Schedulability Analysis of Engine Control Systems With Dynamic Switching Speeds

Yu Liu, Chao Peng, Yecheng Zhao, Yangyang Li, and Haibo Zeng, *Member, IEEE*

***Abstract*—In cyber-physical systems, certain tasks are activated according to a rotation source. For example, angular tasks in engine control units are triggered whenever the engine crankshaft reaches a specific angular position. To reduce the workload at high speeds, these tasks also adopt different implementations within different rotation speed intervals. However, current studies are limited to the case that the switching speeds at which task implementations should change are configured at design time. In this article, we propose to dynamically adjust the switching speeds at runtime. We develop schedulability analysis techniques for such systems, including a new digraph-based task model to safely approximate the workload from software tasks triggered at predefined rotation angles. We prove that such task transformation has bounded pessimism. We present exact algorithms to find a finite number of representatives to avoid enumerating (an infinite number of) all job sequences. Experiments on synthetic task systems demonstrate that the proposed approach provides substantial benefits on system schedulability.**

***Index Terms*—Adaptive variable-rate (AVR) tasks, cyber-physical systems, engine control, schedulability analysis.**

I. Introduction

MODERN cyber-physical systems may contain tasks that respond to external events generated by a rotation source. Hence, their activation periods and deadlines are dependent on the angular speed. Also, to avoid CPU overload on the host microprocessor at high speeds, they are designed to be self-adaptive in that they switch to simplified implementations at higher speeds. For this reason, these tasks are often referred to as adaptive variable-rate (AVR) tasks in [2]. An example is the engine control system in internal combustion vehicles, which determines the timing and amount of fuel injection. Certain software tasks (called *angular tasks*) in it are triggered at predefined rotation angles of the engine crankshaft. It adopts different control strategies within different engine rotation speed intervals [3]. The most complex control strategy (e.g., with multiple fuel injections in one engine revolution) has the best performance (with respect to emission and fuel efficiency), but it comes with the highest computational demand.

The existing studies on systems with AVR tasks all assume that the *switching speeds* (at which AVR tasks switch implementations) are configured *offline*. This means that the optimization of switching speeds will have to be based on design-time information, which is clearly suboptimal since vehicle speeds (and thus engine speeds) will change at runtime. Hence, we propose the concept of *AVR tasks with dynamic switching speeds*, where the switching speeds are dynamically adjusted according to runtime information. We term the corresponding AVR tasks as dynamic AVR tasks (*dAVR* tasks). In contrast, the AVR tasks in systems with statically configured switching speeds are called static AVR tasks (*sAVR* tasks).

Our proposal is inspired by the upcoming era of connected and automated vehicles (CAVs), which is envisioned to transform the transportation systems. In this new era, vehicles can access valuable information about the driving environment at runtime, using various sensing (e.g., camera, radar, and lidar) and communication (such as vehicle-to-vehicle and vehicle-to-infrastructure) capabilities. This provides rich opportunities to substantially improve vehicle operations using such real-time information [4], including path and speed planning [5], [6], vehicle dynamics control [7], and as a potential application of this article, engine control [8].

Specifically, the engine control parameters including the switching speeds are configured at design time, typically using the distribution of engine speeds in a standard *driving cycle* (i.e., data points representing the vehicle speed over time). This may result in noticeably suboptimal engine performance, as the standard driving cycles can be substantially different from the actual ones, not to mention the variations of engine speeds within a driving cycle. With the CAV techniques making the driving cycle readily predictable [4], it becomes possible to dynamically adjust the switching speeds according to the upcoming engine speed. This has been shown to provide significant control performance gain (sometimes over 80%) over static switching speeds [8] using simulation.

In this article, we study the schedulability analysis of systems with dAVR tasks. We first review the related work.

A. Related Work

Systems with sAVR tasks are studied in a number of papers, see a recent survey [2]. The model from Buttle [3] introduced above is adopted by most researchers, with a couple of exceptions. Kim *et al.* [9] proposed the rhythmic task model and associated analysis that have a few restrictions, e.g., the task inter-release time is shortened by a fixed ratio during any acceleration. Pollex *et al.* [10], [11] assumed angular task release and WCET are independent, which may lead to high pessimism in the analysis. Feld and Slomka [12] considered

Manuscript received April 2, 2019; revised June 27, 2019 and September 28, 2019; accepted October 8, 2019. Date of publication November 11, 2019; date of current version September 18, 2020. This work was supported by NSF under Grant 1812963. This article was recommended by Associate Editor S. Pasricha. *(Corresponding author: Haibo Zeng.)*

Y. Liu and Y. Li are with the College of Mechanical and Vehicle Engineering, Hunan University, Changsha 410082, China.

C. Peng is with the Department of Computer Science and Technology, National University of Defense Technology, Changsha 410073, China.

Y. Zhao and H. Zeng are with the Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA 24060 USA (e-mail: zyecheng@vt.edu; hbzeng@vt.edu).

Digital Object Identifier 10.1109/TCAD.2019.2951124

the rate and offset based dependencies among the engine control tasks, but the task WCET is assumed to be a continuous function of the engine speed (instead of a number of discrete modes as in [3]).

Below we summarize the related work that are consistent with the model by Buttle [3]. Much of the research for the schedulability analysis focuses on handling the major difficulty that both WCET and inter-release time of an angular task strongly depend on the engine rotation speed, and our review reflects such a focus. We note that there are several variations on the assumptions, task models, and naming of angular tasks, and refer the readers to [2] for details.

For systems with fixed priority scheduling, Davis *et al.* [13] presented a number of sufficient analysis techniques on the worst-case interference from angular tasks such as quantization of the continuous engine speed space. Biondi *et al.* [14] proposed the concept of dominant speed that can represent a range of speeds in terms of the exact worst-case interference. This avoids quantization without loss of accuracy. An exact response time analysis is then derived [15], [16], and a set of design optimization techniques is proposed to optimize the switching speeds at design time [1], [17]. Feld and Slomka [18] improved the runtime of schedulability analysis, which is exact if the maximum acceleration and deceleration have the same absolute value. They also provided a sufficient analysis considering the angular phases between tasks [19]. Huang and Chen [20] assigned each mode of an angular task with a unique priority, and proposed a utilization-based schedulability test.

For EDF scheduled systems, a number of sufficient utilization-based schedulability tests are presented [21]–[23]. Differently, Biondi *et al.* [24], [25] proposed an exact analysis based on the concept of dominant speed as in [14]. Mohaqeqi *et al.* [26] proposed to partition the speed space and transform angular tasks to digraph real-time (DRT) tasks [27], [28]. Bijinemula *et al.* [29] further speeded up the calculation of exact worst-case demand using a knapsack-based method.

B. Our Contributions

Overall, all the previous studies assume the sAVR task model where the switching speeds are fixed offline. Differently, we propose the dAVR task model to allow dynamic adjustment to the switching speeds. We focus on the schedulability analysis of systems with dAVR tasks, under fixed priority scheduling on a uni-processor. However, such a new model introduces significant challenges to schedulability analysis. In particular, unlike all prior studies on sAVR tasks, it is no longer safe to characterize the interference from a dAVR task with a minimum inter-release time between its consecutive jobs, assuming the angular speeds at the job release times are all known.

In this article, we develop novel techniques to analyze the interference of dAVR tasks on a periodic task. Specifically, to avoid enumerating the speed in the (continuous) speed space, we partition it into a finite number of speed intervals, and transform a dAVR task to a new type of digraph-based real-time task model (called dDRT task) where each vertex represents each of the partitioned speed intervals, and the edges are labeled with both minimum and maximum inter-release times. We prove that the task transformation is always safe but sufficient only. We further quantify the pessimism and show that theoretically the proposed analysis based on the task transformation has a problem-dependent, but always bounded speedup factor. Hence, the pessimism introduced by the task transformation is always bounded in the speed of task execution compared to the exact analysis. We present algorithms to find a finite number of representative job sequences in the transformed dDRT task, to avoid enumerating (an infinite number of) all job sequences.

This article is organized as follows. In Section II, we describe the system model, including the model of dAVR tasks. In Section III, we present the schedulability analysis for a periodic task interfered by dAVR tasks. In Section IV, we describe the analysis for dAVR tasks. In Section V, we use synthetic systems to show the benefits of the proposed model and analysis techniques in terms of system schedulability.

II. System Model

Our system model extends that of sAVR tasks (e.g., [14] and [26]). The notations are summarized in Table I.

The rotation source, i.e., the engine, is described by its current rotation angle θ, angular speed ω, and angular acceleration α. Due to its physical attributes, the angular speed and acceleration are restricted in certain ranges, i.e., $\omega \in [\omega^{\min}, \omega^{\max}]$ and $\alpha \in [\alpha^{\min}, \alpha^{\max}]$. All these parameters are positive except the minimum acceleration $\alpha^{\min}$, and $|\alpha^{\min}| = -\alpha^{\min}$ is the maximum deceleration.

We consider a real-time system Γ containing a set of tasks scheduled with *fixed priority on a uni-processor*. A task either is periodic or is an AVR task. For convenience, a periodic task is denoted as τ_i while an AVR task is denoted as τ_i^*.

A periodic task τ_i is characterized by a tuple $\langle T_i, C_i, D_i, P_i \rangle$, where T_i is the period, C_i is the worst-case execution time (WCET), $D_i \leq T_i$ is the constrained deadline, and P_i is the priority. The execution of the periodic tasks, and consequently their parameters, are all *independent from the rotation source*.

An AVR task τ_i^* is triggered at predefined crankshaft angles $\theta_i = \Psi_i + k\Theta_i, \forall k \in \mathbb{N}$, where $\mathbb{N}$ is the set of non-negative integers, Ψ_i is the angular phase (the offset from the predefined reference angle), and Θ_i is the angular period. Its angular deadline is $\Delta_i = \lambda_i \cdot \Theta_i$ where $\lambda_i \leq 1$ (hence τ_i^* also has a constrained deadline). Similar to [15], we assume that AVR tasks *share a common rotation source, and they have the same angular period and phase*. Thus, we drop the task index from the angular period and phase, and denote them as Θ and Ψ. The AVR task parameters (WCET, inter-release time, and deadline) all depend on the dynamics of the rotation source.

A. Dynamics of Rotation Source

In this article, we assume *instantaneous angular speed at the job release time is known at runtime* [14], [15], [26]. We note that the exact knowledge of instantaneous engine speed may be impossible, and refer the readers to a detailed discussion of speed estimations in [30]. The speed after rotating θ angles with an initial speed ω and a constant acceleration α is calculated as [14]

$$\Omega(\omega, \alpha, \theta) = \sqrt{\omega^2 + 2\alpha\theta}. \tag{1}$$

Consider two angular speeds ω_k and ω_{k+1}, such that ω_{k+1} is *reachable* from ω_k after one angular period Θ. We denote

TABLE I
LIST OF NOTATIONS

Notation	Definition
θ	Rotation angle
ω	Angular speed
$\omega^{\min}/\omega^{\max}$	Minimum/maximum angular speed
α	Angular acceleration
$\alpha^{\min}/\alpha^{\max}$	Minimum/maximum angular acceleration
τ_i	Periodic task i
$\langle C_i, T_i, D_i \rangle$	$\langle$ WCET, Period, Deadline $\rangle$ of τ_i
τ_i^*	AVR task i
Δ_i	Angular deadline of τ_i^*
$D_i(\omega)$	Relative deadline of a job released by τ_i^* at speed ω
Ψ/Θ	Angular phase/period of all AVR tasks
$\Omega(\omega, \alpha, \theta)$	Speed after rotating θ angles with initial speed ω and constant acceleration α
$\omega_k \rightsquigarrow \omega_{k+1}$	ω_{k+1} is reachable from ω_k after one angular period Θ
$T^{\min}(\omega, \omega')/T^{\max}(\omega, \omega')$	Minimum/maximum time to rotate Θ angles, with initial speed ω and final speed ω'
$\mathcal{SM}_i = \{(SC_i^m, \varsigma\omega_i^m)\}$	Set of execution modes of an sAVR task τ_i^*
$\mathcal{SC}_i(\omega)$	WCET of a job of sAVR task τ_i^* released at speed ω
$\mathcal{T} = \{\gamma_1, \cdots, \gamma_T\}$	Set of reconfiguration times
$\mathcal{M}_{i,k} = \{(C_{i,k}^m, \omega_{i,k}^m)\}$	Execution mode sets of a dAVR task τ_i^* in time interval $[\gamma_k, \gamma_{k+1})$
$M_{i,k}$	Number of execution modes of a dAVR task τ_i^* in interval $[\gamma_k, \gamma_{k+1})$
$\mathcal{Q}_i = \{(\mathcal{M}_{i,k}, \gamma_k)\}$	Series of execution mode sets of a dAVR task τ_i^*
$\mathcal{C}_i(t, \omega)$	WCET of a job released by a dAVR task τ_i^* at time t and speed ω
$hp(i)$	Set of higher-priority periodic tasks than task i
$hp^*(i)$	Set of higher-priority AVR tasks than task i
τ_A^*	Representative dAVR task for a set of dAVR tasks
(σ, ω)	dAVR job released at time σ with angular speed ω
$\mathcal{A} = [(\sigma_1, \omega_1), \ldots, (\sigma_n, \omega_n)]$	dAVR job sequence
$\mathcal{A}.I(t)$	Interference function of $\mathcal{A}$ over an interval of length t
$R(\tau_i, \mathcal{A})$	Response time of τ_i interfered by a set of periodic tasks and a dAVR job sequence $\mathcal{A}$
$R(\tau_i, \tau_A^*)$	Response time of τ_i interfered by a set of periodic tasks and a representative dAVR task τ_A^*
$\mathcal{B} = \{\beta_0, \beta_1 \ldots \beta_B\} = \{(\beta_0, \beta_1] \ldots (\beta_{B-1}, \beta_B]\}$	Speed partition
$\tau_D^* = (\mathbb{V}, \mathbb{E})$	dDRT task, where $\mathbb{V}$ is the set of vertices, and $\mathbb{E}$ is the set of edges
$v.\mathcal{C}(t)$	WCET function of a vertex v in τ_D^*
$p^{\min}(v_i, v_j)/p^{\max}(v_i, v_j)$	Min/max inter-release time for edge (v_i, v_j)
$l(v_1, \ldots, v_n)$	Cumulative minimum inter-release time of all edges in path $(v_1, \ldots, v_n)$
$l^{\min}(v_i, v_j)$	Inter-release time of shortest path from v_i to v_j
(π, ν)	dDRT job of vertex ν released at time π
$\mathcal{D} = [(\pi_1, \nu_1), \ldots, (\pi_n, \nu_n)]$	dDRT job sequence
$\mathcal{D}.I(t)$	Interference function of $\mathcal{D}$ over an interval of length t
$R(\tau_i, \mathcal{D})$	Response time of τ_i interfered by a set of periodic tasks and a dDRT job sequence $\mathcal{D}$
$R(\tau_i, \tau_D^*)$	Response time of τ_i interfered by a set of periodic tasks and a dDRT task τ_D^*
$\tau_D^*(\mathcal{B})$	Transformed dDRT task based on speed partition $\mathcal{B}$
$f_{\mathcal{B}/\mathcal{B}'}(v_i, v_j)$	Edge factor of (v_i, v_j) in $\tau_D^*(\mathcal{B})$ with respect to $\mathcal{B}'$ where $\mathcal{B} \subseteq \mathcal{B}'$
$f_{\mathcal{B}/\mathcal{B}'}^{\max}$	Edge factor of $\mathcal{B}$ with respect to $\mathcal{B}'$ where $\mathcal{B} \subseteq \mathcal{B}'$
$l^{\min}(\beta_i, \beta_j)$	Minimum inter-release time from speed β_i to speed β_j after rotating an integer number of angular periods
$f_{\mathcal{C}}^{\max}$	WCET factor for a dAVR task
$R^u(\tau_i, \tau_D^*)$	Response time of τ_i interfered by a set of periodic tasks and a dDRT task τ_D^* running on a speed-u processor
$\mathbb{C} = [(c_1^-, c_1^+, \nu_1), \ldots, (c_n^-, c_n^+, \nu_n)]$	dDRT job sequence set
$\widetilde{\mathbb{C}} = [(\tilde{c}_1^-, \tilde{c}_1^+, \nu_1), \ldots, (\tilde{c}_n^-, \tilde{c}_n^+, \nu_n)]$	Legalized dDRT job sequence set
$\mathcal{D}^c = [(\pi_1^c, \nu_1), \ldots, (\pi_n^c, \nu_n)]$	Critical dDRT job sequence
$R(\tau_i^*, \sigma, \omega)$	Response time of a job released from τ_i^* at time σ and speed ω
$R(\tau_i^*, \omega)$	Response time of a job released from τ_i^* at speed ω

it as $\omega_k \rightsquigarrow \omega_{k+1}$, and ω_k and ω_{k+1} shall satisfy

$$\Omega\Big(\omega_k, \alpha^{\min}, \Theta\Big) \le \omega_{k+1} \le \Omega\big(\omega_k, \alpha^{\max}, \Theta\big). \tag{2}$$

The minimum (resp. maximum) inter-release time between them is denoted as $T^{\min}(\omega_k, \omega_{k+1})$ [resp. $T^{\min}(\omega_k, \omega_{k+1})$], which can be calculated following the equations in [26].

We denote an *AVR job* as (σ, ω) where σ is its release time and ω is the angular speed at time σ. For any two consecutive AVR jobs (σ_l, ω_l) and $(\sigma_{l+1}, \omega_{l+1})$, they must satisfy

$$\omega_l \rightsquigarrow \omega_{l+1} \wedge T^{\min}(\omega_l, \omega_{l+1}) \le \sigma_{l+1} - \sigma_l \le T^{\max}(\omega_l, \omega_{l+1}). \tag{3}$$

The deadline of (σ, ω) in the time domain, denoted as $D_i(\omega)$, is the minimum time to rotate $\Delta_i = \lambda_i\Theta$ angles [26].

We summarize a few useful properties from the literature.

Property 1: $T^{\min}(\omega_k, \omega_{k+1})$ is strictly decreasing with ω_k and ω_{k+1} [16].

Property 2: $T^{\max}(\omega_k, \omega_{k+1})$ is strictly decreasing with ω_k and ω_{k+1} [31].

Property 3: $D_i(\omega)$ is strictly decreasing with ω [16].

B. sAVR Task Model

The sAVR task model, as proposed in the literature, assumes fixed switching speeds. An sAVR task τ_i^* implements a set $\mathcal{SM}_i$ of SM_i execution modes. Each mode m implements a control strategy characterized by a WCET SC_i^m, and is executed when the angular speed at the task release time is in the range $(\varsigma\omega_i^{m-1}, \varsigma\omega_i^m]$. Here $\varsigma\omega_i^0 = \omega^{\min}$, $\varsigma\omega_i^{SM_i} = \omega^{\max}$, and $\forall m < SM_i$, it is $SC_i^m \ge SC_i^{m+1}$ and $\varsigma\omega_i^m < \varsigma\omega_i^{m+1}$. Hence, the set of execution modes of an sAVR task τ_i^* can be described as

$$\mathcal{SM}_i = \big\{\big(SC_i^m, \varsigma\omega_i^m\big), m = 1, \ldots, SM_i\big\}. \tag{4}$$

The WCET of a job of τ_i^* only depends on the instantaneous angular speed ω at its release time. Hence, we may define a WCET function for the sAVR task τ_i^* as

$$\mathcal{SC}_i(\omega) = SC_i^m \quad \text{if } \omega \in \Big(\varsigma\omega_i^{m-1}, \varsigma\omega_i^m\Big]. \tag{5}$$

C. dAVR Task Model

We now introduce the concept of AVR tasks with dynamic execution modes, where the switching speeds are adjusted at runtime. We assume that the reconfiguration happens at times $\mathcal{T} = \{\gamma_1, \ldots, \gamma_T\}$. The reconfiguration may be triggered by events independent from those activating the periodic or AVR tasks. The associated AVR task τ_i^*, termed as a dAVR task, has a series of execution mode sets defined as

$$\begin{aligned} \mathcal{Q}_i &= \big\{\big(\mathcal{M}_{i,k}, \gamma_k\big), k = 1, \ldots, T\big\}, \text{ where} \\ \mathcal{M}_{i,k} &= \big\{\big(C_{i,k}^m, \omega_{i,k}^m\big), m = 1, \ldots, M_{i,k}\big\}. \end{aligned} \tag{6}$$

The WCET of the job released at time t with instantaneous speed ω is determined as

$$\mathcal{C}_i(t, \omega) = C_{i,k}^m \text{ if } t \in \big[\gamma_k, \gamma_{k+1}\big) \wedge \omega \in \Big(\omega_{i,k}^{m-1}, \omega_{i,k}^m\Big]. \tag{7}$$

We note that an sAVR task can be regarded as a special case of the dAVR task model, by assuming $\gamma_1 = 0$ and $\gamma_2 = +\infty$.

Example 1: Table II shows an illustrative example of a dAVR task τ_i^*. Within [0, 100) ms it uses an execution mode set $\mathcal{M}_{i,1}$ with three modes, while at time 100 ms it switches to an execution mode set $\mathcal{M}_{i,2}$ with four modes.

TABLE II
ILLUSTRATIVE EXAMPLE OF A DAVR TASK τ_i^*

Time Interval (ms)	Execution Modes					
$[\gamma_1, \gamma_2) = [0, 100)$	$\mathcal{M}_{i,1}$	m-th mode	1	2	3	
		$\omega_{i,1}^m$ (rpm)	2500	4500	6500	
		$C_{i,1}^m$ (μs)	600	400	200	
$[\gamma_2, \gamma_3) = [100, 200)$	$\mathcal{M}_{i,2}$	m-th mode	1	2	3	4
		$\omega_{i,2}^m$ (rpm)	1500	2500	4500	6500
		$C_{i,2}^m$ (μs)	600	400	300	200
$[\gamma_3, \gamma_4) = [200, +\infty)$	$\mathcal{M}_{i,3}$	m-th mode	1	2		
		$\omega_{i,3}^m$ (rpm)	3500	6500		
		$C_{i,3}^m$ (μs)	600	300		

III. SCHEDULABILITY ANALYSIS OF PERIODIC TASKS

Let $hp(i)$ ($hp^*(i)$) denote the set of periodic (dAVR) tasks with higher priority than the periodic task τ_i under analysis. With the assumption that the dAVR tasks share the same angular period and phase, we can construct a representative dAVR task τ_A^* to model the accumulative workload of tasks from $hp^*(i)$. For each time interval $[\gamma_k, \gamma_{k+1})$ within which the execution modes for any task in $hp^*(i)$ remain the same, the set of execution modes and their WCETs for τ_A^* can be constructed in the same way as those of sAVR tasks, i.e., with the procedure in [15]. In the following, we focus on the analysis of τ_i interfered by a set of periodic tasks $hp(i)$ and a (representative) dAVR task τ_A^*.

We first establish an exact schedulability analysis method, based on an exhaustive enumeration of all job sequences of τ_A^*. We define two useful concepts for a dAVR task, namely a dAVR job sequence and its interference function.

Definition 1 (dAVR Job Sequence): A job sequence $\mathcal{A} = [(\sigma_1, \omega_1), \ldots, (\sigma_n, \omega_n)]$ released by a dAVR task τ_A^*, written as $\mathcal{A} \in \tau_A^*$, is composed of a legal sequence of jobs, such that any two consecutive jobs (σ_l, ω_l) and $(\sigma_{l+1}, \omega_{l+1})$, $\forall l = 1, \ldots, n-1$ satisfy (3).

Definition 2 (Interference Function of dAVR Job Sequence): $\forall t > 0$, the interference function $\mathcal{A}.I(t)$ of a dAVR job sequence $\mathcal{A} = [(\sigma_1, \omega_1), \ldots, (\sigma_n, \omega_n)]$ in τ_A^* is its cumulative execution request within the interval $[\sigma_1, \sigma_1 + t)$. That is

$$\mathcal{A}.I(t) = \mathcal{C}_A(\sigma_1, \omega_1) + \sum_{l=2}^{n} \delta(\sigma_1 + t, \sigma_l) \cdot \mathcal{C}_A(\sigma_l, \omega_l) \quad (8)$$

where function $\delta(\cdot, \cdot)$ is defined as

$$\delta(a, b) = \begin{cases} 1 & \text{if } a > b \\ 0 & \text{otherwise.} \end{cases}$$

We now discuss how to calculate the response time of τ_i. We note that the periodic tasks and the dAVR tasks are triggered by independent sources. Hence, the worst-case response time (WCRT) of τ_i occurs when it is released simultaneously with all its interfering tasks. The WCRT $R(\tau_i, \mathcal{A})$ of τ_i interfered by a set of periodic tasks $hp(i)$ and a dAVR job sequence $\mathcal{A} = [(\sigma_1, \omega_1), \ldots, (\sigma_n, \omega_n)]$ of τ_A^* is achieved when τ_i is released together with $\mathcal{A}$ (i.e., at σ_1), and all periodic tasks in $hp(i)$ are also released at σ_1[1]

$$R(\tau_i, \mathcal{A}) = \min_{t>0} \left\{ t \mid C_i + \sum_{\tau_j \in hp(i)} \left\lceil \frac{t}{T_j} \right\rceil C_j + \mathcal{A}.I(t) \le t \right\}. \quad (9)$$

[1]Note that in this article time 0 is assumed to be the start of the engine operation, and the critical instant in the schedulability analysis is σ_1, which is in general nonzero. This shall not affect the soundness of the analysis.

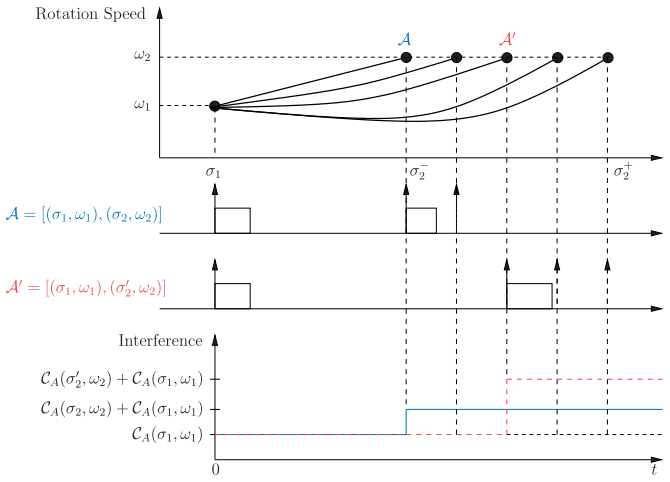


Fig. 1. Illustration of two job sequences $\mathcal{A}$ and $\mathcal{A}'$ released by a dAVR task, and the corresponding interference functions.

Note in (9), under certain conditions (e.g., if the utilization is $> 100\%$) no such t exists, and $R(\tau_i, \mathcal{A})$ is defined as infinity. The WCRT $R(\tau_i, \tau_A^*)$ of τ_i is the maximum over all possible dAVR job sequences of τ_A^*

$$R(\tau_i, \tau_A^*) = \max_{\mathcal{A} \in \tau_A^*} R(\tau_i, \mathcal{A}). \quad (10)$$

However, the analysis in (10) is obviously impractical as the number of dAVR job sequences is infinite (due to the continuous spaces for both job release time and angular speed). In the following we develop a safe, but sufficient-only analysis.

Before detailing our techniques, we first highlight that the existing methods developed for sAVR tasks are no longer safe for dAVR tasks. Specifically, consider two job sequences $\mathcal{A} = [(\sigma_1, \omega_1), (\sigma_2, \omega_2), \ldots, (\sigma_n, \omega_n)]$ and $\mathcal{A}' = [(\sigma_1, \omega_1), (\sigma_2', \omega_2), \ldots, (\sigma_n', \omega_n)]$ from an AVR task, such that $\sigma_l \le \sigma_l', \forall l = 2, \ldots, n$. In other words, $\mathcal{A}$ and $\mathcal{A}'$ release jobs at the same sequence of angular speeds, but jobs in $\mathcal{A}$ are always released no later than $\mathcal{A}'$. The analysis presented in [15], [16], and [26] will only consider $\mathcal{A}$. This is safe for sAVR tasks, since the WCET of an sAVR job is independent from its release time (5) and consequently

$$\forall t, \mathcal{A}.I(t) \le \mathcal{A}'.I(t). \quad (11)$$

This dominance relationship can be generalized to two job sequences released at different angular speeds but sharing the same sequence of job WCETs. Combining Property 1, it leads to the concept of dominant speed, a speed that dominates a range of smaller speeds whenever they always produce job sequences with the same sequence of job WCETs [14]. Thus, the dominant speed allows shorter inter-release times than the dominated ones while matching their sequence of job WCETs.

However, as in (7) the WCET of a dAVR job also depends on its actual release time. Hence, (11) no longer holds for dAVR tasks, and consequently the analysis developed for sAVR task systems [14], [26] is not directly applicable. An illustrative example is shown below.

Example 2: Fig. 1 illustrates two dAVR job sequences $\mathcal{A} = [(\sigma_1, \omega_1), (\sigma_2, \omega_2)]$ and $\mathcal{A}' = [(\sigma_1, \omega_1), (\sigma_2', \omega_2)]$ ($\sigma_2 < \sigma_2'$). Let $\sigma_2^- = \sigma_1 + T^{\min}(\omega_1, \omega_2)$ and $\sigma_2^+ = \sigma_1 + T^{\max}(\omega_1, \omega_2)$. We assume $\mathcal{C}_A(\sigma_2, \omega_2) < \mathcal{C}_A(\sigma_2', \omega_2)$, which is possible under the dAVR task model. The interference functions of $\mathcal{A}$ (denoted

as solid blue line) and $\mathcal{A}'$ (dashed red line) are also illustrated in the figure, which obviously violate (11). Hence, the maximum job inter-release times are needed to correctly model the execution of dAVR tasks and calculate the interference function.

We now present our new analysis techniques. Specifically, to avoid enumerating the speed in the (continuous) speed space, we partition it into a finite number of speed intervals, and transform a dAVR task to a new type of dDRT task model where each vertex represents each of the partitioned speed intervals, and the edges are labeled with both minimum and maximum inter-release times (Section III-A). We then prove the speed space partition (hence any transformation to dDRT task) is safe but sufficient only (Section III-B). We also demonstrate that the pessimism of the transformation is bounded (Section III-C). Finally, to avoid exhaustive enumeration of the job release times, we study the dominance relationship between dDRT job sequences (Section III-D).

We note that the dominant speeds [14] implicitly partition the speed space: they find a set of dominant speeds, each of which represents a speed interval in terms of the worst-case interference. In this article, we leverage the more explicit approach of speed partition and task transformation in [26], for its intuitive graphical representation.

A. dAVR to dDRT Transformation

The DRT task model [27], [28] uses a directed graph to model a real-time task, where the vertices represent the types of jobs, and the edges represent possible flows of control. As a suitable model for sAVR tasks, each vertex v_i represents a speed interval completely contained in the speed interval of an execution mode, hence is characterized by a constant WCET. Each edge is labeled with a parameter $p(v_i, v_j)$ that denotes the minimum separation time between the releases of v_i and v_j. By (11), this is sufficient.

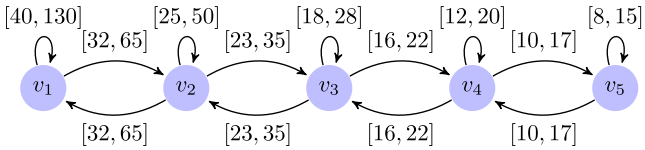
However, as explained above, we cannot assume jobs are released with minimum inter-release times for dAVR tasks. Also, the WCET of a dAVR task is a function of time to model the fact that it also depends on the job release time (7). This is formalized in the definition below.

Definition 3 (dDRT): A dDRT τ_D^* is characterized by a directed graph $(\mathbb{V}, \mathbb{E})$, where the set of vertices $\mathbb{V}$ represents the types of jobs of τ_D^*. Each vertex $v_i \in \mathbb{V}$ (or type of job) is characterized by a WCET function $v_i.\mathcal{C}(t)$, where t denote the release time of the job of v_i. Edges $\mathbb{E}$ represent possible flows of control, i.e., the release order of the jobs of τ_D^*. Each edge $(v_i, v_j) \in \mathbb{E}$ is labeled with a range $[p^{\min}(v_i, v_j), p^{\max}(v_i, v_j)]$, where $p^{\min}(v_i, v_j)$ (resp. $p^{\max}(v_i, v_j)$) denotes the minimum (resp. maximum) time between the releases of v_i and v_j.

By the definition, the dDRT task model is a generalization of the DRT task model.

We now explain how to use dDRT to discretize the continuous space of engine speeds and capture the switch among the control strategies. We define a speed partition and the corresponding task transformation where: 1) each speed interval is mapped to a distinct vertex in the dDRT task and 2) the control strategy (and thus WCET) of a vertex is the same for any speed in the corresponding speed interval.

Definition 4 (Valid Speed Partition): For a dAVR task τ_A^*, a *valid* speed partition $\mathcal{B}$ defines a set of speed intervals $\{(\beta_0, \beta_1], \ldots, (\beta_{B-1}, \beta_B]\}$ $(\forall i \leq B, \beta_{i-1} < \beta_i)$ that satisfy the following.



Time Interval (ms)	WCET (μs)				
	v_1	v_2	v_3	v_4	v_5
$[\gamma_1, \gamma_2) = [0, 100)$	600	600	400	400	200
$[\gamma_2, \gamma_3) = [100, 200)$	600	400	300	300	200
$[\gamma_3, \gamma_4) = [200, +\infty)$	600	600	600	300	300

Fig. 2. Transformed dDRT task for the dAVR task in Table II with the speed partition $\mathcal{B} = \{500, 1500, 2500, 3500, 4500, 6500\}$ (top). The WCET function of each vertex (bottom).

1) They partition the complete speed range $(\omega^{\min}, \omega^{\max}]$ (hence $\beta_0 = \omega^{\min}$, $\beta_B = \omega^{\max}$).
2) For any two speeds ω and ω' belonging to the same speed interval, the WCET functions are the same, i.e., $\forall i \leq B$, $\forall \omega \in (\beta_{i-1}, \beta_i], \omega' \in (\beta_{i-1}, \beta_i]$, $\forall t \geq 0$, $\mathcal{C}_A(t, \omega) = \mathcal{C}_A(t, \omega')$.

For convenience, we also use the *ordered* set of boundary speeds to denote $\mathcal{B}$, i.e., $\mathcal{B} = \{\beta_0, \beta_1, \ldots, \beta_B\}$. By the second condition in Definition 4, the smallest valid speed partition for τ_A^* consists of all switching speeds and the two speed limits

$$\mathcal{B} = \left\{\omega_{A,k}^m | \forall k = 1 \cdots T, \ \forall m = 1 \cdots M_{A,k}\right\} \cup \left\{\omega^{\min}, \omega^{\max}\right\}. \tag{12}$$

Given a valid speed partition $\mathcal{B} = \{\beta_0, \beta_1, \ldots, \beta_B\}$, the dDRT task $\tau_D^*(\mathcal{B}) = (\mathbb{V}, \mathbb{E})$ can be constructed as follows (in case there is no confusion, we also drop the partition $\mathcal{B}$ from the notation of dDRT task and simply denote it as τ_D^*).

1) $\mathbb{V}$ is composed of a set of B vertices $\{v_1, \ldots, v_B\}$, where the speed interval $(\beta_{i-1}, \beta_i]$ is mapped to vertex v_i. Each vertex v_i is labeled with the WCET function $\mathcal{C}_A(t, \omega)$, i.e., $v_i.\mathcal{C}(t) = \mathcal{C}_A(t, \omega)$, where ω is any speed in $(\beta_{i-1}, \beta_i]$.
2) For each two vertices v_i and v_j (which may be the same), if $\beta_{j-1} < \Omega(\beta_i, \alpha^{\max}, \Theta)$ and $\beta_j > \Omega(\beta_{i-1}, \alpha^{\min}, \Theta)$, then there must exist $\beta_i' \in (\beta_{i-1}, \beta_i]$ and $\beta_j' \in (\beta_{j-1}, \beta_j]$ such that $\beta_i' \rightsquigarrow \beta_j'$. In this case, we add an edge (v_i, v_j) to the set $\mathbb{E}$, and label it with $[p^{\min}(v_i, v_j), p^{\max}(v_i, v_j)]$ where

$$\begin{cases} p^{\min}(v_i, v_j) = \min\limits_{\forall \beta_i' \rightsquigarrow \beta_j'} \left\{T^{\min}(\beta_i', \beta_j')\right\} \\ p^{\max}(v_i, v_j) = \max\limits_{\forall \beta_i' \rightsquigarrow \beta_j'} \left\{T^{\max}(\beta_i', \beta_j')\right\}. \end{cases} \tag{13}$$

These two parameters can be efficiently calculated as in [26].

Example 3: Considering the dAVR task in Table II and the speed partition $\mathcal{B} = \{500, 1500, 2500, 3500, 4500, 6500\}$ revolutions per minute (rpm). $\mathcal{B}$ is the smallest valid speed partition, and Fig. 2 gives the transformed dDRT task, where vertices $v_1, \ldots, v_5$ represent the intervals $(500, 1500]$, $(1500, 2500]$, $(2500, 3500]$, $(3500, 4500]$, $(4500, 6500]$, respectively. The minimum and maximum inter-release times are labeled along the edges. For example, the inter-release time from v_4 to v_5 must be within $[10, 17]$ ms. The WCET function of each vertex is shown in the table. The inter-release times are simplified for illustration purposes and may not match the actual rotational dynamics.

Since there is a total order among the (nonoverlapping) speed intervals in the partition, we can use it to define an order among vertices in the transformed dDRT.

Definition 5 (Vertex Order): For any two *distinguished* vertices v_i and v_j in the dDRT task τ_D^* which, respectively, correspond to the speed intervals $(\beta_{i-1}, \beta_i]$ and $(\beta_{j-1}, \beta_j]$, if $\beta_{i-1} \geq \beta_j$, we say v_i is larger than v_j, denoted as $v_i \succ v_j$; if $\beta_i \leq \beta_{j-1}$, we say v_i is smaller than v_j, denoted as $v_i \prec v_j$.

We now provide a useful property on the dDRT task.

Lemma 1: For any three vertices v_i, v_j and v_k in the dDRT task $\tau_D^* = (\mathbb{V}, \mathbb{E})$ where $v_i \prec v_j \prec v_k$, if $(v_i, v_k) \in \mathbb{E}$, then it must be that $(v_j, v_k) \in \mathbb{E}$ and $p^{\min}(v_i, v_k) > p^{\min}(v_j, v_k)$.

Proof: Let the speed intervals $(\beta_{i-1}, \beta_i]$, $(\beta_{j-1}, \beta_j]$ and $(\beta_{k-1}, \beta_k]$, respectively, correspond to v_i, v_j and v_k. Since $(v_i, v_k) \in \mathbb{E}$, by the construction procedure of τ_D^*, there must exist $\omega_i \in (\beta_{i-1}, \beta_i]$ and $\omega_k \in (\beta_{k-1}, \beta_k]$, such that $\omega_i \rightsquigarrow \omega_k$. Note that $\beta_i < \beta_j \leq \beta_{k-1}$ (due to $v_i \prec v_j \prec v_k$), it is $\omega_i \leq \beta_i < \beta_j \leq \beta_{k-1} \leq \omega_k$. By (2), we have $\beta_j \rightsquigarrow \omega_k$ and consequently $(v_j, v_k) \in \mathbb{E}$. In addition, by Property 1, it also holds that $p^{\min}(v_i, v_k) > p^{\min}(v_j, v_k)$. ■

We now study how the shortest path in the dDRT task can be constructed. Lemma 2 provides the special case $(v_i, v_j) \in \mathbb{E}$ before considering the general case in Theorem 1.

Definition 6 (Path Length): Given a path $(v_1, \ldots, v_n)$ in $\tau_D^* = (\mathbb{V}, \mathbb{E})$, the path length of $(v_1, \ldots, v_n)$ is defined as the cumulative minimum inter-release time of all edges in the path, denoted as $l(v_1, \ldots, v_n) = \sum_{l=1}^{n-1} p^{\min}(v_l, v_{l+1})$.

Definition 7 (Shortest Path): Given two vertices v_i and v_j in $\tau_D^* = (\mathbb{V}, \mathbb{E})$, the shortest path from v_i to v_j is defined as the path with a length no larger than any other path from v_i to v_j. The length of such a path is denoted as $l^{\min}(v_i, v_j)$.

Lemma 2: Given the transformed dDRT task $\tau_D^* = (\mathbb{V}, \mathbb{E})$, for any edge $(v_i, v_j) \in \mathbb{E}$, edge (v_i, v_j) is the shortest path from v_i to v_j, i.e., $l^{\min}(v_i, v_j) = p^{\min}(v_i, v_j)$.

Proof: Assume the intervals for v_i and v_j are $(\beta_{i-1}, \beta_i]$ and $(\beta_{j-1}, \beta_j]$, respectively. For convenience, we denote

$$\Omega_m = \Omega\Big(\beta_{i-1}, \alpha^{\min}, \Theta\Big) = \sqrt{\beta_{i-1}^2 + 2\alpha^{\min}\Theta}$$
$$\Omega_{M^-} = \Omega\Big(\beta_i, \alpha^{\min}, \Theta\Big) = \sqrt{\beta_i^2 + 2\alpha^{\min}\Theta}$$
$$\Omega_{M^+} = \Omega\big(\beta_i, \alpha^{\max}, \Theta\big) = \sqrt{\beta_i^2 + 2\alpha^{\max}\Theta}.$$

We now discuss the following four cases.

Case 1 ($\Omega_{M^+} \leq \beta_{j-1}$ or $\Omega_m > \beta_j$): This implies that there are no speeds $\omega_i \in (\beta_{i-1}, \beta_i]$ and $\omega_j \in (\beta_{j-1}, \beta_j]$ such that $\omega_i \rightsquigarrow \omega_j$, which contradicts the fact that $(v_i, v_j) \in \mathbb{E}$.

Case 2 ($\beta_{j-1} < \Omega_{M^+} \leq \beta_j$): This means $\Omega_{M^+} \in (\beta_{j-1}, \beta_j]$ is the maximum speed that β_i can accelerate to in one angular period, and there is no vertex v_k such that $(v_i, v_k) \in \mathbb{E}$ and $v_k \succ v_j$. By Property 1, no other outgoing edge of v_i has a smaller minimum inter-release time than $p^{\min}(v_i, v_j)$. Hence, any other path from v_i to v_j must be longer than $p^{\min}(v_i, v_j)$.

Case 3 ($\Omega_m < \beta_j \leq \Omega_{M^-}$): Thus, the minimum speed that any speed $\omega > \beta_i$ can decelerate to in one angular period is higher than β_j. That is, there is no vertex v_k such that $(v_k, v_j) \in \mathbb{E}$ and $v_k \succ v_i$. By Property 1, no other incoming edge of v_j has a smaller minimum inter-release time than $p^{\min}(v_i, v_j)$, and any other path has a longer length than $p^{\min}(v_i, v_j)$.

Case 4 ($\Omega_{M^-} \leq \beta_j \leq \Omega_{M^+}$): This means that β_i can reach β_j in one angular period, thus $p^{\min}(v_i, v_j) = T^{\min}(\beta_i, \beta_j)$. By the definition of $T^{\min}(\beta_i, \beta_j)$ [26], the minimum inter-release time by rotating only one angular period is smaller than that of multiple periods. Hence, any other path, which has to reach from v_i to v_j through multiple edges (thus taking multiple periods) has a length larger than $p^{\min}(v_i, v_j)$.

Combining the above four cases, also noting that (v_i, v_j) is a path from v_i to v_j, we have $l^{\min}(v_i, v_j) = p^{\min}(v_i, v_j)$. ■

Algorithm 1 Constructing Shortest Path From v_i to v_j

C1: if $v_i = v_j$, then return the path (v_i, v_j).
C2: if $v_i \prec v_j$, then
1) $v_0 \leftarrow v_i$, $k = 0$.
2) $k = k + 1$; if $(v_{k-1}, v_j) \in \mathbb{E}$, then $v_k \leftarrow v_j$ and return the path $(v_0, \ldots, v_k)$; else $v_k \leftarrow \underset{v:(v_{k-1}, v) \in \mathbb{E}}{\arg\min}\ p^{\min}(v_{k-1}, v)$; repeat Step 2).

C3: if $v_i \succ v_j$, then
1) $v_0 \leftarrow v_j$, $k = 0$.
2) $k = k + 1$; if $(v_i, v_{k-1}) \in \mathbb{E}$, then $v_k \leftarrow v_i$ and return the path $(v_k, \ldots, v_0)$; else $v_k \leftarrow \underset{v:(v, v_{k-1}) \in \mathbb{E}}{\arg\min}\ p^{\min}(v, v_{k-1})$; repeat Step 2).

For any pair of vertices v_i and v_j, we construct a path with Algorithm 1. Intuitively, when $v_i \prec v_j$, the path is constructed forward from v_i, by accelerating from v_i at the maximum acceleration until reaching v_j. When $v_i \succ v_j$, it is constructed similarly but backward from v_j. Theorem 1 shows that indeed Algorithm 1 constructs the shortest path from v_i to v_j.

Theorem 1: For any two vertices v_i and v_j in the transformed dDRT task $\tau_D^* = (\mathbb{V}, \mathbb{E})$, the path constructed in Algorithm 1 must be the shortest from v_i to v_j.

Proof: According to the algorithm, we discuss three cases.

C1: Obviously $(v_i, v_j) \in \mathbb{E}$ and hence by Lemma 2, (v_i, v_j) is the shortest path.

C2: Suppose the returned path is $(v_0 = v_i, \ldots, v_k = v_j)$. For any $l = 1, \ldots, k-1$, since edge (v_{l-1}, v_l) has the smallest minimum inter-release time among the outgoing edges from v_{l-1}, v_l is reached from v_{l-1} by accelerating with the maximum acceleration. Hence, the shortest path from v_i to v_j has at least k edges and $k+1$ vertices.

Now we consider any other path P' where the first k vertices are $v_i, v'_1, \ldots, v'_{k-1}$. By Property 1, since $p^{\min}(v_i, v_1) \leq p^{\min}(v_i, v'_1)$, it must be $v'_1 \preceq v_1$. By induction, we have $v'_{k-1} \preceq v_{k-1}$. If $v_{k-1} = v'_{k-1}$, by Lemma 2

$$\begin{aligned} l\big(v_i, v_1, \ldots, v_{k-1}, v_j\big) &= l(v_i, v_1, \ldots, v_{k-1}) + p^{\min}\big(v_{k-1}, v_j\big) \\ &\leq l\big(v_i, v'_1, \ldots, v'_{k-1}\big) + l^{\min}\big(v'_{k-1}, v_j\big) \\ &= l\big(v_i, v'_1, \ldots, v'_{k-1}, v_j\big). \end{aligned}$$

Otherwise $v_{k-1} \succ v'_{k-1}$. There are two cases: 1) $(v'_{k-1}, v_j) \in \mathbb{E}$, by Lemma 1 we have $p^{\min}(v_{k-1}, v_j) < p^{\min}(v'_{k-1}, v_j)$ and 2) $(v'_{k-1}, v_j) \notin \mathbb{E}$, that is, v_j is not reachable from v'_{k-1}, and the minimum inter-release time of any edge from v'_{k-1} must be $> p^{\min}(v_{k-1}, v_j)$. Both 1) and 2) imply

$$\begin{aligned} l(v_i, v_1, \ldots, v_{k-1}) &< l\big(v_i, v'_1, \ldots, v'_{k-1}\big) \\ p^{\min}\big(v_{k-1}, v_j\big) &< l^{\min}\big(v'_{k-1}, v_j\big). \end{aligned}$$

Hence, P' must be longer than the path from Algorithm 1.

C3: This can be proved similarly as the case C2. ■

B. Transformation to dDRT Is Sufficient-Only

Before studying the properties of the task transformation, we first establish how the schedulability analysis can be performed with the transformed dDRT task.

Definition 8 (dDRT Job Sequence): A job of a dDRT task τ_D^* is denoted as (π_l, ν_l) where π_l and ν_l are the job release time and vertex (type of job), respectively. A dDRT job sequence $\mathcal{D} = [(\pi_1, \nu_1), \ldots, (\pi_n, \nu_n)]$ of τ_D^* is composed of a sequence of jobs (π_l, ν_l) such that $\forall l < n$

$$(\nu_l, \nu_{l+1}) \in \mathbb{E} \wedge p^{\min}(\nu_l, \nu_{l+1}) \leq \pi_{l+1} - \pi_l \leq p^{\max}(\nu_l, \nu_{l+1}).$$

For convenience, we also denote $\mathcal{D} \in \tau_D^*$, and regard τ_D^* as the *set of all its job sequences*.

Definition 9 (Interference Function of dDRT Job Sequence): For a dDRT job sequence $\mathcal{D} = [(\pi_1, \nu_1), \ldots, (\pi_n, \nu_n)]$ of τ_D^*, the cumulative execution request within the time window $[\pi_1, \pi_1 + t)$ is defined as its interference function $\mathcal{D}.I(t)$, i.e.,

$$\forall t \geq 0, \quad \mathcal{D}.I(t) = \nu_1.\mathcal{C}(\pi_1) + \sum_{l=2}^{n} \delta(\pi_1 + t, \pi_l)\ \nu_l.\mathcal{C}(\pi_l). \tag{14}$$

With these two definitions, similar to (9) and (10), the WCRT of a periodic task τ_i interfered by a dDRT task τ_D^* and a set of higher priority periodic tasks $hp(i)$ is

$$R(\tau_i, \tau_D^*) = \max_{\mathcal{D} \in \tau_D^*} R(\tau_i, \mathcal{D}), \quad \text{where}$$

$$R(\tau_i, \mathcal{D}) = \min_{t>0} \left\{ t \mid C_i + \sum_{\tau_j \in hp(i)} \left\lceil \frac{t}{T_j} \right\rceil C_j + \mathcal{D}.I(t) \leq t \right\}. \tag{15}$$

We now study the task transformation in terms of the following two desired properties.

Definition 10 (Safe Transformation): The transformation is safe if for any dAVR task system the schedulability based on the transformed dDRT task entails that of the original system.

Definition 11 (Exact Transformation): The transformation is exact if the schedulability of any dAVR task system and that of its transformed dDRT task system entail each other.

Intuitively, if for any dAVR job sequence $\mathcal{A}$, we can find a job sequence $\mathcal{D}$ in the transformed dDRT task such that their interference functions satisfy $\forall t, \mathcal{D}(t) \geq \mathcal{A}(t)$, then the task transformation must be safe. For a safe transformation, if for any job sequence $\mathcal{D}$ in the transformed dDRT task, we can find a dAVR job sequence $\mathcal{A}$ such that $\forall t, \mathcal{A}(t) \geq \mathcal{D}(t)$, then the task transformation must be exact. These properties are leveraged in the following two theorems.

Theorem 2: The task transformation with any valid speed partition is safe.

This theorem demonstrates that the analysis with the transformed dDRT task provides an upper bound on the WCRT of τ_i interfered by the dAVR task. However, the proposed transformation is not exact, as shown in Theorem 3. The proofs of Theorems 2 and 3 are omitted and can be found in [32].

Theorem 3: The task transformation with any valid speed partition is inexact for any rotation source with $\omega^{\max} > \omega^t = \sqrt{(25/12) \cdot ([-\alpha^{\max}\alpha^{\min}\Theta]/[\alpha^{\max} - \alpha^{\min}])}$.

Remark 1: We remark that the condition $\omega^{\max} > \omega^t = \sqrt{(25/12) \cdot ([-\alpha^{\max}\alpha^{\min}\Theta]/[\alpha^{\max} - \alpha^{\min}])}$ in Theorem 3 is satisfied by typical engine dynamics. According to [15], the maximum acceleration/deceleration are typically selected to be able to accelerate/decelerate between the minimum and maximum speeds in about 35 revolutions. By (1), $\omega^{\max} = \sqrt{(\omega^{\min})^2 + 70\alpha^{\max}\Theta} \geq \sqrt{70\alpha^{\max}\Theta}$. Meanwhile, $\omega^t = \sqrt{(25/12) \cdot ([-\alpha^{\max}\alpha^{\min}\Theta]/[\alpha^{\max} - \alpha^{\min}])} \leq \sqrt{(25/12)\alpha^{\max}\Theta}$. Hence, $\omega^{\max} > 5.79\ \omega^t$.

C. Bounded Pessimism of Task Transformation

Theorem 3 essentially states that the analysis with the transformed tasks is always pessimistic (under the typical engine dynamics). This means that there is always a price we need to pay if we use dDRT tasks to approximate the dAVR tasks. Such a price is necessary since we cannot afford to enumerate the (infinitely many) speeds in the continuous speed space. However, Theorem 3 does not quantify how large the pessimism is. This section will provide an answer to that with the metric of speedup factor [33], a popular method for evaluating the accuracy of a schedulability analysis in real-time systems. A schedulability analysis S has a speedup factor of $u \geq 1$ if the task set schedulable on a processor of speed 1 will be deemed schedulable by S when the processor's speed is increased to u [33]. This means that to adopt the (pessimistic) analysis S, the resources shall be augmented such that the task WCETs are all shortened by a factor of u. Hence, a bounded speedup factor u means that the price of adopting S is bounded in term of the required resource augmentation.

We prove two significant theoretical results in this section. First, we show that the task transformation has a problem-dependent speedup factor that is always bounded, hence the task transformation always has bounded pessimism (Theorem 6). Second, we prove that the analysis on the transformed dDRT task system with a finer speed partition is at least as accurate (Theorem 5). This implies a tradeoff between the analysis runtime and accuracy: the finer the speed partition, the better the analysis accuracy; on the other hand, however, a task transformation with finer speed partition also results in a larger dDRT task and consequently longer analysis runtime. We will experimentally study this tradeoff in Section V.

Below we first study two valid speed partitions where one is completely contained in the other.

Definition 12 (Subset Relation of Speed Partition): Given two valid speed partitions $\mathcal{B}$ and $\mathcal{B}'$, we say $\mathcal{B}$ is a subset of $\mathcal{B}'$ (denoted as $\mathcal{B} \subseteq \mathcal{B}'$), if $\forall \beta \in \mathcal{B}$ it is $\beta \in \mathcal{B}'$.

Any two speed partitions $\mathcal{B}$ and $\mathcal{B}'$ with $\mathcal{B} \subseteq \mathcal{B}'$ should have the following properties.

Lemma 3: If $\mathcal{B} \subseteq \mathcal{B}'$, for any speed interval $(\beta'_{k-1}, \beta'_k] \in \mathcal{B}'$, there must exist a speed interval $(\beta_{k-1}, \beta_k] \in \mathcal{B}$, such that $\beta_{k-1} \leq \beta'_{k-1} < \beta'_k \leq \beta_k$.

Proof: We construct such a speed interval by letting

$$\beta_{k-1} = \max\{\beta \mid \beta \in \mathcal{B}, \beta \leq \beta'_{k-1}\}$$
$$\beta_k = \min\{\beta \mid \beta \in \mathcal{B}, \beta \geq \beta'_k\}. \tag{16}$$

We prove $(\beta_{k-1}, \beta_k]$ is indeed a speed interval in $\mathcal{B}$, by showing that no other speed β in the partition $\mathcal{B}$ satisfies $\beta_{k-1} < \beta < \beta_k$. For any $\beta \in \mathcal{B}$, if $\beta_{k-1} < \beta$, it must be $\beta'_{k-1} < \beta$. Otherwise, it violates the definition of β_{k-1}. Likewise, if $\beta_k > \beta$, it must be $\beta'_k > \beta$. Hence, any speed $\beta \in \mathcal{B}$ that satisfies $\beta_{k-1} < \beta < \beta_k$ also satisfies $\beta'_{k-1} < \beta < \beta'_k$. However, as $\beta \in \mathcal{B} \subseteq \mathcal{B}'$, this contradicts the fact that $(\beta'_{k-1}, \beta'_k]$ is a speed interval in $\mathcal{B}'$. ■

Lemma 3 directly implies a uni-directional mapping from vertices in $\mathcal{B}'$ to those in $\mathcal{B}$ if $\mathcal{B} \subseteq \mathcal{B}'$, as stated below.

Lemma 4: If $\mathcal{B} \subseteq \mathcal{B}'$, for any vertex v'_k in the dDRT task $\tau_D^*(\mathcal{B}')$ with a corresponding speed interval $(\beta'_{k-1}, \beta'_k]$, there exists a vertex v_k in $\tau_D^*(\mathcal{B})$ whose speed interval $(\beta_{k-1}, \beta_k]$ is constructed as in (16).

The mapping can be generalized to the edges in $\mathcal{B}'$ and $\mathcal{B}$.

Lemma 5: If $\mathcal{B} \subseteq \mathcal{B}'$, for any edge (v'_i, v'_j) in the dDRT task $\tau_D^*(\mathcal{B}')$, define the corresponding vertices of v'_i and v'_j in

$\tau_D^*(\mathcal{B})$ as v_i and v_j, respectively. Then there must exist an edge (v_i, v_j) in $\tau_D^*(\mathcal{B})$ with

$$p^{\min}(v_i, v_j) \le p^{\min}(v'_i, v'_j) < p^{\max}(v'_i, v'_j) \le p^{\max}(v_i, v_j).$$

Proof: Let $(\beta'_{i-1}, \beta'_i]$ and $(\beta'_{j-1}, \beta'_j]$ be the intervals corresponding to v'_i and v'_j, respectively. By Lemma 4, the intervals $(\beta_{i-1}, \beta_i]$ and $(\beta_{j-1}, \beta_j]$ corresponding to v_i and v_j, respectively, satisfy $(\beta'_{i-1}, \beta'_i] \subseteq (\beta_{i-1}, \beta_i]$, $(\beta'_{j-1}, \beta'_j] \subseteq (\beta_{j-1}, \beta_j]$.

Since (v'_i, v'_j) is an edge in $\tau_D^*(\mathcal{B}')$, there must exist a pair of speeds $\omega'_i \in (\beta'_{i-1}, \beta'_i] \subseteq (\beta_{i-1}, \beta_i]$, $\omega'_j \in (\beta'_{j-1}, \beta'_j] \subseteq (\beta_{j-1}, \beta_j]$, such that $\omega'_i \rightsquigarrow \omega'_j$. Hence, v_j contains the speed ω'_i that can reach ω'_j contained in v_j in one angular period, and $\tau_D^*(\mathcal{B})$ must contain the edge (v_i, v_j).

Further, assume the pair $\omega'_i \rightsquigarrow \omega'_j$ such that $p^{\min}(v'_i, v'_j) = T^{\min}(\omega'_i, \omega'_j)$. Since $\omega'_i \in (\beta_{i-1}, \beta_i]$, $\omega'_j \in (\beta_{j-1}, \beta_j]$, it is

$$p^{\min}(v'_i, v'_j) = T^{\min}(\omega'_i, \omega'_j) \ge \min_{\substack{\forall \omega_i \rightsquigarrow \omega_j, \\ \omega_i \in (\beta_{i-1}, \beta_i], \\ \omega_j \in (\beta_{j-1}, \beta_j]}} \left\{ T^{\min}(\omega_i, \omega_j) \right\} = p^{\min}(v_i, v_j).$$

Likewise, we can prove $p^{\max}(v_i, v_j) \ge p^{\max}(v'_i, v'_j)$. ∎

We now prove that a speed partition $\mathcal{B}'$ provides an analysis at least as accurate as any of its subsets $\mathcal{B} \subseteq \mathcal{B}'$, i.e., the system which is deemed schedulable by the analysis based on $\mathcal{B}$ must also be schedulable by adopting the analysis based on $\mathcal{B}'$.

Theorem 4: Given two valid speed partitions $\mathcal{B}$ and $\mathcal{B}'$ with $\mathcal{B} \subseteq \mathcal{B}'$, $\mathcal{B}'$ provides an analysis that is at least as accurate as $\mathcal{B}$, i.e., $R(\tau_i, \tau_D^*(\mathcal{B}')) \le R(\tau_i, \tau_D^*(\mathcal{B}))$.

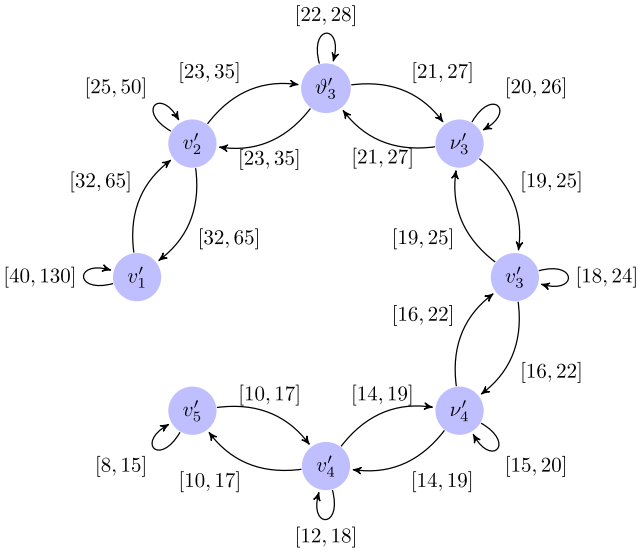
Proof: Consider an arbitrary dDRT job sequence of $\tau_D^*(\mathcal{B}')$, $\mathcal{D}' = [(\pi_1, \nu'_1), \ldots, (\pi_n, \nu'_n)] \in \tau_D^*(\mathcal{B}')$. For each $(\pi_k, \nu'_k) \in \mathcal{D}'$, suppose the speed interval of ν'_k is $(\beta'_{k-1}, \beta'_k]$. Since $\mathcal{B} \subseteq \mathcal{B}'$, by Lemma 4, in $\tau_D^*(\mathcal{B})$ there are a vertex ν_k and its corresponding speed interval $(\beta_{k-1}, \beta_k]$ such that $(\beta'_{k-1}, \beta'_k] \subseteq (\beta_{k-1}, \beta_k]$.

Since $\forall j = 2 \ldots n$, $p^{\min}(\nu'_{j-1}, \nu'_j) \le \pi_j - \pi_{j-1} \le p^{\max}(\nu'_{j-1}, \nu'_j)$, by Lemma 5, it is $p^{\min}(\nu_{j-1}, \nu_j) \le \pi_j - \pi_{j-1} \le p^{\max}(\nu_{j-1}, \nu_j)$. Hence, $\mathcal{D} = [(\pi_1, \nu_1), \ldots, (\pi_n, \nu_n)]$, composed of the same job release times as $\mathcal{D}'$ but those corresponding vertices in $\tau_D^*(\mathcal{B})$, is a valid dDRT job sequence of $\tau_D^*(\mathcal{B})$. Combining that $\nu_j.\mathcal{C}(\pi_j) = \nu'_j.\mathcal{C}(\pi_j)$ $(\forall j = 1, \ldots, n)$, it holds $R(\tau_i, \mathcal{D}) = R(\tau_i, \mathcal{D}')$. As a result, $R(\tau_i, \tau_D^*(\mathcal{B}')) \le R(\tau_i, \tau_D^*(\mathcal{B}))$. ∎

This theorem demonstrates the dominance relationship between two valid speed partitions $\mathcal{B}$ and $\mathcal{B}'$ satisfying $\mathcal{B} \subseteq \mathcal{B}'$ in terms of the analysis accuracy, but in a qualitative way. In the following, we will quantify this relationship. We first define the vertex subset relationship.

Definition 13 (Subset Relation on Vertices' Speed Interval): Given two valid speed partitions $\mathcal{B}$ and $\mathcal{B}'$ with $\mathcal{B} \subseteq \mathcal{B}'$ and two vertices $v_i \in \tau_D^*(\mathcal{B})$ and $v'_i \in \tau_D^*(\mathcal{B}')$ as well as the corresponding speed intervals (β_{i-1}, β_i) and $(\beta'_{i-1}, \beta'_i]$, we say the interval of v'_i is a subset of that of vertex v_i (denoted as $v'_i \subseteq v_i$) if $(\beta'_{i-1}, \beta'_i] \subseteq (\beta_{i-1}, \beta_i]$.

We note that for two valid speed partitions $\mathcal{B} \subseteq \mathcal{B}'$, any vertex $v_i \in \tau_D^*(\mathcal{B})$ may correspond to multiple vertices $v'_i \in \tau_D^*(\mathcal{B}')$ such that $v'_i \subseteq v_i$. We introduce the following definition to obtain the maximum among them.



Time Interval (ms)	WCET (μs)							
	v'_1	v'_2	ϑ'_3	ν'_3	v'_3	ν'_4	v'_4	v'_5
$[\gamma_1, \gamma_2) = [0, 100)$	600	600	400	400	400	400	400	200
$[\gamma_2, \gamma_3) = [100, 200)$	600	400	300	300	300	300	300	200
$[\gamma_3, \gamma_4) = [200, +\infty)$	600	600	600	600	600	300	300	300

Fig. 3. Transformed dDRT task $\tau_D^*(\mathcal{B}')$ for the dAVR task in Table II with the speed partition $\mathcal{B}' = \{500, 1500, 2500, 2800, 3200, 3500, 4000, 4500, 6500\}$ (top). The WCET function of each vertex (bottom).

Definition 14 (Maximum Subset Vertex): Given two valid speed partitions $\mathcal{B}$ and $\mathcal{B}'$ with $\mathcal{B} \subseteq \mathcal{B}'$ and one vertex $v_i \in \tau_D^*(\mathcal{B})$, we say $v'_i \in \tau_D^*(\mathcal{B}')$ is the maximum subset vertex of v_i, if $v'_i \subseteq v_i$ and for any $v'_j \in \tau_D^*(\mathcal{B}')$ and $v'_j \subseteq v_i$, it holds that $v'_i \succeq v'_j$ (i.e., $v'_i \succ v'_j$ or $v'_i = v'_j$).

Example 4: For the dAVR task in Table II, consider the speed partition which is a superset of $\mathcal{B}$ in Example 3: $\mathcal{B}' = \{500, 1500, 2500, 2800, 3200, 3500, 4000, 4500, 6500\}$ (rpm). Fig. 3 gives the transformed dDRT task $\tau_D^*(\mathcal{B}')$, where vertices v'_1, v'_2, ϑ'_3, ν'_3, v'_3, ν'_4, v'_4, and v'_5, respectively, represent the intervals (500, 1500], (1500, 2500], (2500, 2800], (2800, 3200], (3200, 3500], (3500, 4000], (4000, 4500], and (4500, 6500].

For v_3 in $\tau_D^*(\mathcal{B})$, there are three vertices in $\tau_D^*(\mathcal{B}')$ such that $\nu'_3, v'_3, \vartheta'_3 \subseteq v_3$. Since $v'_3 \succ \nu'_3 \succ \vartheta'_3$, v'_3 is the maximum subset vertex of v_3.

Next, we introduce the concepts to measure the ratio of path lengths between two speed partitions $\mathcal{B} \subseteq \mathcal{B}'$.

Definition 15 (Edge Factor and Partition Factor): Consider two dDRT tasks $\tau_D^*(\mathcal{B}) = (\mathbb{V}, \mathbb{E})$ and $\tau_D^*(\mathcal{B}') = (\mathbb{V}', \mathbb{E}')$ with $\mathcal{B} \subseteq \mathcal{B}'$. For any edge $(v_i, v_j) \in \mathbb{E}$, assume v'_i and v'_j, respectively, are the maximum subset vertices of v_i and v_j, the edge factor $f_{\mathcal{B}/\mathcal{B}'}(v_i, v_j)$ is defined as the ratio between the lengths of their shortest paths, i.e.,

$$f_{\mathcal{B}/\mathcal{B}'}(v_i, v_j) = \frac{l^{\min}(v'_i, v'_j)}{l^{\min}(v_i, v_j)} = \frac{l^{\min}(v'_i, v'_j)}{p^{\min}(v_i, v_j)}.$$

The partition factor $f^{\max}_{\mathcal{B}/\mathcal{B}'}$ is defined as the maximum edge factor over all the edges in $\mathbb{E}$, i.e.,

$$f^{\max}_{\mathcal{B}/\mathcal{B}'} = \max_{(v_i, v_j) \in \mathbb{E}} f_{\mathcal{B}/\mathcal{B}'}(v_i, v_j).$$

Example 5: Consider v_2 and v_3 in Fig. 2. The corresponding maximum subset vertices are v'_2 and v'_3 in Fig. 3. By Theorem 1, $l^{\min}(v'_2, v'_3) = 23 + 21 + 19 = 63$. Since $p^{\min}(v_2, v_3) = 23$, it is $f_{\mathcal{B}/\mathcal{B}'}(v_2, v_3) = 63/23$. Likewise

$$\begin{aligned} f_{\mathcal{B}/\mathcal{B}'}(v_1, v_2) &= f_{\mathcal{B}/\mathcal{B}'}(v_2, v_1) = 1 \\ f_{\mathcal{B}/\mathcal{B}'}(v_2, v_3) &= f_{\mathcal{B}/\mathcal{B}'}(v_3, v_2) = 63/23 \\ f_{\mathcal{B}/\mathcal{B}'}(v_3, v_4) &= f_{\mathcal{B}/\mathcal{B}'}(v_4, v_3) = 15/8 \\ f_{\mathcal{B}/\mathcal{B}'}(v_4, v_5) &= f_{\mathcal{B}/\mathcal{B}'}(v_5, v_4) = 1. \end{aligned}$$

Also, for any v, $f_{\mathcal{B}/\mathcal{B}'}(v, v) = 1$. Hence, $f^{\max}_{\mathcal{B}/\mathcal{B}'} = 63/23$.

In the above example, we may observe that the relative edge factor is always no smaller than 1. This can be generalized to any such partitions, as demonstrated in the following lemma.

Lemma 6: Given two dDRT tasks $\tau^*_D(\mathcal{B}) = (\mathbb{V}, \mathbb{E})$ and $\tau^*_D(\mathcal{B}') = (\mathbb{V}', \mathbb{E}')$ with $\mathcal{B} \subseteq \mathcal{B}'$, for any edge $(v_i, v_j) \in \mathbb{E}$, it holds that $f_{\mathcal{B}/\mathcal{B}'}(v_i, v_j) \geq 1$. Thus, $f^{\max}_{\mathcal{B}/\mathcal{B}'} \geq 1$.

Proof: Assume v'_i and v'_j are the corresponding maximum subset vertices. Consider the shortest path $P' = (v'_i, \ldots, v'_j)$ from v'_i to v'_j. For each vertex v'_l in P', by Lemma 4 we can find a corresponding vertex v_l in $\mathbb{V}$. By Lemma 5, for any edge (v'_{l-1}, v'_l) in P', there exists a corresponding edge (v_{l-1}, v_l) in $\mathbb{E}$. Hence, we can construct a valid path $(v_i, \ldots, v_j)$ in $\tau^*_D(\mathcal{B})$ corresponding to P'. Furthermore, by Lemma 5, $l(v_i, \ldots, v_j) \leq l(v'_i, \ldots, v'_j) = l^{\min}(v'_i, v'_j)$. Since (v_i, v_j) is an edge in $\mathbb{E}$, by Lemma 2, $p^{\min}(v_i, v_j) = l^{\min}(v_i, v_j) \leq l(v_i, \ldots, v_j) \leq l^{\min}(v'_i, v'_j)$. Hence, $f_{\mathcal{B}/\mathcal{B}'}(v_i, v_j) \geq 1$. ∎

We now define a metric to quantify the maximum variation of WCET in the dAVR task (and the transformed dDRT task).

Definition 16 (WCET Factor): For a dAVR task, the WCET factor $f^{\max}_{\mathcal{C}}$ is defined as the maximum ratio between the maximum and minimum WCETs over any speed ω, i.e.,

$$f^{\max}_{\mathcal{C}} = \max_{\omega} \frac{\max_t \mathcal{C}(t, \omega)}{\min_t \mathcal{C}(t, \omega)}.$$

By definition, $f^{\max}_{\mathcal{C}} \geq 1$. The following lemma shows the WCET factor applies to the transformed dDRT tasks.

Lemma 7: Consider two dDRT tasks $\tau^*_D(\mathcal{B}) = (\mathbb{V}, \mathbb{E})$ and $\tau^*_D(\mathcal{B}') = (\mathbb{V}', \mathbb{E}')$ with $\mathcal{B} \subseteq \mathcal{B}'$. For any vertex $v \in \mathbb{V}$ and $v' \in \mathbb{V}'$ where $v' \subseteq v$, for any t, t', we have $v.\mathcal{C}(t) \leq v'.\mathcal{C}(t') \cdot f^{\max}_{\mathcal{C}}$ and $v'.\mathcal{C}(t') \leq v.\mathcal{C}(t) \cdot f^{\max}_{\mathcal{C}}$.

Proof: Since $\mathcal{B}$ is a valid partition, and $v' \subseteq v$, we have $v.\mathcal{C}(t) = v'.\mathcal{C}(t), \forall t$. This, combined with the definition of WCET factor, implies that the lemma is correct. ∎

Example 6: Consider the dAVR task in Table II as an example, $f^{\max}_{\mathcal{C}}$ is 2, achieved at any speed $\omega \in (2500, 3500]$.

We now extend the concept of speedup factor [33], to measure the approximation quality of a valid speed partition $\mathcal{B}$ compared to another one $\mathcal{B}'$ with $\mathcal{B} \subseteq \mathcal{B}'$ as well as the exact analysis.

Definition 17 (Speedup Factor): Considering any dAVR task τ^*_A and its two valid speed partitions $\mathcal{B}$ and $\mathcal{B}'$ with $\mathcal{B} \subseteq \mathcal{B}'$, $\mathcal{B}$ has a relative speedup factor of u compared to $\mathcal{B}'$, if the response time of τ_i estimated by using $\tau^*_D(\mathcal{B})$ on a speed-u processor, denoted as $R^u(\tau_i, \tau^*_D(\mathcal{B}))$, is bounded by $R(\tau_i, \tau^*_D(\mathcal{B}'))$, i.e., $R^u(\tau_i, \tau^*_D(\mathcal{B})) \leq R(\tau_i, \tau^*_D(\mathcal{B}'))$. We say $\mathcal{B}$ has a speedup factor of u if $R^u(\tau_i, \tau^*_D(\mathcal{B})) \leq R(\tau_i, \tau^*_A)$.

The following theorem upper bounds the relative speedup factor of a valid speed partition comparing to its superset.

Theorem 5: Given two valid speed partitions $\mathcal{B}$ and $\mathcal{B}'$ with $\mathcal{B} \subseteq \mathcal{B}'$, $\mathcal{B}$ has a relative speed factor compared to $\mathcal{B}'$ that is upper bounded by $f^{\max}_{\mathcal{B}/\mathcal{B}'} \cdot f^{\max}_{\mathcal{C}}$.

Proof: Consider a periodic task τ_i interfered by the higher-priority periodic task set $hp(i)$ and a dAVR task τ^*_A. Let $\tau^*_D(\mathcal{B}) = (\mathbb{V}, \mathbb{E})$ and $\tau^*_D(\mathcal{B}') = (\mathbb{V}', \mathbb{E}')$. Denote $u_1 = f^{\max}_{\mathcal{B}/\mathcal{B}'}$, $u_2 = f^{\max}_{\mathcal{C}}$, and $u = u_1 \cdot u_2$. Hence, $u \geq u_1$ and $u \geq u_2$.

For an arbitrary job sequence $\mathcal{D} = [(\pi_1, v_1) \ldots (\pi_n, v_n)]$ of $\tau^*_D(\mathcal{B})$, we denote $R^u(\tau_i, \mathcal{D})$ as the response time of τ_i interfered by $\mathcal{D}$ on a speed-u processor. Without loss of generality, let $R^u(\tau_i, \mathcal{D}) > \pi_n - \pi_1$ (otherwise we can always find such a subsequence of $\mathcal{D}$). We now prove there exists another dDRT job sequence $\mathcal{D}' \in \tau^*_D(\mathcal{B}')$ such that $R^u(\tau_i, \mathcal{D}) \leq R(\tau_i, \mathcal{D}')$, which is sufficient to yield $R^u(\tau_i, \tau^*_D(\mathcal{B})) \leq R(\tau_i, \tau^*_D(\mathcal{B}'))$.

Let v'_l be the maximum subset vertex of v_l ($l = 1, \ldots, n$). We construct $\mathcal{D}'$ by iteratively replacing each job (π_l, v_l) with:

if $l = 1$: job $(\pi'_1 = \pi_1, v'_1)$;

if $l > 1$: a sequence of jobs following the shortest path $(v'_{l-1}, v'_1, \ldots, v'_m, v'_l)$ from v'_{l-1} to v'_l, i.e.,

$$\begin{aligned} &\left(\chi'_1 = \pi'_{l-1} + p^{\min}\left(v'_{l-1}, v'_1\right), v'_1\right) \\ &\left(\chi'_2 = \chi'_1 + p^{\min}\left(v'_1, v'_2\right), v'_2\right), \ldots \\ &\left(\chi'_m = \chi'_{m-1} + p^{\min}\left(v'_{m-1}, v'_m\right), v'_m\right) \\ &\left(\pi'_l = \chi'_m + p^{\min}\left(v'_m, v'_l\right), v'_l\right). \end{aligned}$$

This gives $\pi'_l = \pi'_{l-1} + l^{\min}(v'_{l-1}, v'_l)$. Hence $\pi'_1 = \pi_1$ and $\forall 2 \leq l \leq n, \pi'_l = \pi_1 + \sum_{k=2}^{l} l^{\min}(v'_{k-1}, v'_k)$. Obviously, the resultant job sequence $\mathcal{D}'$ is valid in $\tau^*_D(\mathcal{B}')$.

We now prove by contradiction that $R(\tau_i, \mathcal{D}') > \pi'_n - \pi'_1$. Assume $R(\tau_i, \mathcal{D}') \leq \pi'_l - \pi'_1$ where $2 \leq l \leq n$. This implies there exists $t_0 \leq \pi'_l - \pi'_1$ such that

$$\begin{aligned} &C_i + \sum_{\tau_j \in hp(i)} \left\lceil \frac{t_0}{T_j} \right\rceil C_j + \sum_{k=1}^{l-1} v'_k.\mathcal{C}\left(\pi'_k\right) \\ &\leq C_i + \sum_{\tau_j \in hp(i)} \left\lceil \frac{t_0}{T_j} \right\rceil C_j + \mathcal{D}'.I(t_0) = t_0. \end{aligned} \tag{17}$$

Further, by the definition of u_1 (partition factor), it is

$$\begin{aligned} t_0/u_1 &\leq \left(\pi'_l - \pi'_1\right)/u_1 = \left(\sum_{k=2}^{l} l^{\min}\left(v'_{l-1}, v'_l\right)\right)/u_1 \\ &\leq \sum_{k=2}^{l} p^{\min}(v_{l-1}, v_l) \leq \pi_l - \pi_1. \end{aligned}$$

Hence, $\mathcal{D}.I(t_0/u_1) \leq \sum_{k=1}^{l-1} v_k.\mathcal{C}(\pi_k)$. This enables to derive

$$\begin{aligned} &\frac{1}{u} \cdot \left(C_i + \sum_{\tau_j \in hp(i)} \left\lceil \frac{t_0/u_1}{T_j} \right\rceil C_j + \mathcal{D}.I(t_0/u_1)\right) \\ &\leq \frac{1}{u} \cdot \left(C_i + \sum_{\tau_j \in hp(i)} \left\lceil \frac{t_0/u_1}{T_j} \right\rceil C_j + \sum_{k=1}^{l-1} v_k.\mathcal{C}(\pi_k)\right) \\ &\overset{(a)}{\leq} \frac{1}{u_1} \cdot \left(C_i + \sum_{\tau_j \in hp(i)} \left\lceil \frac{t_0}{T_j} \right\rceil C_j + \frac{1}{u_2} \sum_{k=1}^{l-1} v_k.\mathcal{C}(\pi_k)\right) \\ &\overset{(b)}{\leq} \frac{1}{u_1} \cdot \left(C_i + \sum_{\tau_j \in hp(i)} \left\lceil \frac{t_0}{T_j} \right\rceil C_j + \sum_{k=1}^{l-1} v'_k.\mathcal{C}\left(\pi'_k\right)\right) \overset{(c)}{\leq} t_0/u_1. \end{aligned}$$

Here (a) is due to $u_2 \geq 1$, (b) is because of Lemma 7, and (c) comes from (17). The above equation implies $R^u(\tau_i, \mathcal{D}) \leq t_0/u_1 \leq \pi_n - \pi_1$, which contradicts $R^u(\tau_i, \mathcal{D}) > \pi_n - \pi_1$. As a result, it must be $R(\tau_i, \mathcal{D}') > \pi'_n - \pi'_1$.

Now we compare the total workloads for calculating $R^u(\tau_i, \mathcal{D})$ and $R(\tau_i, \mathcal{D}')$. For any $t > \pi'_n - \pi'_1$

$$\begin{aligned} & C_i + \sum_{\tau_j \in hp(i)} \left\lceil \frac{t}{T_j} \right\rceil C_j + \mathcal{D}'.I(t) \\ &= C_i + \sum_{\tau_j \in hp(i)} \left\lceil \frac{t}{T_j} \right\rceil C_j + \sum_{k=1}^{n} v'_k.\mathcal{C}(\pi'_k) \\ &\geq C_i + \sum_{\tau_j \in hp(i)} \left\lceil \frac{t}{T_j} \right\rceil C_j + \frac{1}{u_2} \cdot \sum_{k=1}^{n} v_k.\mathcal{C}(\pi_k) \\ &\geq \frac{1}{u} \cdot \left(C_i + \sum_{\tau_j \in hp(i)} \left\lceil \frac{t}{T_j} \right\rceil C_j + \sum_{k=1}^{n} v_k.\mathcal{C}(\pi_k) \right) \\ &= \frac{1}{u} \cdot \left(C_i + \sum_{\tau_j \in hp(i)} \left\lceil \frac{t}{T_j} \right\rceil C_j + \mathcal{D}.I(t) \right) \end{aligned}$$

this inequality yields that $R^u(\tau_i, \mathcal{D}) \leq R(\tau_i, \mathcal{D}')$. ∎

Given the above theorem, we can conclude that the pessimism introduced in any valid speed partition is bounded.

Theorem 6: For any valid speed partition $\mathcal{B}$, its speedup factor must be finite.

Proof: We consider $\mathcal{B}'$ which partitions the speed space evenly with size ϵ, where ϵ is sufficiently small such that $\mathcal{B} \subseteq \mathcal{B}'$. Obviously when $\epsilon \to 0$, $\mathcal{B}'$ provides an exact analysis.

For any edge (v_i, v_j) in $\tau^*_D(\mathcal{B})$, if the vertices v_i and v_j in $\mathbb{V}$ correspond, respectively, to the speed intervals $(\beta_{i-1}, \beta_i]$ and $(\beta_{j-1}, \beta_j]$, then the two maximum subset vertices v'_i and v'_j in $\tau^*_D(\mathcal{B}')$ correspond, respectively, to $(\beta_i - \epsilon, \beta_i]$ and $(\beta_j - \epsilon, \beta_j]$. When $\epsilon \to 0$, it holds that

$$f_{\mathcal{B}/\mathcal{B}'}(v_i, v_j) = \frac{l^{\min}(v'_i, v'_j)}{p^{\min}(v_i, v_j)} \to \frac{l^{\min}(\beta_i, \beta_j)}{p^{\min}(v_i, v_j)}$$

where $l^{\min}(\beta_i, \beta_j)$ represents the minimum time from β_i to β_j after rotating an integer number of angular periods. Thus, by Theorem 5, the speedup factor of $\mathcal{B}$ must be bounded by

$$\max_{(v_i, v_j) \in \mathbb{E}} f_{\mathcal{B}/\mathcal{B}'}(v_i, v_j) \cdot f^{\max}_{\mathcal{C}} = \max_{(v_i, v_j) \in \mathbb{E}} \frac{l^{\min}(\beta_i, \beta_j)}{p^{\min}(v_i, v_j)} \cdot f^{\max}_{\mathcal{C}}$$

which is finite as $l^{\min}(\beta_i, \beta_j)$ for any β_i and β_j is finite. ∎

D. Finding Critical dDRT Job Sequences

We now discuss how to efficiently analyze the system schedulability based on the transformed dDRT task system. We note that (15) still requires to enumerate all the (infinitely many) job sequences of the dDRT task, which is obviously impractical. In the following, we develop techniques to find a finite set of representative dDRT job sequences *without losing any accuracy*. The idea is that, if two dDRT job sequences share the same sequence of job WCETs, then the one always with a shorter inter-release time will dominate the other in terms of their interference functions. Hence, we may partition the space of dDRT job release times such that the WCET of each vertex is constant in each release time interval. We note that it is sufficient to have each release time interval within two consecutive reconfigurations. This idea is captured with the following definitions and algorithms.

Definition 18 (Common-WCET dDRT Job Sequence Set): For a dDRT task $\tau^*_D = (\mathbb{V}, \mathbb{E})$, a common-WCET dDRT job sequence set, denoted as $\mathbb{C} = [(c^-_1, c^+_1, v_1), \ldots, (c^-_n, c^+_n, v_n)]$, is a sequence of release time ranges $[c^-_l, c^+_l]$ and vertices v_l that satisfies:

1) $\forall 1 \leq l \leq n$, $c^-_l \leq c^+_l$;
2) the WCET of $v_l \in \mathbb{V}$ is the same within $[c^-_l, c^+_l]$, i.e., $\forall t_1, t_2 \in [c^-_l, c^+_l]$, $v_l.\mathcal{C}(t_1) = v_l.\mathcal{C}(t_2)$;
3) $\forall 1 \leq l < n$, $(v_l, v_{l+1}) \in \mathbb{E}$, $c^-_{l+1} \geq c^-_l + p^{\min}(v_l, v_{l+1})$ and $c^+_{l+1} \leq c^+_l + p^{\max}(v_l, v_{l+1})$.

In Definition 18, the second condition is satisfied if $[c^-_l, c^+_l]$ is in $[\gamma_k, \gamma_{k+1})$, i.e., it is contained in the interval between two consecutive reconfigurations. The third condition means that (v_l, v_{l+1}) is an edge of τ^*_D, and $[c^-_{l+1}, c^+_{l+1}]$ shall be reachable from $[c^-_l, c^+_l]$.

Definition 19 (Job Sequence of $\mathbb{C}$): $\mathcal{D} = [(\pi_1, v_1), \ldots, (\pi_n, v_n)]$ is called a job sequence of $\mathbb{C} = [(c^-_1, c^+_1, v_1), \ldots, (c^-_n, c^+_n, v_n)]$, denoted as $\mathcal{D} \in \mathbb{C}$, if:

4) $\mathcal{D}$ has the same sequence of vertices as $\mathbb{C}$;
5) $\forall 1 \leq l \leq n, \quad \pi_l \in [c^-_l, c^+_l]$, i.e., each job is released in the corresponding range in $\mathbb{C}$.

Definition 20 (Critical Job Sequence of $\mathbb{C}$): $\mathcal{D}^c = [(\pi^c_1, v_1), \ldots, (\pi^c_n, v_n)] \in \mathbb{C}$ is called a critical job sequence of $\mathbb{C}$ if:

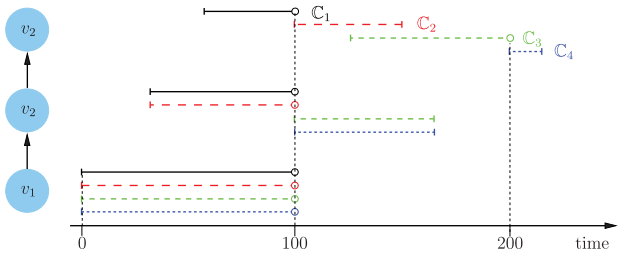
6) $\mathcal{D}^c \in \tau^*_D$, i.e., it is a job sequence of τ^*_D according to Definition 8.
7) $\forall \mathcal{D} \in \mathbb{C} \cap \tau^*_D, \forall 1 \leq l \leq n, \ \pi^c_l - \pi^c_1 \leq \pi_l - \pi_1.$ (18)

With these definitions, we first claim that the length n of the job sequence set $\mathbb{C}$ is always well bounded for checking the schedulability of a periodic task τ_i with deadline D_i. Specifically, it is sufficient to consider all job sequence sets $\mathbb{C} = [(c^-_1, c^+_1, v_1), \ldots, (c^-_n, c^+_n, v_n)]$ satisfying $c^-_n - c^+_1 \leq D_i$. For those $\mathbb{C}$ with $c^-_n - c^+_1 > D_i$, there is no job sequence $\mathcal{D} \in \mathbb{C}$ such that the inter-release time between the last and first jobs is no larger than D_i, and τ_i will not suffer interferences from all jobs in $\mathcal{D}$ if it is schedulable.

We now explain how to enumerate all necessary job sequence sets. By Definition 18, this involves: 1) generating all valid paths (i.e., valid sequences of types of jobs) in the dDRT with limited length n and 2) finding all suitable release time ranges. The former follows that of a generic digraph such as the DRT tasks [27]. Given a path $(v_1, \ldots, v_n)$, the algorithm to generate all the related common-WCET job sequence sets can be found in [32, Algorithm 1]. Below we give an example on constructing a collection of common-WCET dDRT job sequence sets for a given path $(v_1, \ldots, v_n)$.

Example 7: Given the dDRT task τ^*_D in Fig. 2, we consider one path (v_1, v_2, v_2) with an initial tuple $(0, 100, v_1)$. For the next vertex v_2 where the edge (v_1, v_2) is labeled as $[32, 65]$, the release time of v_2 is in the range $[32, 165]$. However, it should be partitioned into two intervals $[32, 100)$ and $[100, 165)$ since $\gamma_2 = 100$ is a reconfiguration time.[2] As edge (v_2, v_2) is labeled with $[25, 50]$, the last vertex has two possible release time ranges $[57, 150)$ and $[125, 215]$ since there are two intervals for its preceding vertex. Likewise, these release time intervals will be further partitioned by the reconfiguration times ($\gamma_2 = 100$ and $\gamma_3 = 200$): $[57, 150)$ is split into $[57, 100)$ and

[2]In Examples 7 and 8, for simplicity we *treat closed and half-open intervals the same*.



(a)

$\mathbb{C}$	(c_1^-, c_1^+, ν_1)	(c_2^-, c_2^+, ν_2)	(c_3^-, c_3^+, ν_3)
1	$(0, 100, v_1)$	$(32, 100, v_2)$	$(57, 100, v_2)$
2			$(100, 150, v_2)$
3		$(100, 165, v_2)$	$(125, 200, v_2)$
4			$(200, 215, v_2)$

(b)

Fig. 4. Illustrative example of common-WCET dDRT job sequence sets in the dDRT task τ_D^* of Fig. 2, for a given path (v_1, v_2, v_2) and an initial range $c_1^- = 0, c_1^+ = 100$.

[100, 150), and [125, 215] is divided into [125, 200) and [200, 215]. Thus, we obtain four common-WCET dDRT job sequence sets, as shown in Fig. 4(b).

In Definition 20, (18) implies that the interference function of $\mathcal{D}^c$ is always no smaller than that of $\mathcal{D}$, i.e., $\forall t \geq 0, \mathcal{D}^c.I(t) \geq \mathcal{D}.I(t)$. This is because $\mathcal{D}^c$ and $\mathcal{D}$ share the same job WCETs, but jobs in $\mathcal{D}^c$ are always released tighter than $\mathcal{D}$. Hence, for the purpose of schedulability analysis we can use a critical job sequence $\mathcal{D}^c$ of $\mathbb{C}$ to represent $\mathbb{C}$, and ignore all other sequences in $\mathbb{C}$. Finding such a $\mathcal{D}^c$ for a given $\mathbb{C}$ is detailed in Algorithm 2.

Specifically, Algorithm 2 first constructs a new job sequence set $\widetilde{\mathbb{C}}$ (lines 2–6). We term this process as "legalization," since any $\mathcal{D} \in \mathbb{C}$ but $\mathcal{D} \notin \widetilde{\mathbb{C}}$ must be $\mathcal{D} \notin \tau_D^*$ (i.e., $\mathcal{D}$ is illegal) [32, Lemma 7]. Such illegal regions in $\mathbb{C}$ is from the necessary splitting of release time ranges by the configuration times.

The legalization starts with setting all vertices of $\widetilde{\mathbb{C}}$ to be those of $\mathbb{C}$ (line 2). It initializes the release time range of the last vertex as that in $\mathbb{C}$ (line 3). The iteration in lines 4–6 performs a backward pass on the other vertices, to shorten their release time ranges such that jobs released outside of these ranges are always illegal. Given $(\tilde{c}_l^-, \tilde{c}_l^+, \nu_l)$, lines 5 and 6 will obtain the valid release time range of its previous vertex v_{l-1} by considering the minimum/maximum inter-release times and satisfying the time constraint on v_{l-1} in $\mathbb{C}$.

Then Algorithm 2 constructs a critical job sequence $\mathcal{D}^c$ of $\widetilde{\mathbb{C}}$ (and consequently of $\mathbb{C}$) in lines 7–10. It sets the vertices of $\mathcal{D}^c$ as those of $\widetilde{\mathbb{C}}$ (line 7), and the release time of the first job as late as possible (line 8). Subsequently, it does a forward pass (lines 9 and 10) to release other jobs as early as possible, subject to the requirement that $\mathcal{D}^c \in \widetilde{\mathbb{C}}$.

The following example explains how Algorithm 2 works.

Example 8: Take the third common-WCET job sequence set in Fig. 4(b) as an example, i.e., $\mathbb{C} = [(0, 100, v_1), (100, 165, v_2), (125, 200, v_2)]$. Algorithm 2 first initializes the release time range of the last vertex in the legalized common-WCET job sequence set $\widetilde{\mathbb{C}}$ as $\tilde{c}_3^- = 125$, $\tilde{c}_3^+ = 200$. It then performs a backward pass on the other vertices to determine their legalized release time ranges. This will find

$$\tilde{c}_2^- = \max\left(\tilde{c}_3^- - p^{\max}(v_2, v_2), c_2^-\right) = \max(125 - 50, 100) = 100$$
$$\tilde{c}_2^+ = \min\left(\tilde{c}_3^+ - p^{\min}(v_2, v_2), c_2^+\right) = \min(200 - 25, 165) = 165$$
$$\tilde{c}_1^- = \max\left(\tilde{c}_2^- - p^{\max}(v_1, v_2), c_1^-\right) = \max(100 - 65, 0) = 35$$
$$\tilde{c}_1^+ = \min\left(\tilde{c}_2^+ - p^{\min}(v_1, v_2), c_1^+\right) = \min(165 - 32, 100) = 100.$$

[Note that $p^{\min}(v_1, v_2) = 32$, $p^{\max}(v_1, v_2) = 65$, $p^{\min}(v_2, v_2) = 25$, $p^{\max}(v_2, v_2) = 50$.] This legalization process will shorten the first release time range from [0, 100] to [35, 100]: if the first job is released in [0, 35), then the job sequence cannot be legal as the inter-release time between the first and second jobs are always larger than $p^{\max}(v_1, v_2) = 65$.

Algorithm 2 then uses $\pi_1^c = \tilde{c}_1^+ = 100$ for the forward pass, to get $\pi_2^c = \max(\tilde{c}_2^-, \pi_1^c + p^{\min}(v_1, v_2)) = \max(100, 100 + 32) = 132$ and $\pi_3^c = \max(\tilde{c}_3^-, \pi_2^c + p^{\min}(v_2, v_2)) = \max(125, 132 + 25) = 157$. Table III illustrates the generated critical job sequences for the four common-WCET job sequence sets in Fig. 4(b).

The correctness of Algorithm 2 is guaranteed by the following theorem (the formal proof is given in [32]).

Theorem 7: Given any $\mathbb{C}$, Algorithm 2 generates a critical job sequence $\mathcal{D}^c$ according to Definition 20.

Algorithm 2 Constructing a Critical dDRT Job Sequence $\mathcal{D}^c$ for a Given Common-WCET dDRT Job Sequence Set $\mathbb{C} = [(c_1^-, c_1^+, \nu_1), \ldots, (c_n^-, c_n^+, \nu_n)]$

```
1:  procedure ConstructCriticalJobSequence(ℂ)
2:      ℂ̃ ← [(c̃_1^-, c̃_1^+, ν_1), ..., (c̃_n^-, c̃_n^+, ν_n)];
3:      c̃_n^- ← c_n^-, c̃_n^+ ← c_n^+;
4:      for l = n to 2 do // Backward Pass
5:          c̃_{l-1}^- ← max(c̃_l^- − p^max(ν_{l-1}, ν_l), c_{l-1}^-);
6:          c̃_{l-1}^+ ← min(c̃_l^+ − p^min(ν_{l-1}, ν_l), c_{l-1}^+);
7:      D^c ← [(π_1^c, ν_1), ..., (π_n^c, ν_n)];
8:      π_1^c ← c̃_1^+;
9:      for l = 1 to n − 1 do // Forward Pass
10:         π_{l+1}^c ← max(π_l^c + p^min(ν_l, ν_{l+1}), c̃_{l+1}^-);
11:     return D^c;
```

TABLE III
ILLUSTRATIVE EXAMPLE ON ALGORITHM 2: CONSTRUCTION OF CRITICAL JOB SEQUENCES FOR THE FOUR SETS IN FIG. 4(b)

$\mathbb{C}$	Line 3		Lines 4–6				Line 8	Lines 9–10	
	$\tilde{c}_3^-$	$\tilde{c}_3^+$	$\tilde{c}_2^-$	$\tilde{c}_2^+$	$\tilde{c}_1^-$	$\tilde{c}_1^+$	π_1^c	π_2^c	π_3^c
1	57	100	32	75	0	43	43	75	100
2	100	150	50	100	0	68	68	100	125
3	125	200	100	165	35	100	100	132	157
4	200	215	150	165	85	100	100	150	200

IV. SCHEDULABILITY ANALYSIS OF DAVR TASKS

Now consider a dAVR task τ_i^* interfered by a set of periodic tasks $hp(i)$ and a set of dAVR tasks $hp^*(i)$. For a job (σ, ω) of τ_i^*, since the dAVR tasks share the same angular period and phase, all tasks in $hp^*(i)$ also release a job at time σ. Furthermore, since τ_i^* has a constrained deadline, it has to finish before its next release [and any new job from $hp^*(i)$]. Hence, τ_i^* will only be interfered by one job from each task

$\tau_j^* \in hp^*(i)$, and the response time of (σ, ω) is computed as

$$R(\tau_i^*, \sigma, \omega) = \min_{t>0} \left\{ t \mid \mathcal{C}_i(\sigma, \omega) + \sum_{\tau_j^* \in hp^*(i)} \mathcal{C}_j(\sigma, \omega) + \sum_{\tau_j \in hp(i)} \left\lceil \frac{t}{T_j} \right\rceil C_j \le t \right\}. \quad (19)$$

We now reduce the set of jobs of τ_i^* to be checked for its schedulability. First, by (7) the WCET of any dAVR job at any speed ω only changes at the set of reconfiguration times $\mathcal{T} = \{\gamma_1, \ldots, \gamma_T\}$. Hence, it is sufficient to only consider $\mathcal{T}$ as the set of representative release times for dAVR jobs

$$R(\tau_i^*, \omega) = \max_{\sigma \in \mathcal{T}} R(\tau_i^*, \sigma, \omega). \quad (20)$$

Second, we denote the *ordered* set of switching speeds from τ_i^* itself and all higher priority dAVR tasks as

$$\mathbb{W}_i = \left\{ \omega_{j,k}^m | j \in hp^*(i) \cup \{i\},\ k = 1 \ldots T, m = 1 \ldots M_{j,k} \right\}$$

and consider two consecutive switching speeds ω_l and ω_{l+1} in $\mathbb{W}_i$. By (7) the WCET of a dAVR job remains the same for any $\omega \in (\omega_l, \omega_{l+1}]$. Also, by Property 3, the deadline $D_i(\omega)$ is monotonically decreasing with ω. Hence, we can use ω_{l+1} to represent all angular speeds in $(\omega_l, \omega_{l+1}]$, and it is sufficient to only check the schedulability of jobs of τ_i^* released with an angular speed ω in $\mathbb{W}_i$. That is, τ_i^* is schedulable if the following condition is satisfied:

$$\forall \omega \in \mathbb{W}_i,\ R(\tau_i^*, \omega) \le D_i(\omega). \quad (21)$$

In the end, the schedulability analysis of τ_i^* in (19)–(21) only requires to check a finite number of jobs of τ_i^*.

V. Experimental Evaluation

In this section, we evaluate the benefits of the proposed approach on system schedulability, using randomly generated synthetic task systems. We adopt the parameters on a practical engine in [1] and [17]: 1) the minimum/maximum rotational speed is 500/6500 rpm and 2) the maximum acceleration/deceleration is 1.62×10^{-4} rev/ms^2. Hence, the engine needs 35 revolutions to accelerate/decelerate between the minimum and maximum speeds.

We compare three analysis methods as follows.

1) *dAVR:* The analysis proposed in this article, based on the transformation of dAVR tasks to dDRT tasks.
2) *dAVR2sAVR:* Since there is no existing safe analysis for dAVR task systems, we consider a simple, sufficient-only analysis as the baseline. Specifically, we approximate a dAVR task τ_i^* with an sAVR task $\tilde{\tau}_i^*$, where the WCET of $\tilde{\tau}_i^*$ released at speed ω is set as the maximum WCET of τ_i^* released at speed ω over all configurations. We then apply the analysis for sAVR task systems proposed in [15] and [16].
3) *UB:* The necessary-only analysis from [1] and [17]. It underestimates the workload from a dAVR task τ_i^* in the kth configuration as the maximum over a series of virtual execution mode sets $\{(C_{i,k}^m, \omega_{i,k}^m), (C_{i,k}^{M_{i,k}}, \omega^{\max})\}, \forall m = 1, \ldots, M_{i,k} - 1$, i.e., the mth mode with WCET $C_{i,k}^m$ executed in the speed interval $(\omega^{\min}, \omega_{i,k}^m]$, and the simplest control strategy with WCET $C_{i,k}^{M_{i,k}}$ effective in $(\omega_{i,k}^m, \omega^{\max}]$.

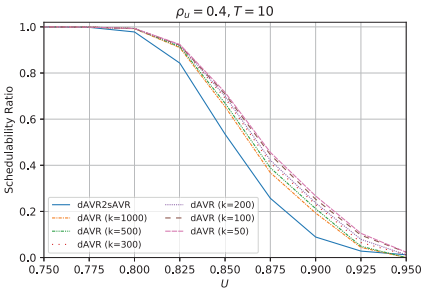


Fig. 5. Schedulability ratio versus system utilization U.

For the method dAVR2sAVR, we use the speed partition in [14] and [26] that is exact for schedulability analysis of sAVR tasks. For dAVR, we follow [26] and use $k \in \{50, 100, 200, 300, 500, 1000\}$ to obtain speed partitions with different granularities. Given k, the speed partition is the union of the smallest valid partition in (12) and the set $\{\omega^{\min} + i \cdot k \mid i = 1, \ldots, \lceil [\omega^{\max} - \omega^{\min}]/k \rceil\}$. Hence, the smaller k is, the finer the speed partition.

A. Random Task Systems

The random task systems are generated following [15]. Each system consists of 20 periodic tasks and one dAVR task τ_A^*. Let U_P and U_A denote the total utilization of the periodic tasks and the maximum utilization of τ_A^*, respectively, where $U_A = \max_{t,\omega}(\mathcal{C}_A(t, \omega)/[T^{\min}(\omega, \omega)])$. The total system utilization is $U = U_P + U_A$ and the fraction for τ_A^* is ρ_u (i.e., $U_A = \rho_u \cdot U$). The utilization of each periodic task is computed by the UUnifast algorithm [34] and the period is uniformly distributed between 3 and 100 ms. The periodic tasks have implicit deadlines. The angular period and deadline of the dAVR task are both equal to one revolution. The task priorities are assigned with the deadline monotonic policy, where the deadline of the dAVR task is its minimum value $D_A(\omega^{\max})$.

The dAVR task includes a set of reconfiguration times $\{\gamma_1, \ldots, \gamma_T\}$ where $\gamma_l = (l-1) \cdot T_s$ for $1 \le l \le T$. We set $T_s = 500$ ms, which is the sample period in most standard driving cycles [35]. Moreover, for each configuration, we generate M execution modes as follows. We first randomly select one mode for the maximum utilization U_A, and the utilizations of the other modes are set within the range $[0.85 \times U_A, U_A]$. The switching speeds are randomly chosen from a uniformly distributed set between [1000, 6000] rpm. As a result, the WCET of each mode m effective in the speed interval $(\omega^m, \omega^{m+1}]$ is set as the product of its utilization and minimum inter-release time $T^{\min}(\omega^{m+1}, \omega^{m+1})$.

B. Schedulability Results

We vary one of the four parameters U, ρ_u, M, and T. Since our motivation is to provide better performance while guaranteeing schedulability, we filter out systems that is deemed unschedulable by UB. Thus, the schedulability ratio shown in Figs. 5–8 is a normalization with respect to UB (hence UB is omitted). Each data point in the figures is the average over 1000 random task sets.

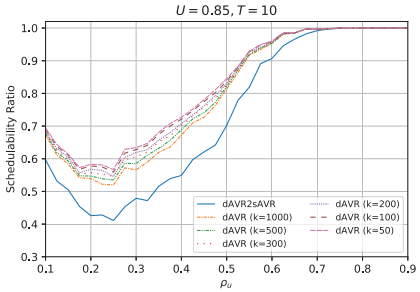


Fig. 6. Schedulability ratio versus dAVR utilization fraction ρ_u.

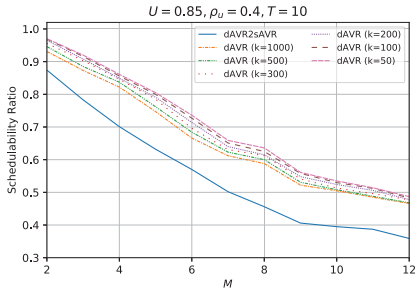


Fig. 7. Schedulability ratio versus number of modes M.

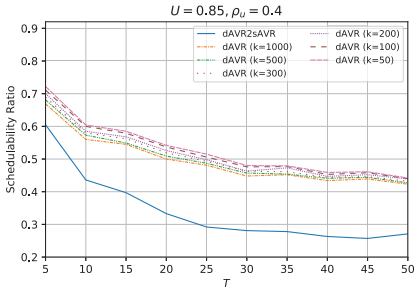


Fig. 8. Schedulability ratio versus number of configurations T.

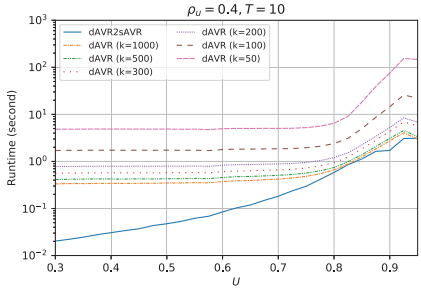


Fig. 9. Runtime versus system utilization U.

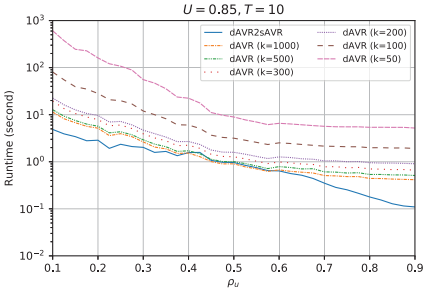


Fig. 10. Runtime versus dAVR utilization fraction ρ_u.

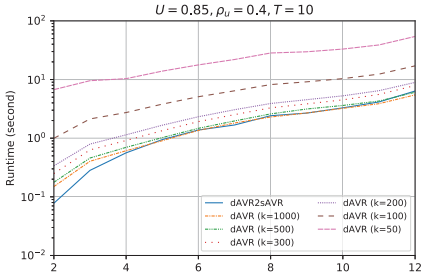


Fig. 11. Runtime versus number of modes M.

In the first experiment, the generated dAVR tasks have ten configurations (i.e., $T = 10$) where the number of execution modes in each configuration is randomly set between 4 and 8. The total utilization U is varied from 0.3 to 0.95 with $\rho_u = 0.4$. The schedulability ratios of all methods are always 1 when $U < 0.75$, hence, Fig. 5 only reports the results for $U \geq 0.75$. As shown in the figure, no matter the value of k is, dAVR always has a higher schedulability ratio (hence better analysis accuracy) than dAVR2sAVR. For example, at $U = 0.875$, the schedulability ratio of dAVR with $k = 50$ is around 20% higher than that of dAVR2sAVR. As studied in Section III-C, dAVR with a smaller k has a higher schedulability ratio and hence is more precise. Finally, dAVR has substantially lower schedulability ratios compared to the necessary-only analysis UB, which indicates that dAVR may be significantly pessimistic. However, we show in [8] that their experimental control performances are always indistinguishable.

The comparison between dAVR and dAVR2sAVR is the same in the next three experiments. In the second experiment, the generation of the dAVR tasks is similar to the first, but the dAVR utilization fraction ρ_u is varied within $[0.1, 0.9]$ and the total utilization is fixed at $U = 0.85$. In the third experiment, we vary the number of modes M of all the ten configurations while the total utilization is $U = 0.85$ and the fraction of dAVR task utilization is $\rho_u = 0.4$. In the fourth experiment, we consider a varying number of configurations T, where the total utilization is fixed at 0.85, each configuration has 4–8 execution modes, and $\rho_u = 0.4$. The results are illustrated in Figs. 6–8, respectively. As in these figures, dAVR is always more precise than dAVR2sAVR regardless of the value of k. The difference between dAVR with $k = 50$ and dAVR2sAVR is typically around 6%–22%. The only exception is when $\rho_u \geq 0.6$ in Fig. 6, where these seven methods all have a schedulability ratio close to 1 (i.e., close to that of UB). This is because when the dAVR task has a high workload, most task systems are either easily unschedulable (such that UB also deems the system unschedulable), or easily schedulable (most periodic tasks have a higher priority than the dAVR task and are not affected by its high workload).

C. Runtime Results

We also study the runtime of these methods, which are implemented in the C++ language and executed on a machine with an Intel Core i7 3.4-GHz CPU. The runtime results are illustrated in Figs. 9–12. As observed in these figures, a finer speed partition requires a longer analysis runtime but provides a more precise analysis. Specifically, the maximum difference in the runtime between dAVR with $k = 1000$ and

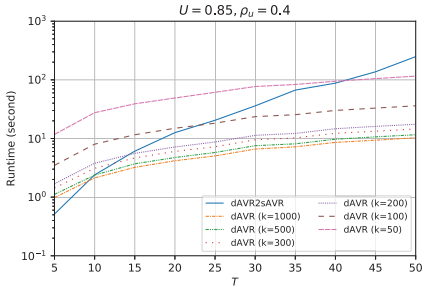


Fig. 12. Runtime versus number of configurations T.

dAVR with $k = 50$ is close to two orders of magnitude. Combining the schedulability ratio results, another observation is that the analysis runtime of one finer speed partition significantly increases, while its analysis accuracy has relatively small improvement. Compared with dAVR, dAVR2sAVR typically has a smaller runtime due to its lower complexity. The obvious exceptions are in Fig. 12, e.g., when $T > 40$, the runtime of dAVR2sAVR is larger than all the variations of dAVR. This is because with more configurations T, the consolidated sAVR task has more switching speeds and needs much longer analysis time.

VI. Conclusion

In this article, we propose the dAVR task model to reconfigure engine switching speeds at runtime. We provide a sufficient-only response time analysis, which partitions the speed space and approximates the workload of dAVR task with a new type of digraph tasks. We prove that the approximation method is safe and has a bounded speedup factor (hence bounded pessimism). Experiments show that our analysis is substantially more accurate than simple extensions of those for sAVR systems.

Acknowledgment

The authors would like to thank A. Biondi for sharing the source code in [1].

References

[1] A. Biondi, M. D. Natale, and G. Buttazzo, "Performance-driven design of engine control tasks," in *Proc. IEEE/ACM Conf. Cyber Phys. Syst.*, Vienna, Austria, 2016, pp. 1–10.

[2] T. Feld, A. Biondi, R. I. Davis, G. Buttazzo, and F. Slomka, "A survey of schedulability analysis techniques for rate-dependent tasks," *J. Syst. Softw.*, vol. 138, pp. 100–107, Apr. 2018.

[3] D. Buttle, "Keynote speech: Real-time in the prime-time," in *Proc. Euromicro Conf. Real Time Syst.*, 2012, pp. 12–13.

[4] J. Guanetti, Y. Kim, and F. Borrelli, "Control of connected and automated vehicles: State of the art and future challenges," *Annu. Rev. Control*, vol. 45, pp. 18–40, May 2018.

[5] J. Lin, W. Yu, X. Yang, Q. Yang, X. Fu, and W. Zhao, "A real-time en-route route guidance decision scheme for transportation-based cyberphysical systems," *IEEE Trans. Veh. Technol.*, vol. 66, no. 3, pp. 2551–2566, Mar. 2017.

[6] W. Schwarting, J. Alonso-Mora, and D. Rus, "Planning and decision-making for autonomous vehicles," *Annu. Rev. Control Robot. Auton. Syst.*, vol. 1, pp. 187–210, May 2018.

[7] S. E. Li *et al.*, "Dynamical modeling and distributed control of connected and automated vehicles: Challenges and opportunities," *IEEE Intell. Transp. Syst. Mag.*, vol. 9, no. 3, pp. 46–58, Jul. 2017.

[8] C. Peng, Y. Zhao, and H. Zeng, "Dynamic switching speed reconfiguration for engine performance optimization," in *Proc. 56th ACM/IEEE Design Autom. Conf.*, Las Vegas, NV, USA, 2019, Art. no. 11.

[9] J. Kim, K. Lakshmanan, and R. Rajkumar, "Rhythmic tasks: A new task model with continually varying periods for cyber-physical systems," in *Proc. IEEE/ACM Conf. Cyber Phys. Syst.*, 2012, pp. 55–64.

[10] 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. Conf. Design Autom. Test Europe*, Grenoble, France, 2013, pp. 1335–1338.

[11] V. Pollex, T. Feld, F. Slomka, U. Margull, R. Mader, and G. Wirrer, "Sufficient real-time analysis for an engine control unit," in *Proc. Int. Conf. Real Time Netw. Syst.*, Sophia Antipolis, France, 2013, pp. 247–254.

[12] T. Feld and F. Slomka, "Sufficient response time analysis considering dependencies between rate-dependent tasks," in *Proc. Conf. Design Autom. Test Europe*, Grenoble, France, 2015, pp. 519–524.

[13] R. I. Davis, T. Feld, V. Pollex, and F. Slomka, "Schedulability tests for tasks with variable rate-dependent behaviour under fixed priority scheduling," in *Proc. IEEE Real Time Embedded Technol. Appl. Symp.*, Berlin, Germany, 2014, pp. 51–62.

[14] A. Biondi, A. Melani, M. Marinoni, M. D. Natale, and G. Buttazzo, "Exact interference of adaptive variable-rate tasks under fixed-priority scheduling," in *Proc. Euromicro Conf. Real Time Syst.*, Madrid, Spain, 2014, pp. 165–174.

[15] A. Biondi, M. D. Natale, and G. Buttazzo, "Response-time analysis for real-time tasks in engine control applications," in *Proc. IEEE/ACM Conf. Cyber Phys. Syst.*, Seattle, WA, USA, 2015, pp. 120–129.

[16] A. Biondi, M. Di Natale, and G. Buttazzo, "Response-time analysis of engine control applications under fixed-priority scheduling," *IEEE Trans. Comput.*, vol. 67, no. 5, pp. 687–703, May 2018.

[17] A. Biondi, M. D. Natale, G. C. Buttazzo, and P. Pazzaglia, "Selecting the transition speeds of engine control tasks to optimize the performance," *ACM Trans. Cyber Phys. Syst.*, vol. 2, no. 1, pp. 1–26, Feb. 2018.

[18] T. Feld and F. Slomka, "Exact interference of tasks with variable rate-dependent behavior," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 5, pp. 954–967, May 2018.

[19] T. Feld and F. Slomka, "A sufficient response time analysis considering angular phases between rate-dependent tasks," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 38, no. 11, pp. 2008–2021, Nov. 2019.

[20] W.-H. Huang and J.-J. Chen, "Techniques for schedulability analysis in mode change systems under fixed-priority scheduling," in *Proc. IEEE Conf. Embedded Real Time Comput. Syst. Appl.*, Hong Kong, 2015, pp. 176–186.

[21] G. C. Buttazzo, E. Bini, and D. Buttle, "Rate-adaptive tasks: Model, analysis, and design issues," in *Proc. Conf. Design Autom. Test Europe*, Dresden, Germany, 2014, pp. 1–6.

[22] Z. Guo and S. K. Baruah, "Uniprocessor EDF scheduling of AVR task systems," in *Proc. ACM/IEEE Conf. Cyber Phys. Syst.*, Seattle, WA, USA, 2015, pp. 159–168.

[23] A. Biondi and G. Buttazzo, "Engine control: Task modeling and analysis," in *Proc. Conf. Design Autom. Test Europe*, Grenoble, France, 2015, pp. 525–530.

[24] A. Biondi, G. Buttazzo, and S. Simoncelli, "Feasibility analysis of engine control tasks under EDF scheduling," in *Proc. Euromicro Conf. Real Time Syst.*, Lund, Sweden, 2015, pp. 139–148.

[25] A. Biondi, "Analysis and design optimization of engine control software," Ph.D. dissertation, Scuola Superiore Sant Anna, Pisa, Italy, 2017.

[26] M. Mohaqeqi, J. Abdullah, P. Ekberg, and W. Yi, "Refinement of workload models for engine controllers by state space partitioning," in *Proc. Euromicro Conf. Real Time Syst.*, 2017, pp. 1–22.

[27] M. Stigge and W. Yi, "Graph-based models for real-time workload: A survey," *Real Time Syst.*, vol. 51, no. 5, pp. 602–636, 2015.

[28] C. Peng and H. Zeng, "Response time analysis of digraph real-time tasks scheduled with static priority: Generalization, approximation, and improvement," *Real Time Syst.*, vol. 54, no. 1, pp. 91–131, 2018.

[29] S. K. Bijinemula, A. Willcock, T. Chantem, and N. Fisher, "An efficient knapsack-based approach for calculating the worst-case demand of AVR tasks," in *Proc. IEEE Real Time Syst. Symp.*, 2018, pp. 384–395.

[30] A. Biondi and G. Buttazzo, "Real-time analysis of engine control applications with speed estimation," in *Proc. Conf. Design Autom. Test Europe*, Dresden, Germany, 2016, pp. 193–198.

[31] C. Peng, Y. Zhao, and H. Zeng. (2018). *Schedulability Analysis of Adaptive Variable-Rate Tasks With Dynamic Switching Speeds*. [Online]. Available: https://github.com/ChaoPeng13/EvaluateDynamicAVR

[32] C. Peng, Y. Zhao, and H. Zeng, "Schedulability analysis of adaptive variable-rate tasks with dynamic switching speeds," in *Proc. IEEE Real Time Syst. Symp.*, Nashville, TN, USA, 2018, pp. 396–407.

[33] B. Kalyanasundaram and K. Pruhs, "Speed is as powerful as clairvoyance," *J. ACM*, vol. 47, no. 4, pp. 617–643, 2000.

[34] E. Bini and G. C. Buttazzo, "Measuring the performance of schedulability tests," *Real Time Syst.*, vol. 30, nos. 1–2, pp. 129–154, 2005.

[35] T. J. Barlow, S. Latham, I. McCrae, and P. G. Boulter, "A reference book of driving cycles for use in the measurement of road vehicle emissions," TRL, Crowthorne, U.K., Rep. PPR354, 2009.