as also shown in [13, 21] for a similar setting, the optimal control function, i.e., the engine acceleration which minimizes the time, should be

$$\alpha^*(t) = \begin{cases} \alpha^+, & \text{for all } t \in [0, t^*], \text{ or} \\ \alpha^-, & \text{for all } t \in [0, t^*], \text{ or} \\ \alpha^+, & \text{for } t \in [0, t_1) \text{ and } \alpha^-, \text{ for } t \in [t_1, t^*], \text{ or} \\ \alpha^-, & \text{for } t \in [0, t_1) \text{ and } \alpha^+, \text{ for } t \in [t_1, t^*]. \end{cases} \tag{16}$$

[2] The Hamiltonian does not directly depend on θ in our problem. However, to follow the general form, we include θ in the arguments list. For a generic definition of the Hamiltonian see [13].



(a) Case 1 (infeasible) **(b)** Case 2,3 (fixed accel.) **(c)** Case 4 (variable accel.)

Figure 2 Possible cases for minimum time revolution. Dashed lines indicate speed variation in the optimal solution; dot-dashed lines show speed evolution under an extreme (minimum or maximum) acceleration; t^* stands for the instant of one full rotation.

(as long as the system constraints, i.e., lower and upper speed and acceleration bounds, are not violated.) In order to determine which case in (16) provides the optimal solution, we need a secondary condition. For this, we notice that, in the beginning, the system must start with its maximum possible speed that can lead to a feasible solution. The reason is as follows. Assume an optimal solution where the system does not start from the maximum speed in the permitted range. This means that there exists a higher speed at which the system can start and yet provide a feasible solution. Based on Property 1, this solution provides a smaller time, or equivalently, a better solution compared to the presumed optimal one, which is a contradiction. This observation, which is referred to as the *transversality condition* [21], is used to obtain the actual optimal solutions, as described in the following.

The optimal solution of Problem 1 is determined according to the value of the problem parameters, i.e., α^-, α^+, ρ_i, ρ_{i+1}, ρ_f, and ρ_{f+1}. More particularly, with respect to the existence of a solution, and also the criteria specified by (16) for computing the optimal result, if it exists, different cases can be identified. To introduce and deal with these cases, we first define the following notations (using Eq. (8))

$$\Omega_m \doteq \Omega_1(\rho_i, \alpha^-) = \sqrt{\rho_i^2 + 2\alpha^-} \tag{17a}$$

$$\Omega_{M^-} \doteq \Omega_1(\rho_{i+1}, \alpha^-) = \sqrt{\rho_{i+1}^2 + 2\alpha^-} \tag{17b}$$

$$\Omega_{M^+} \doteq \Omega_1(\rho_{i+1}, \alpha^+) = \sqrt{\rho_{i+1}^2 + 2\alpha^+} \tag{17c}$$

Figure 2 depicts all possible situations regarding an optimal solution. We formally specify and treat each case as follows.

Case 1: $\Omega_{M^+} \leq \rho_f$ or $\Omega_m \geq \rho_{f+1}$ (Fig. 2a). In this case, the problem has no feasible solution (this is implied by Property 2 and the continuity of $\Omega(.,.,.)$ defined in (7)).

Case 2: $\rho_f < \Omega_{M^+} \leq \rho_{f+1}$ (depicted in the upper part of Fig. 2b). Under this condition, selecting an initial speed of ρ_{i+1} and assigning $\alpha = \alpha^+$, $\forall t \in [0, t^*]$ reveal the minimum time. Therefore, the duration of one rotation can be calculated through Eq. (6).

Case 3: $\Omega_m < \rho_{f+1} \leq \Omega_{M^-}$ (seen in the lower part of Fig 2b). In this case, the optimal time is achieved when the acceleration is selected to be α^- and the initial speed is selected such that the final speed is ρ_{f+1}. In other words, from (8), the initial speed (denoted by ρ_0) must satisfy $\rho_{f+1} = \sqrt{\rho_0^2 + 2\alpha^-}$. This yields

$$\rho_0 = \sqrt{\rho_{f+1}^2 - 2\alpha^-}. \tag{18}$$

Again, the duration of one rotation can be calculated by Eq. (6) with assigning $\omega_0 = \rho_0$, $\alpha = \alpha^-$, and $\Delta\theta = 1$.

Case 4: $\Omega_{M^-} \leq \rho_{f+1} \leq \Omega_{M^+}$ (seen in Fig 2c). In this situation, in the optimal solution, the initial and final speed will be the maximum possible values, namely ρ_{i+1} and ρ_{f+1}, respectively. Also, the acceleration will be chosen to be α^+ until a certain instant, denoted by t_1, and then it will be switched to α^- (due to the transversality condition [21]). To obtain the minimum time, we use the result of the following lemma.

▶ **Lemma 3.** *Let ρ_1 denote the rotational speed at time t_1 defined in case 4. Then, ρ_1 can be computed as*

$$\rho_1 = \sqrt{\frac{2\alpha^-\alpha^+ + \alpha^-\rho_{i+1}^2 - \alpha^+\rho_{f+1}^2}{\alpha^- - \alpha^+}}. \tag{19}$$

Proof. Let θ_1 represent the position at t_1. From (5) it follows that $\theta_1 = \frac{1}{2\alpha^+}(\rho_1^2 - \rho_{i+1}^2)$. Also, as the final position is assumed to be 1, we can write $1 = \frac{1}{2\alpha^-}(\rho_{f+1}^2 - \rho_1^2) + \theta_1$. Substituting θ_1 from the former equation to the latter one reveals $1 = \frac{\rho_{f+1}^2}{2\alpha^-} - \frac{\rho_{i+1}^2}{2\alpha^+} + \rho_1^2(\frac{1}{2\alpha^+} - \frac{1}{2\alpha^-})$. Solving this equation for ρ_1 gives (19). ◀

Two different cases may arise according to the value of ρ_1 calculated in Lemma 3. If the speed limit is not violated, i.e., if $\rho_1 \leq \omega_{max}$, then the minimum time for one rotation, denoted by t^*, can be computed as (based on (4))

$$t^* = \frac{\rho_1 - \rho_{i+1}}{\alpha^+} + \frac{\rho_{f+1} - \rho_1}{\alpha^-}. \tag{20}$$

On the other hand, if $\rho_1 > \omega_{max}$, then after reaching the maximum speed, the acceleration is forced to be 0 for a while and then, turning to α^-. Under this scenario, we have $t^* = t_1^* + t_2^* + t_3^*$, where t_1^*, t_2^*, and t_3^* denote the time durations in which the acceleration is α^+, 0, and α^-, respectively, and are calculated as

$$\begin{aligned} t_1^* &= (\omega_{max} - \rho_{i+1})/\alpha^+, \\ t_2^* &= \frac{1}{\omega_{max}}\left(1 - \frac{\omega_{max}^2 - \rho_{i+1}^2}{2\alpha^+} - \frac{\rho_{f+1}^2 - {\omega_{max}}^2}{2\alpha^-}\right), \\ t_3^* &= (\rho_{f+1} - \omega_{max})/\alpha^-. \end{aligned} \tag{21}$$

Algorithm 2 summarizes the described procedure for solving Problem 1. As seen, according to each case, a respective value is calculated, and finally returned.

▶ **Lemma 4.** *Consider two vertices v_i and v_f in the constructed DRT graph associated with speed intervals $[\rho_i, \rho_{i+1})$ and $[\rho_f, \rho_{f+1})$, respectively. Additionally, assume two jobs J_1 and J_2 released by the AVR task such that the rotational speed at the release time of J_1 lies in $[\rho_i, \rho_{i+1})$, and at the release time of J_2 lies in $[\rho_f, \rho_{f+1})$. Then, the inter-release time of J_1 and J_2 is not smaller than $p(v_i, v_f)$.*

Proof. The Lemma follows from the optimality (minimality) of the solution provided by Algorithm 2 for Problem 1. ◀

Algorithm 2 Computing minimum time

input: $[\rho_i, \rho_{i+1})$: Initial speed range; $[\rho_f, \rho_{f+1})$: Final speed range.
output: p: A lower bound on the duration of one rotation.

1: **procedure** MinTime($[\rho_i, \rho_{i+1})$, $[\rho_f, \rho_{f+1})$)
2: Take Ω_m, Ω_{M^-}, and Ω_{M^+} as defined in (17).
3: **if** $\rho_f \geq \Omega_{M^+}$ **or** $\rho_{f+1} \leq \Omega_m$ **then**
4: $p \leftarrow \infty$
5: **else if** $\Omega_{M^+} \in (\rho_f, \rho_{f+1}]$ **then**
6: $p \leftarrow T(\rho_{i+1}, \alpha^+)$ ▷ Using (6)
7: **else if** $\rho_{f+1} \in (\Omega_m, \Omega_{M^-}]$ **then**
8: $p \leftarrow T(\rho_0, \alpha^-)$ ▷ ρ_0 is computed in (18)
9: **else**
10: Take ρ_1 as in (19).
11: **if** $\rho_1 \leq \omega_{max}$ **then**
12: $p \leftarrow t^*$ ▷ t^* is calculated by (20)
13: **else**
14: $p \leftarrow t_1^* + t_2^* + t_3^*$ ▷ See (21)
15: **end if**
16: **end if**
17: **return** p
18: **end procedure**

4 An illustrative example

In this section, we show how the proposed method delivers minimum inter-release times in comparison with existing methods using a sample AVR task. Also, it is shown how the accuracy of the analysis can be improved by using more vertices in the DRT task. For our purpose, we consider an AVR task with the parameters used in [6] and [8]. Accordingly, the minimum and maximum speeds are assumed as $\omega_{min} = 500$ rpm and $\omega_{max} = 6500$ rpm. Further, the acceleration range is set as $[\alpha^-, \alpha^+] = [-600000, 600000]$.[3] This setting is regarded as a reasonable representative for typical production car engines [8].

4.1 Calculating minimum inter-release time

This section demonstrates how the proposed method can provide a safe and also accurate value for minimum inter-release times. We assume two modes with speed ranges $[500, 600)$ and $[700, 800)$. The goal is to calculate the minimum time of one full rotation, while the initial and final speeds are restricted to be in the former and latter speed range, respectively. The diagram in the bottom of Fig. 3 shows the acceleration during one rotation for three scenarios. The first scenario, represented with solid lines, is related to the proposed method, where case 4 applies (Lines 11-12 in Algorithm 2). As seen, the acceleration is selected to get its maximum value, namely 600×10^3, up to a certain point, and then, its minimum value, i.e., -600×10^3, during the rest of the interval. In contrast, the dashed line indicates the acceleration which leads to the minimum time, when the system is supposed to select a fixed acceleration during one rotation. In addition, the dot-dashed line corresponds to a pessimistic approach where the acceleration constantly is assigned the maximum value.

The diagram in the middle of Fig. 3 shows the corresponding speeds for the mentioned three approaches as a function of time. Also, the topmost diagram shows the respective

[3] We use the *revolution per minute (rpm)* unit for rotational speed and *rpm per minute* for acceleration.

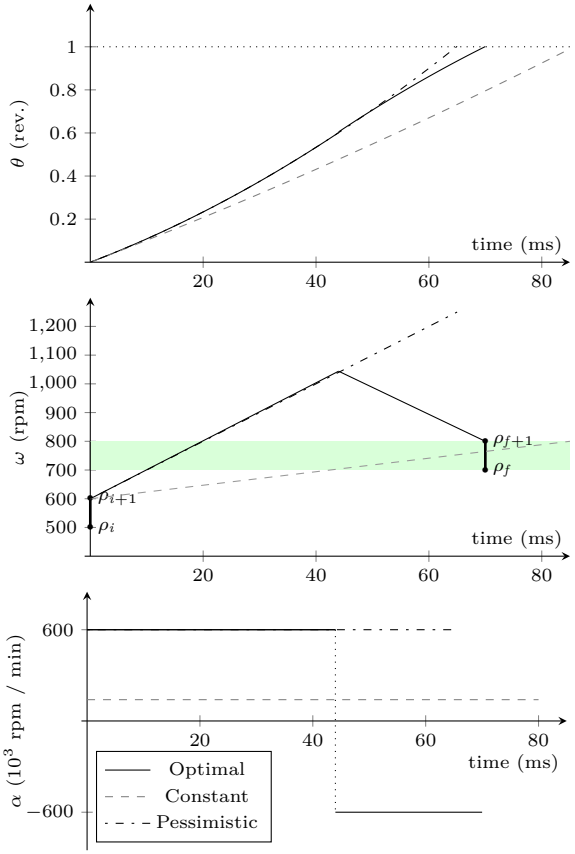


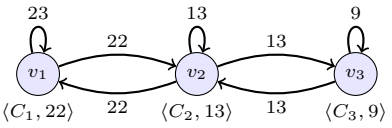
Figure 3 Illustration of a rotation that leads to minimum time using three approaches. The diagram in the bottom shows the acceleration, the middle one shows the respective speeds, and the upper one depicts the corresponding angular positions.

angular variations. In this diagram, the time for one rotation is the point at which the associated curve hits the horizontal (dotted) line. As seen, the minimum time (related to the solid line) is around 70 ms, while the respective time for the case of a fixed acceleration will be nearly 85 ms. This comparison shows that the assumption of fixed acceleration between the release of two successive jobs, as is made in [5], [6], and [4], may lead to an optimistic result. On the other hand, one can obtain a pessimistic result by relaxing the constraint on the final speed, and then, selecting a constant maximum acceleration (as used in [10]). This situation is shown by the dot-dashed line in the diagram, revealing a minimum time of around 65 ms. In contrast, our proposed method exhibits a tighter, yet safe, result.

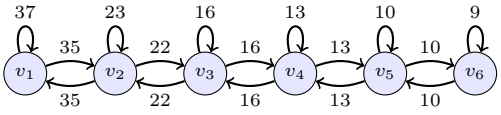
4.2 Deriving a DRT task

For extracting a DRT graph, we consider an AVR task with the same speed and acceleration ranges as before. Further, we assume three modes recognized by the following speed ranges: $[500, 2500)$, $[2500, 4500)$, and $[4500, 6500)$. WCETs of the respective jobs for these modes are assumed to be 5 ms, 3 ms, and 1 ms, respectively.

Figures 4a and 4b show the corresponding DRT tasks obtained through the proposed method when using three and six vertices, respectively. The values on edges are in milliseconds and have been rounded down to get a safe approximation of the minimum inter-release times.



(a) An extracted DRT with three vertices.



(b) An extracted DRT with six vertices.

Figure 4 DRT graphs for a sample AVR task.

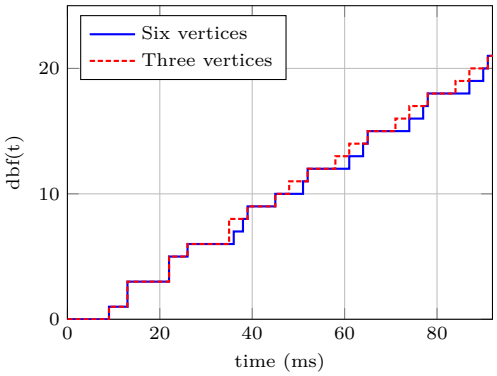


Figure 5 Demand-bound functions.

In the DRT graph shown in Fig. 4a, one vertex is considered per mode. In contrast, the graph shown in Fig. 4b has two vertices for each mode of the AVR task, providing a model with finer granularity. For this model, we considered six speed intervals with equal lengths.

To observe how the number of vertices influences timing analysis, we have calculated the respective *demand-bound function* (dbf) [20] for each DRT task. Informally, a dbf is a function of time; for any time instant $t \geq 0$ its value denotes the maximum accumulated workload that can arrive within any time interval of length t and have a deadline in the same interval. This function provides a direct way to express a necessary and sufficient condition for feasibility of a DRT task set [17]. As seen in Fig. 5, the task model with six vertices always delivers an equal or smaller (i.e., tighter) workload than that of the model with three vertices. This means that an AVR task which is actually schedulable, may be (pessimistically) deemed as an unschedulable task when analyzed with a coarse grain model, i.e., the DRT graph with three vertices. But when the graph is *refined* to a DRT task with more vertices, it may be (correctly) recognized as a schedulable task.

5 Faithfulness of the method

Intuitively, a DRT graph is said to be a faithful (or sound) model for an AVR task if its schedulability implies schedulability of the AVR task. In the following, we formalize this definition, and then, validate that our method constructs a faithful model. For this purpose, we first provide a number of required definitions.

5.1 Definitions

For an AVR task with a given set of modes, let $C(\omega)$ denote a function which takes a speed value ω and returns the WCET of the mode to which ω belongs. Further, by a triple (R_i, C_i, ω_i) we denote a job released by the AVR task with a WCET of C_i and an instantaneous rotational speed of ω_i at time instant R_i. Also, we define a notion of *valid trajectory* as follows.

▶ **Definition 5** (Valid trajectory)**.** Given two speeds ω_0 and ω_f, and a subset of real numbers $I = [r_1, r_2]$ with $r_1, r_2 \in \mathbb{R}_{\geq 0}$, a function $\alpha(t) : I \mapsto \mathbb{R}$ is said to be a valid acceleration

trajectory, or simply a valid trajectory, from ω_0 to ω_f in the domain of I if it satisfies

$$\begin{aligned} &\alpha^- \leq \alpha(t) \leq \alpha^+, && \forall t \in [r_1, r_2] \quad \wedge \\ &\omega_{min} \leq \tilde{\omega}(t) \leq \omega_{max}, && \forall t \in [r_1, r_2] \quad \wedge \\ &\tilde{\omega}(r_2) = \omega_f, && \wedge \\ &\int_{r_1}^{r_2} \tilde{\omega}(t)dt = 1, \end{aligned} \tag{22}$$

where $\tilde{\omega}(x)$ is defined as $\tilde{\omega}(x) = \omega_0 + \int_{r_1}^{x} \alpha(t)dt$.

In order to capture the workload generated by AVR tasks, we define the notion of $\mathcal{A}$-*trace*.

▶ **Definition 6** ($\mathcal{A}$-trace)**.** Take an AVR task as specified in Section 2.2. Then, any sequence of triples $[(R_1, C_1, \omega_1), \ldots, (R_n, C_n, \omega_n)]$ is called an "$\mathcal{A}$-trace" generated by the AVR task if

- $C_i = C(\omega_i)$ for all $i \in \{1, \ldots, n\}$, and
- for each $i \in \{1, \ldots, n-1\}$ there exists a valid trajectory from ω_i to ω_{i+1}, defined over the domain of $[R_i, R_{i+1}]$.

Similarly, a notion of $\mathcal{D}$-*trace* is defined for a DRT task.

▶ **Definition 7** ($\mathcal{D}$-trace)**.** Consider a DRT task $G = (V, E)$. Let $v_1 \to v_2 \to \ldots \to v_n$ be an arbitrary path in G. Then, any sequence of triples $[(R_1, C_1, v_1), \ldots, (R_n, C_n, v_n)]$ is called a $\mathcal{D}$-trace generated by the task if

- $C_i \leq e(v_i)$ for all $i \in \{1, \ldots, n\}$, and
- $R_{i+1} - R_i \geq p(v_i, v_{i+1})$ for all $i \in \{1, \ldots, n-1\}$.

For specifying the workload associated to the AVR and DRT task models in an abstract and also comparable way, we further define a notion of job sequence.

▶ **Definition 8** (Job sequence)**.** Any sequence of tuples $[(R_1, C_1), \ldots, (R_n, C_n)]$ with $R_i, C_i \in \mathbb{R}_{\geq 0}$, for $1 \leq i \leq n$, and $R_i < R_{i+1}$, for $1 \leq i < n$, is called a job sequence.

A job sequence $\sigma = [(R_1, C_1), \ldots, (R_n, C_n)]$ is said to be producible by an AVR task if and only if there exist values $\omega_1, \omega_2, \ldots, \omega_n$ such that $[(R_1, C_1, \omega_1), \ldots, (R_n, C_n, \omega_n)]$ is an $\mathcal{A}$-trace generated by the task. Analogously, σ can be generated by a DRT task G if and only if there exists a path $v_1 \to v_2 \to \ldots \to v_n$ in G such that $[(R_1, C_1, v_1), \ldots, (R_n, C_n, v_n)]$ is a $\mathcal{D}$-trace generated by G. Such workload characterization conforms to the semantics of DRT [20], which enables us to employ the respective timing analysis methods [19, 20]. According to these definitions, the notion of behavioral inclusion is defined as follows.

▶ **Definition 9** (Behavior inclusion)**.** Consider a DRT task, denoted by T_D, and an AVR task, denoted by T_A. Let σ_D and σ_A denote the set of all job sequences generated by T_D and T_A, respectively. Then, the behavior of T_A is said to be included by T_D if $\sigma_A \subseteq \sigma_D$.

Finally, we define faithfulness of a model as follows.

▶ **Definition 10** (Faithfulness)**.** A DRT graph T_D provides a faithful model for an AVR task T_A if schedulability of T_D entails schedulability of T_A.

5.2 Faithfulness validation

The goal is to demonstrate that Algorithm 1 reveals a faithful model for a given AVR task. For this purpose, we first review a result with respect to behavior inclusion.

▶ **Lemma 11.** *Consider a set of AVR tasks τ_A and a set of corresponding DRT tasks τ_D where the behavior of each AVR task is included by the respective DRT task. Then, schedulability of τ_D implies schedulability of τ_A.*

Proof. We prove the contrapositive. Let τ_A be unschedulable. Then, there exists a set of job sequences, each produced by one task, in which a job misses its deadline. As it is supposed that the behavior of every AVR task is included by its corresponding DRT task, the same set of job sequences can be generated by τ_D as well. As a result, there exists a scenario in which τ_D is unschedulable, too, by which the lemma is followed. ◀

Now, we will show that the DRT graph created by Algorithm 1 includes the behavior of the given AVR task. For this goal, we review the following property regarding the algorithm.

▶ **Property 3.** *Consider an AVR task T_A and the corresponding DRT task T_D. Let $[(R_1, C_1, \omega_1), \ldots, (R_n, C_n, \omega_n)]$ denote an arbitrary $\mathcal{A}$-trace generated by T_A. Further, let $v(\omega_i)$ denote the vertex in the graph of T_D whose corresponded speed range includes ω_i. Then, based on Line 6 in Algorithm 1, we have*

$$C(\omega_i) \leq e(v(\omega_i)), \quad \text{for } 1 \leq i \leq n. \tag{23}$$

Moreover, according to Lemma 4, it holds that

$$p\left(v(\omega_i), v(\omega_{i+1})\right) \leq R_{i+1} - R_i, \quad \text{for } 1 \leq i < n. \tag{24}$$

Using this property, we establish a relation between an AVR task and its associated DRT graph in terms of behavioral inclusion.

▶ **Lemma 12.** *The DRT task constructed by Algorithm 1 includes the behavior of the given AVR task.*

Proof. Consider any arbitrary AVR task T_A and the associated DRT task T_D. Based on Definition 9, we need to show that the set of job sequences generated by T_A is a subset of that of T_D. For this purpose, we show that for any $\mathcal{A}$-trace generated by the AVR task, there exists (at least) one $\mathcal{D}$-trace generated by T_D which gets the same job sequence. Take an arbitrary $\mathcal{A}$-trace $\sigma = [(R_1, C_1, \omega_1), \ldots, (R_n, C_n, \omega_n)]$ generated by T_A. Correspondingly, consider a $\mathcal{D}$-trace specified as $\sigma' = [(R_1, C_1, v(\omega_1)), \ldots, (R_n, C_n, v(\omega_n))]$. According to Property 3, relations (23) and (24) hold for σ'. As a result, by the definition of $\mathcal{D}$-trace in Definition 7, σ' constitutes a valid $\mathcal{D}$-trace for T_D. Then, since the job sequence associated to σ and that of σ' are the same, the proof completes. ◀

Putting these together, we show that Algorithm 1 provides a faithful model.

▶ **Lemma 13.** *The DRT graph generated for an AVR task by Algorithm 1 is a faithful model.*

Proof. Based on Lemma 12, the AVR task behavior is included by the obtained DRT graph. According to Lemma 11, this is a sufficient condition for faithfulness defined in Definition 10, which implies faithfulness of the DRT graph. ◀

6 Deriving a tight model

Up to now, we have shown that the constructed DRT graph gives a safe characterization in terms of schedulability analysis. However, it may lead to a pessimistic result. This section provides the criteria under which a DRT task provides a tight workload characterization, i.e., neither pessimistic nor optimistic, for an AVR task.

6.1 Definitions and assumptions

Before tackling tightness of the method, we present two definitions.

▶ **Definition 14** (Point and interval reachability)**.** A speed ω_f is said to be reachable from a speed ω_0, denoted as $\omega_0 \rightsquigarrow \omega_f$, if there exists a valid trajectory from ω_0 to ω_f. Further, a speed interval F is reachable from another speed interval I, denoted as $I \overset{\Delta}{\rightsquigarrow} F$, if $\exists \omega_0 \in I, \exists \omega_f \in F : \omega_0 \rightsquigarrow \omega_f$.

Intuitively, a speed ω_f is reachable from ω_0 if, starting with the speed ω_0, the engine speed can reach ω_f after exactly one rotation, while speed and acceleration constraints are preserved.

▶ **Definition 15** (Tight model)**.** A DRT task T_D is a tight model for an AVR task T_A with respect to a scheduling policy Sch if, when the system is scheduled by Sch,

1. schedulability of T_D entails schedulability of T_A, and
2. unschedulability of T_D entails unschedulability of T_A.

In the following, we assume that the absolute values of the maximum and minimum accelerations are equal, i.e., $\alpha^+ = -\alpha^-$.

6.2 Sufficient conditions for tightness

In this section, we present sufficient conditions under which a DRT task is a tight model for a given AVR task.

▶ **Lemma 16.** *Consider an AVR task, denoted by T_A, and a DRT task, denoted by T_D, which is obtained by Algorithm 1 when applied on T_A with a given speed partitioning. Suppose that*

$$\hat{\sigma}_D \subseteq \sigma_A, \tag{25}$$

where $\hat{\sigma}_D$ denotes the set of those job sequences generated by T_D in which the inter-release separation time between every two successive jobs is exactly equal to the corresponding minimum inter-release time specified by T_D. Then, T_D is a tight model for T_A with respect to EDF and also any fixed priority scheduling algorithm.

Proof. For the proof, we need to verify the two tightness conditions specified in Definition 15. The first one is already shown by Lemma 13. To validate the second property, we notice that $\hat{\sigma}_D$ always provides the critical scheduling instant for both EDF and fixed priority scheduling policies [20]. In other words, if the DRT task is unschedulable, then there exists a job sequence in $\hat{\sigma}_D$ which leads to a deadline miss. In turn, according to (25), such a job sequence is also included by σ_A. As a result, the considered AVR task contains a scenario leading to a deadline miss, which means unschedulability of the task. Thus, the second condition for the tightness also holds. ◀

Using the above lemma, we present conditions under which Algorithm 1 provides a tight DRT task. Take an arbitrary AVR task with a speed range $[\omega_{min}, \omega_{max})$ and a set of M modes. Assume a partitioning of $[\omega_{min}, \omega_{max})$ into K sub-intervals $P = \{[\rho_1, \rho_2), \ldots, [\rho_K, \rho_{K+1})\}$ (with $\rho_1 = \omega_{min}$ and $\rho_{K+1} = \omega_{max}$) for which the following properties hold.

$$\forall i, j \in \{1, \ldots K\} : [\rho_i, \rho_{i+1}) \overset{\Delta}{\rightsquigarrow} [\rho_j, \rho_{j+1}) \implies \rho_{i+1} \rightsquigarrow \rho_{j+1}, \tag{26}$$

$$\forall m \in \{1, \ldots, M\} : \exists i \in \{1, \ldots, K+1\} : \omega_m = \rho_i. \tag{27}$$

The first property states that if there is any speed in the second range which is reachable from one speed in the other range, then the maximum speed of the second range should be

reachable from the maximum speed of the first range. As a result, our method for calculating minimum inter-release times for a vertex in the corresponding DRT graphs, which assumes the maximum feasible speeds (see Fig. 2), does not introduce a pessimism. Property (27) requires that every speed in the boundary of a mode must be a boundary speed in one of the sub-intervals in P.

▶ **Lemma 17.** *The DRT graph obtained by Algorithm 1 for a given AVR task under a given speed partitioning with the properties specified by* (26) *and* (27) *provides a tight model.*

Proof. Let T_D and T_A denote the DRT task and AVR task, respectively, and $\hat{\sigma}_D$ to be as defined as in Lemma 16. According to Lemma 16, it suffices to show that (25) holds. Consider an arbitrary job sequence in $\hat{\sigma}_D$ which corresponds to a $\mathcal{D}$-trace specified as $\sigma' = [(R_1, C_1, v_1), \ldots, (R_n, C_n, v_n)]$. We show that there exists an $\mathcal{A}$-trace as well which produces the same job sequence. Let $[\rho_l(v_i), \rho_u(v_i))$ denote the speed interval associated to the vertex v_i, for $i \in \{1, \ldots, n\}$. Then, consider the $\mathcal{A}$-trace $\sigma = [(R_1, C_1, \rho_u^-(v_1)), \ldots, (R_n, C_n, \rho_u^-(v_n))]$, where $\rho_u^-(v_i)$, for $1 \leq i \leq n$, denotes a value smaller than, but sufficiently close to $\rho_u(v_i)$. Due to the second condition (i.e., (27)), each vertex of the DRT task is associated to only one mode of the AVR task. As a result, the WCET of each vertex is exactly the WCET of that mode. Furthermore, according to the assumed $\mathcal{D}$-trace, there exists at least one speed value in $[\rho_l(v_{i+1}), \rho_u(v_{i+1}))$ which is reachable from $[\rho_l(v_i), \rho_u(v_i))$, for $1 \leq i < n$ (because otherwise, there is no edge from v_i to v_{i+1}, and then, v_i and v_{i+1} cannot be successive jobs). Due to the first assumption above, i.e., (26), this implies that the speed $\rho_u^-(v_{i+1})$ is reachable from $\rho_u^-(v_i)$. The corresponding edge label, that is, the specified minimum inter-release time, in the DRT task is also equal to the time it takes to make one full rotation when the initial speed is $\rho_u^-(v_i)$ and the final speed is $\rho_u^-(v_{i+1})$ (based on Algorithm 1 and Fig. 2). As a result, σ denotes a valid $\mathcal{A}$-trace which can be generated by the AVR task. Hence, $\hat{\sigma}_D \subseteq \sigma_A$. ◀

6.3 An exact DRT model

In this section we provide a method to obtain a speed partitioning satisfying Properties (26) and (27), and as a result, providing a tight DRT task.

Intuitively, in order for a speed partitioning to satisfy (26), all speeds reachable from a boundary speed must be also a boundary speed. Based on this, the set of speeds used to partition the speed range $[\omega_{min}, \omega_{max})$ of an AVR task which contains M modes is defined as:

$$\begin{aligned}\rho = \{\omega_i | 1 \leq i \leq M+1\} \cup \{\Omega_n(\omega_i, \alpha^+) < \omega_{max} | 1 \leq n, 1 \leq i \leq M\} \\ \cup \{\Omega_n(\omega_i, \alpha^-) > \omega_{min} | 1 \leq n, 2 \leq i \leq M+1\}\end{aligned} \tag{28}$$

▶ **Corollary 18.** *Consider the set ρ defined in* (28)*. For any $s \in \rho$, either $\Omega_1(s, \alpha^+) \in \rho$, or $\Omega_1(s, \alpha^+) > \omega_{max}$.*

We then obtain a speed partitioning using the speeds in ρ as interval *boundaries*.

▶ **Lemma 19.** *Consider the set ρ defined in* (28) *sorted in ascending order. Let the interval between any two consecutive speeds of ρ to be a distinct speed range, leading to a set of $|\rho| - 1$ intervals. The speed partitioning obtained in this way satisfies Properties* (26) *and* (27)*.*

Proof. The second property (namely (27)) holds immediately by the definition of ρ. Suppose that the first property does not hold. This means that, $\exists i, j \in \{1, \ldots, |\rho| - 1\}$ such that

Table 1 Mode parameters of the considered AVR task.

i (mode)	1	2	3	4	5	6
ω_i (rpm)	500	1500	2500	3500	4500	5500
C_i (μs)	965	576	424	343	277	246

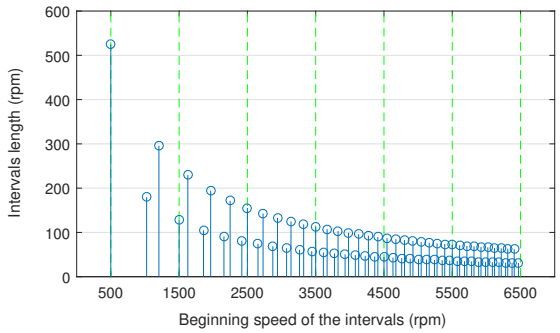


Figure 6 A partitioning of the speed range which leads to a tight characterization.

$[\rho_i, \rho_{i+1}) \overset{\Delta}{\rightsquigarrow} [\rho_j, \rho_{j+1})$ but ρ_{j+1} is not reachable from ρ_{i+1}. Without loss of generality, we assume $\rho_{i+1} \leq \rho_j$. These assumptions imply that

$$\rho_j < \Omega_1(\rho_{i+1}, \alpha^+) < \rho_{j+1}. \tag{29}$$

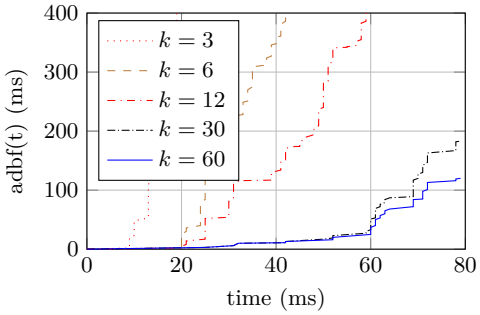
However, according to Corollary 18, $\Omega_1(\rho_{i+1}, \alpha^+)$ should be a boundary speed, which contradicts with (29). ◀

Putting Lemma 17 and Lemma 19 together implies that the DRT graph obtained by using the above speed partitioning is a tight model. It is also worth noting that, based on the definition of ρ in (28) and the definition of Ω_i in (8), the number of speed ranges obtained by the partitioning described in Lemma 19 is $O(M \times \frac{\omega_{max}^2 - \omega_{min}^2}{2\alpha^+})$.

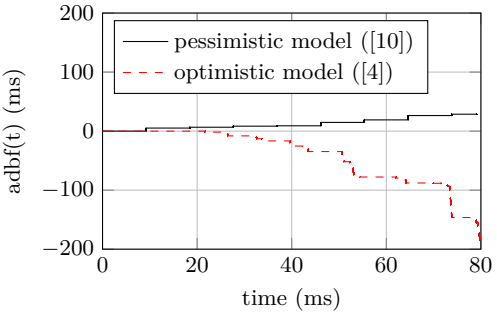
7 Evaluation results

For evaluation, the AVR task adopted in [5] is used. The parameters ω_{min}, ω_{max}, α^-, and α^+ get the same values as in Section 4. Six modes are attributed to the task, described in Table 1 [5]. We first applied the method presented in Section 6 to obtain a speed range partitioning which reveals a DRT model with a tight characterization. The result is a partitioning with 70 sub-intervals, depicted in Fig. 6. The boundary speeds of the intervals are shown on the x-axis. To each boundary speed, a vertical line is associated which shows the length of the interval whose starting point is that boundary speed. In higher speeds, the intervals are typically shorter since the amount of speed evolution during one full rotation is smaller for larger speed values.

In the following, we first examine the accuracy and runtime of the models with different levels of granularity, as well as the models obtained by pessimistic and optimistic approaches, in Subsection 7.1. Next, in Subsection 7.2, we show how the pessimistic and optimistic approaches lead to inaccurate schedulability tests using two counter examples. All the analyses is done using the Python library implemented for timing analysis of DRT tasks [20].



(a) The result for DRT tasks with different granularity in speed partitioning.



(b) Models obtained by the pessimistic and optimistic approaches.

Figure 7 Accuracy of DRT tasks obtained by different methods compared to the exact one.

7.1 Accuracy and run-time performance

As mentioned earlier, the accuracy of a DRT graph can be improved by using more vertices. In our experiments, we extract five models with 3, 6, 12, 30, and 60 vertices. In each case, the speed range $[\omega_{min}, \omega_{max}]$ is divided into equal-length intervals. The result obtained for the task with the tight characterization is used as a reference. This task is obtained by applying our DRT construction method to the speed partitioning of Fig. 6.

To measure the accuracy, we compute the accumulated difference between the demand-bound function (dbf) [17] of each model and the reference one. We define this measure as $adbf(t) = \sum_{i=0}^{t} \left(dbf_k(t) - \hat{dbf}(t) \right)$, where $dbf_k(t)$, for $k \in \{3, 6, 12, 30, 60\}$, denotes dbf of the task with k vertices, and $\hat{dbf}(t)$ denotes that of the reference task model. The results are plotted in Fig. 7a, where the x-axis is the window size for which adbf is computed. As seen, by increasing the number of vertices, the accuracy of the model considerably improves.

We also apply the methods proposed in [10] and [4] to the speed partitioning obtained by our method in Section 6, depicted in Fig. 6. Compared to the tight DRT graph, which is used as a reference, the resultant DRT graphs contain the same number of vertices. However, for the model of [10], i.e., the pessimistic model, minimum inter-release times are equal to or smaller than the values obtained by our method. In contrast, minimum inter-release times for the model of [4], referred to as the optimistic model, are equal to or larger than those of the reference one. The comparison result is shown in Fig. 7b, demonstrating that the pessimistic and optimistic methods for calculation of minimum inter-release times introduce inaccuracies to the analysis, even when applied to an appropriate speed partitioning. We investigate the impact of this inaccuracy on schedulability tests in the next subsection.

We also examine the runtime of the methods. For this, the code is run using the pypy-5.1 compiler on a quad-core processor with 2.40 GHz frequency and a memory of 8 GB. Although the reported results are platform-dependent, they provide a suitable view to the relative complexity and scalability of the analyses. Table 2 gives the time needed for computing the dbf for each model for the interval $[0, 80000]\mu s$. As expected, the runtime grows as the granularity is increased from $k = 3$ vertices to $k = 60$ vertices. Runtime for the tight model is also larger than that of the model with $k = 60$ vertices since it contains 70 vertices.

Additionally, the runtime for the tight model is less than the pessimistic one, and larger than the optimistic one. To reason about this, we point out that DRT graphs in all of these three cases contain the same number of vertices and edges. Meanwhile, to obtain dbf for any time instant t, those paths of the graph that can be traversed within a time interval of

Table 2 Runtime of computing dbf for different models.

DRT model	$k=3$	$k=6$	$k=12$	$k=30$	$k=60$	Tight	Pessimistic	Optimistic
Run-time (s)	0.09	0.5	0.71	1.93	10.36	15.82	16.45	14.01

t need to be enumerated. Hence, smaller minimum inter-release times potentially lead to more number of paths for a specified t. Based on this observation, the pessimistic approach exhibits larger runtime since it reveals smaller minimum inter-release times. On the other hand, the optimistic method consists of equal or larger values of minimum inter-release times, which has implied a shorter running time. Finally, it is worth mentioning that speed partitioning and DRT graph construction steps take negligible time (less than 1%) compared to the time needed for computing dbfs.

7.2 Comparison with respect to schedulability

In order to illustrate the influence of using imprecise models on schedulability tests, we investigate EDF schedulability of a task set which consists of two tasks: an AVR task with the above-described parameters, and a sporadic task whose WCET, relative deadline, and period are denoted by C, D, and P, respectively. We recall that a set of independent real-time tasks τ is EDF schedulable if, and only if, for all $t \geq 0 : \sum_{T \in \tau} dbf_T(t) \leq t$, where $dbf_T(.)$ denotes the dbf of task T [20]. We denote dbf of the tight, pessimistic, and optimistic models, respectively, by $\hat{dbf}(.)$, $\hat{dbf}_P(.)$, and $\hat{dbf}_O(.)$. Further, we denote that of the sporadic task with $dbf_S(.)$. In what follows, the time unit is microsecond.

We first focus on the pessimistic approach, i.e., the method proposed in [10]. For this, the sporadic task parameters are assumed as $C = 8980$, $D = 9210$, and $P = 20000$. To investigate the pessimism, we are interested in an instant t for which $\hat{dbf}_P(t) > \hat{dbf}(t)$. Based on the computed dbfs, we consider $t = 9210\mu s$, for which $\hat{dbf}_P(9210) = 246$, $\hat{dbf}(9210) = 0$, and $dbf_S(9210) = 8980$. When using the pessimistic model, we get $\hat{dbf}_P(t) + dbf_S(t) = 9226 > 9210 = t$, suggesting unschedulability. Besides, it can be verified that, for all $t \geq 0$, $\hat{dbf}(t) + dbf_S(t) \leq t$ holds, meaning that the task set is, in fact, schedulable under EDF.

Next, the optimistic approach proposed in [4] is examined. For this case, we assume the sporadic task parameters as: $C = 25720, D = 26400, P = 50000$. Considering the time instant $t = 26400$, we have $\hat{dbf}_O(26400) = 620$, $\hat{dbf}(26400) = 686$, and $dbf_S(26400) = 25720$. If we use the dbf obtained from the tight model, we get $\hat{dbf}(t) + dbf_S(t) = 26406 > 26400 = t$, which implies unschedulability of the task set. However, writing this condition using the optimistic dbf yields $\hat{dbf}_O(t) + dbf_S(t) = 26340 \leq 26400 = t$. Furthermore, by running a feasibility test using the optimistic dbf, the task set is revealed schedulable, which is not a valid result. The same optimistic assumption has been used in analyses in [5, 6], which can lead to similar unsafe results. To the best of authors knowledge, except for the mentioned studies, other work in the literature have not been impacted by this optimistic assumption.

8 Conclusion and future work

In this paper, we studied engine control applications in which real-time task parameters are functions of the crankshaft speed. For timing analysis of such tasks, we employed the DRT task model [17]. We formulated minimum inter-release time calculation as an optimal control problem where the goal is to find the control input (i.e., the acceleration) that leads to the minimum time for taking a specified change in the angular position. We used a technique from the calculus of variations to solve this minimization problem and determine inter-release

times in the corresponding DRT task. To construct the DRT task, one needs to partition the engine's speed range. In this work, we also proposed a speed range partitioning method which leads to a tight workload characterization in the sense of timing analysis. We showed that our approach provides faithful and tight timing analyses compared to the existing methods.

This work can be extended in multiple directions. In the current study, we focused on independent AVR tasks. The model can be extended to specify tasks which depend on a common rotation source [9, 2]. While the proposed method provides a pessimistic approach to analyze this extended model, it can be improved by employing more expressive models, e.g., the recently proposed synchronous DRT [14] task model. Another direction of extension is considering cyber-physical systems with more general dynamic behavior, for instance, those specified by *hybrid automata* [1]. One can extend our proposed approach to abstract the behavior of such systems for efficient timing analysis, observing potential limitations.

References

1 R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theor. Comput. Sci.*, 138(1):3–34, February 1995. doi:10.1016/0304-3975(94)00202-T.

2 A. Biondi and G. Buttazzo. Engine control: Task modeling and analysis. In *Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 525–530, March 2015. doi:10.7873/DATE.2015.0147.

3 A. Biondi and G. Buttazzo. Real-time analysis of engine control applications with speed estimation. In *Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 193–198, March 2016. doi:10.3850/9783981537079_0273.

4 A. Biondi, G. Buttazzo, and S. Simoncelli. Feasibility analysis of engine control tasks under edf scheduling. In *Euromicro Conference on Real-Time Systems (ECRTS)*, pages 139–148, July 2015. doi:10.1109/ECRTS.2015.20.

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 *Euromicro Conference on Real-Time Systems (ECRTS)*, pages 165–174, July 2014. doi:10.1109/ECRTS.2014.38.

6 A. Biondi, M. Di Natale, and G. Buttazzo. Response-time analysis for real-time tasks in engine control applications. In *International Conference on Cyber-Physical Systems (ICCPS)*, pages 120–129, New York, NY, USA, 2015. ACM. doi:10.1145/2735960.2735963.

7 G. C. Buttazzo, E. Bini, and D. Buttle. Rate-adaptive tasks: Model, analysis, and design issues. In *Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1–6, March 2014. doi:10.7873/DATE.2014.266.

8 R. I. Davis, T. Feld, V. Pollex, and F. Slomka. Schedulability tests for tasks with variable rate-dependent behaviour under fixed priority scheduling. In *Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 51–62, April 2014. doi:10.1109/RTAS.2014.6925990.

9 T. Feld and F. Slomka. Sufficient response time analysis considering dependencies between rate-dependent tasks. In *Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 519–524, March 2015. doi:10.7873/DATE.2015.0150.

10 Z. Guo and S. K. Baruah. Uniprocessor EDF scheduling of AVR task systems. In *International Conference on Cyber-Physical Systems (ICCPS)*, pages 159–168. ACM, 2015. doi:10.1145/2735960.2735976.

11 Lino Guzzella and Christopher Onder. *Introduction to Modeling and Control of Internal Combustion Engine Systems*. Springer Science & Business Media, 2009.

12 J. Kim, K. Lakshmanan, and R. Rajkumar. Rhythmic tasks: A new task model with continually varying periods for cyber-physical systems. In *International Conference on Cyber-Physical Systems (ICCPS)*, pages 55–64, April 2012. doi:10.1109/ICCPS.2012.14.

13 Donald E. Kirk. *Optimal Control Theory: An Introduction.* Dover Publications, 1998.

14 M. Mohaqeqi, J. Abdullah, N. Guan, and W. Yi. Schedulability analysis of synchronous digraph real-time tasks. In *Euromicro Conference on Real-Time Systems (ECRTS)*, pages 176–186, July 2016. doi:10.1109/ECRTS.2016.17.

15 V. Pollex, T. Feld, F. Slomka, U. Margull, R. Mader, and G. Wirrer. Sufficient real-time analysis for an engine control unit. In *International Conference on Real-Time Networks and Systems (RTNS)*, pages 247–254. ACM, 2013. doi:10.1145/2516821.2516838.

16 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 *Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1335–1338, March 2013. doi:10.7873/DATE.2013.275.

17 M. Stigge, P. Ekberg, N. Guan, and W. Yi. The digraph real-time task model. In *Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 71–80, April 2011. doi:10.1109/RTAS.2011.15.

18 M. Stigge, N. Guan, and W. Yi. Refinement-based exact response-time analysis. In *Euromicro Conference on Real-Time Systems (ECRTS)*, pages 143–152, July 2014. doi:10.1109/ECRTS.2014.29.

19 M. Stigge and W. Yi. Combinatorial abstraction refinement for feasibility analysis. In *Real-Time Systems Symposium (RTSS)*, pages 340–349, 2013. doi:10.1109/RTSS.2013.41.

20 Martin Stigge. *Real-Time Workload Models : Expressiveness vs. Analysis Efficiency.* PhD thesis, Uppsala University, Division of Computer Systems, 2014.

21 E. Velenis and P. Tsiotras. Optimal velocity profile generation for given acceleration limits: theoretical analysis. In *American Control Conference (ACC)*, pages 1478–1483 vol. 2, June 2005. doi:10.1109/ACC.2005.1470174.

22 H. Zeng and M. Di Natale. Computing periodic request functions to speed-up the analysis of non-cyclic task models. *Real-Time Syst.*, 51(4):360–394, July 2015. doi:10.1007/s11241-014-9209-5.