Schedulability Analysis of Adaptive Variable-Rate Tasks with Dynamic Switching Speeds

Chao Peng*, Yecheng Zhao†, Haibo Zeng†
*National University of Defense Technology, China. Email: pengchao06@gmail.com
†Virginia Tech, USA. Email: {zyecheng,hbzeng}@vt.edu

***Abstract*—In real-time embedded systems certain tasks are activated according to a rotation source, such as angular tasks in engine control unit triggered whenever the engine crankshaft reaches a specific angular position. To reduce the workload at high speeds, these tasks also adopt different implementations at different rotation speed intervals. However, the current studies limit to the case that the switching speeds at which task implementations should change are configured at design-time. In this paper, we propose to study the task model where switching speeds are dynamically adjusted. 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. Experiments on synthetic task systems demonstrate that the proposed approach provides substantial benefits on system schedulability.**

I. Introduction

Modern real-time embedded systems may contain tasks that respond to external events that are generated by a rotation source. Hence, their activation period and deadline are dependent on the angular speed. Also, to avoid CPU overload on the hosting 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 the literature [1]. An example is the engine control system in internal combustion vehicles, which determines the timing and amount of fuel injected in the engine. Certain software tasks (called *angular tasks*) in it are triggered at predefined rotation angles of the engine crankshaft. It adopts different control strategies at different engine rotation speed intervals [2]. The most sophisticated control strategy (e.g., with multiple fuel injections during one engine revolution) has the best performance (in terms of emission and fuel efficiency), but also comes with the highest amount of computational demand.

The existing studies on systems with AVR tasks all assume that the *switching speeds* (at which AVR tasks switch implementations) is performed *offline*. This means that the optimization of switching speeds will have to be based on design-time information, which is clearly suboptimal. 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, or *dAVR* tasks. In contrast, the AVR tasks in systems with statically configured switching speeds are called static AVR tasks or *sAVR* tasks. Below we illustrate with automotive engine control.

The upcoming era of Connected and Automated Vehicles (CAVs) 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, 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 [3], including path and speed planning [4], [5], vehicle dynamics control [6], and as a potential application of our work, engine control [7].

Specifically, the engine control parameters including the switching speeds are configured at design time, typically using standard *driving cycles* (i.e., a series of data points representing the vehicle speed versus time). This may result in noticeably suboptimal engine performance as the driving cycles used at design time can be substantially different from the actual one. With the CAV techniques making the driving profile readily predictable [7], it becomes possible to dynamically adjust the switching speeds according to the upcoming driving cycle.

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

Related Work. Systems with sAVR tasks are studied in a number of papers in the real-time systems community, see a recent survey [1]. The model from Buttle [2] introduced above is adopted by most researchers, with a couple of exceptions. Kim et al. [8] propose the rhythmic task model and associated analysis that have a few restrictions, e.g., the inter-release time is shortened by a fixed ratio during any acceleration. Pollex et al. [9], [10] assume angular task release and WCET are independent, which may lead to high pessimism in the analysis. Feld and Slomka [11] consider 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 [2]).

Below we summarize the related work that pertains to the model by Buttle [2]. Much of the research for the schedulability analysis focuses on dealing with the major difficulty that both the WCETs and inter-release times of jobs from an angular task strongly depend on the engine rotation speed. Our review below focus on such techniques. We note that there are several variations on the assumptions, task models, and naming of angular tasks (AVR, tasks with Variable Rate-dependent Behavior, etc.), and refer the readers to [1] for details.

For systems with fixed priority scheduling, Davis et al. [12] present 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. [13] propose 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 [14], [15], and a set of design optimization techniques is proposed to optimize the switching speeds at design time [16], [17]. Feld and Slomka [18] improve the runtime of schedulability analysis [13]–[15], which is exact if the maximum acceleration and deceleration have the same absolute value. Huang and Chen [19] assign each mode of an angular task with a unique priority, and propose a utilization-based schedulability test.

With respect to EDF scheduled systems, a number of sufficient utilization-based schedulability tests are presented [20]–[22]. Differently, Biondi et al. [23], [24] propose an exact analysis based on the concept of dominant speed as in [13]. Mohaqeqi et al. [25] propose to partition the speed space and transform angular tasks to digraph real-time (DRT) tasks [26].

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. Specifically, unlike the prior work [13]–[15], [18], [23]–[25], 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. We summarize the contributions and paper organization as follows.

• In Section II, we present the system model, including the model of dAVR tasks.

• In Section III, we develop schedulability analysis techniques for a periodic task interfered by dAVR tasks. Specifically, we propose a new digraph-based task model (called dynamic DRT or dDRT task model), and transform a dAVR task to a dDRT task to approximate its interference. We prove that the task transformation is safe, and there exists no exact transformation. 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.

• In Section IV, we describe the analysis of dAVR tasks.

• In Section V, we use synthetic systems to show the benefits of the proposed model and analysis technique in terms of system schedulability.

II. System Model

Our system model extends the studies on AVR tasks (e.g., [13], [14], [25]). The list of notations is summarized in the table below.

Notation	Definition
θ	Rotation angle
ω	Angular speed
$\omega^{\min}$	Minimum angular speed
$\omega^{\max}$	Maximum angular speed
α	Angular acceleration
$\alpha^{\min}$	Minimum angular acceleration
$\alpha^{\max}$	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	Angular phase of all AVR tasks
Θ	Angular 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
$T^{\min}(\omega_k, \omega_{k+1})$	Minimum time to rotate Θ angles, with initial speed ω_k and final speed ω_{k+1}
$T^{\min}(\omega)$	Simplified notation for $T^{\min}(\omega, \omega)$
$T^{\max}(\omega_k, \omega_{k+1})$	Maximum time to rotate Θ angles, with initial speed ω_k and final speed ω_{k+1}
$T^{\max}(\omega)$	Simplified notation for $T^{\max}(\omega, \omega)$
$\mathcal{SM}_i = \{(SC_i^m, \varsigma\omega_i^m)\}$	Set of execution modes of an sAVR task τ_i^*
$\mathcal{SC}_i(\omega)$	WCET of a job of an sAVR task τ_i^* released at speed ω
$\mathcal{T} = \{\gamma_1, \cdots, \gamma_T\}$	Set of reconfiguration times
T	Number 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 dynamic AVR task τ_i^*
$\mathcal{C}_i(t, \omega)$	WCET of a job released by a dynamic AVR 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 with 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
B	Number of speed intervals in the partition $\mathcal{B}$
$\tau_D^* = (\mathbb{V}, \mathbb{E})$	dDRT task, where $\mathbb{V}$ is the set of vertices, and $\mathbb{E}$ is the set of edges
$v_i.\mathcal{C}(t)$	WCET function of a vertex v_i in τ_D^*
$p^{\min}(v_i, v_j)$	Minimum inter-release time for edge (v_i, v_j)
$p^{\max}(v_i, v_j)$	Maximum inter-release time for edge (v_i, v_j)
(π, ν)	dDRT job released at time π with type ν
$\mathcal{D} = [(\pi_1, \nu_1), \ldots, (\pi_n, \nu_n)]$	dDRT job sequence
$\mathcal{D}.I(t)$	Interference function of $\mathcal{D}$ over an interval with 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^*
$\mathbb{C} = [(c_1^-, c_1^+, \nu_1), \ldots, (c_n^-, c_n^+, \nu_n)]$	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

The rotation source, e.g., 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 in Γ is either periodic or 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), $D_i \le T_i$ 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, and Θ_i is the angular period. Its angular deadline is $\Delta_i = \lambda_i \cdot \Theta_i$ where $\lambda_i \le 1$ (hence τ_i^* also has a constrained deadline). Similar to [14], we assume the 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, deadline) all depend on the dynamics of the rotation source.

Dynamics of Rotation Source. We assume *instantaneous angular speed at the job release time is known at runtime* [13], [14], [25]. The speed after rotating θ angles given an initial speed ω and a constant acceleration α is calculated as [13]

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

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

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

The minimum inter-release time between them, denoted as $T^{\min}(\omega_k, \omega_{k+1})$, is given as [25]

$$\begin{aligned}
&T^{\min}(\omega_k, \omega_{k+1}) = t_1^u + t_2^u + t_3^u, \text{ where} \\
&t_1^u = \tfrac{\omega^U - \omega_k}{\alpha^{\max}}, t_2^u = \tfrac{1}{\omega^{\max}}(\Theta - \tfrac{(\omega^U)^2 - \omega_k^2}{2\alpha^{\max}} - \tfrac{(\omega^U)^2 - \omega_{k+1}^2}{-2\alpha^{\min}}), \\
&t_3^u = \tfrac{\omega^U - \omega_{k+1}}{-\alpha^{\min}}, \omega^U = \min(\omega^{\max}, \omega^u), \\
&\omega^u = \sqrt{\tfrac{\alpha^{\max}\omega_{k+1}^2 - \alpha^{\min}\omega_k^2 - 2\alpha^{\min}\alpha^{\max}\Theta}{\alpha^{\max} - \alpha^{\min}}}
\end{aligned} \tag{3}$$

Specifically, ω^u denotes the maximum speed such that the rotation angle equals the angular period Θ by accelerating from ω_k to ω^u with maximum acceleration $\alpha^{\max}$ and then decelerating from ω^u to ω_{k+1} with maximum deceleration $|\alpha^{\min}|$. However, the actual maximum speed ω^U is bounded by $\omega^{\max}$. $T^{\min}(\omega_k, \omega_{k+1})$ is achieved by accelerating from ω_k to ω^U with $\alpha^{\max}$ (which takes time t_1^u), staying at a constant speed ω^U for a duration of t_2^u, and then decelerating from ω^U to ω_{k+1} with $|\alpha^{\min}|$ (using time t_3^u). Note $t_2^u = 0$ if $\omega^U = \omega^u$.

Similarly, the maximum inter-release time $T^{\max}(\omega_k, \omega_{k+1})$ is composed of a maximum deceleration, a possible duration of constant speed, and a maximum acceleration

$$\begin{aligned}
&T^{\max}(\omega_k, \omega_{k+1}) = t_1^l + t_2^l + t_3^l, \text{ where} \\
&t_1^l = \tfrac{\omega_k - \omega^L}{-\alpha^{\min}}, t_2^l = \tfrac{1}{\omega^{\min}}(\Theta - \tfrac{\omega_k^2 - (\omega^L)^2}{-2\alpha^{\min}} - \tfrac{\omega_{k+1}^2 - (\omega^L)^2}{2\alpha^{\max}}), \\
&t_3^l = \tfrac{\omega_{k+1} - \omega^L}{\alpha^{\max}}, \quad \omega^L = \begin{cases} \max(\omega^{\min}, \omega^l), & \text{if } x \geq 0 \\ \omega^{\min}, & \text{otherwise} \end{cases} \\
&\omega^l = \sqrt{\tfrac{x}{\alpha^{\max} - \alpha^{\min}}}, x = \alpha^{\max}\omega_k^2 - \alpha^{\min}\omega_{k+1}^2 + 2\alpha^{\min}\alpha^{\max}\Theta
\end{aligned} \tag{4}$$

For convenience, we also use the following simplified notations

$$T^{\min}(\omega_k) = T^{\min}(\omega_k, \omega_k), \quad T^{\max}(\omega_k) = T^{\max}(\omega_k, \omega_k) \tag{5}$$

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

$$\omega_l \rightsquigarrow \omega_{l+1} \wedge T^{\min}(\omega_l, \omega_{l+1}) \leq \sigma_{l+1} - \sigma_l \leq T^{\max}(\omega_l, \omega_{l+1}) \tag{6}$$

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

$$\begin{aligned}
&D_i(\omega) = t_1^d + t_2^d, \text{ where} \\
&t_1^d = \tfrac{\omega^D - \omega}{\alpha^{\max}}, t_2^d = \tfrac{1}{\omega^{\max}}(\lambda_i\Theta - \tfrac{(\omega^D)^2 - \omega^2}{2\alpha^{\max}}), \\
&\omega^D = \min\{\omega^{\max}, \Omega(\omega, \alpha^{\max}, \lambda_i\Theta)\}
\end{aligned} \tag{7}$$

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		

TABLE I: An illustrative example of a dAVR task τ_i^*.

We note a few useful properties as follows. Properties 1 and 3 are proved in [15] by noting the corresponding derivatives (e.g., $T^{\min}(\omega_k, \omega_{k+1})$ with respect to ω_k) are always negative. Property 2 is proved similarly in the appendix.

Property 1: $T^{\min}(\omega_k, \omega_{k+1})$ is strictly decreasing with ω_k and ω_{k+1} [15].

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

Property 3: $D_i(\omega)$ is strictly decreasing with ω [15].

sAVR Task Model. The **static AVR** (or **sAVR**) task model, as proposed in the literature, assumes a fixed configuration including the switching speeds. In this model, 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 $SC_i^m \geq SC_i^{m+1}$ 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

$$\mathcal{SM}_i = \{(SC_i^m, \varsigma\omega_i^m), m = 1, \ldots, SM_i\} \tag{8}$$

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

$$\mathcal{SC}_i(\omega) = SC_i^m \quad \text{if } \omega \in (\varsigma\omega_i^{m-1}, \varsigma\omega_i^m] \tag{9}$$

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 $\mathcal{T} = \{\gamma_1, \cdots, \gamma_T\}$. 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 **dynamic AVR** or **dAVR** task, has a series of execution mode sets defined as

$$\begin{aligned}
&\mathcal{Q}_i = \{(\mathcal{M}_{i,k}, \gamma_k), k = 1, \ldots, T\}, \text{ where} \\
&\mathcal{M}_{i,k} = \{(C_{i,k}^m, \omega_{i,k}^m), m = 1, \ldots, M_{i,k}\}
\end{aligned} \tag{10}$$

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

$$\mathcal{C}_i(t, \omega) = C_{i,k}^m \quad \text{if } t \in [\gamma_k, \gamma_{k+1}) \wedge \omega \in (\omega_{i,k}^{m-1}, \omega_{i,k}^m] \tag{11}$$

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

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 dynamic AVR 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 [14]. 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 $(\sigma_{l+1}, \omega_{l+1})$, $\forall l = 1, \cdots, n-1$ satisfy Eq. (6).

Definition 2 (Interference Function of dAVR Job Sequence): $\forall t \geq 0$, 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,

$$\mathcal{A}.I(t) = \mathcal{C}_A(\sigma_1, \omega_1) + \sum_{l=2}^{n} \delta(\sigma_1 + t, \sigma_l) \cdot \mathcal{C}_A(\sigma_l, \omega_l) \tag{12}$$

where function $\delta(\cdot, \cdot)$ is defined as $\delta(a, b) = \begin{cases} 1 & \text{if } a \geq b \\ 0 & \text{otherwise.} \end{cases}$

We now discuss how to calculate the response time of τ_i. We note that the periodic tasks and the dynamic AVR 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

$$R(\tau_i, \mathcal{A}) = \min_{t>0} \left\{ t \mid C_i + \sum_{\tau_j \in hp(i)} \left\lceil \frac{t}{T_j} \right\rceil C_j + \mathcal{A}.I(t) \leq t \right\} \tag{13}$$

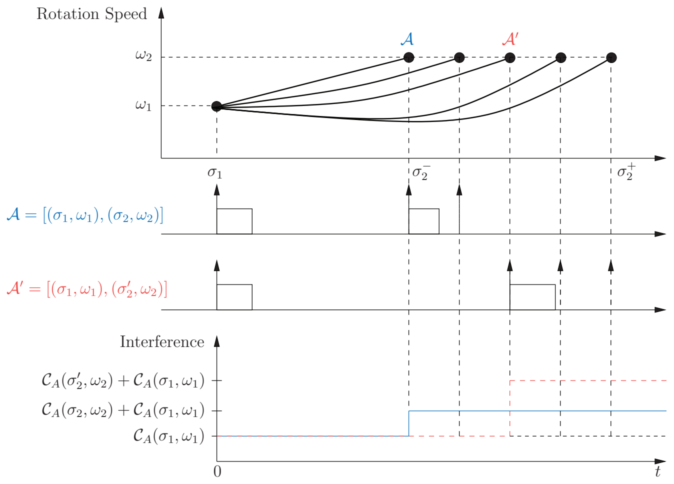
Note in Eq. (13), under certain conditions (i.e., if the utilization is $> 100\%$) no such t exists, and $R(\tau_i, \mathcal{A})$ is defined as infinity. The WCRT $R(\tau_i, \tau_A^*)$ of τ_i is the maximum over all possible dAVR job sequences of τ_A^*

$$R(\tau_i, \tau_A^*) = \max_{\mathcal{A} \in \tau_A^*} R(\tau_i, \mathcal{A}) \tag{14}$$

However, the analysis in Eq. (14) 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 $\mathcal{A} = [(\sigma_1, \omega_1), \ldots, (\sigma_n, \omega_n)]$ and $\mathcal{A}' = [(\sigma_1, \omega_1), \ldots, (\sigma_n', \omega_n)]$ from an AVR task, such that $\sigma_l \leq \sigma_l', \forall l = 2, \cdots, n$. 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 [14], [15], [25] will only consider $\mathcal{A}$. This is safe for sAVR tasks, since the WCET of an sAVR job is independent from its release time (Eq. (9)) and consequently

$$\forall t, \mathcal{A}.I(t) \leq \mathcal{A}'.I(t) \tag{15}$$



Rotation Speed
ω_2
ω_1
$\mathcal{A}$
$\mathcal{A}'$
σ_1
σ_2^-
σ_2^+
$\mathcal{A} = [(\sigma_1, \omega_1), (\sigma_2, \omega_2)]$
$\mathcal{A}' = [(\sigma_1, \omega_1), (\sigma_2', \omega_2)]$
Interference
$\mathcal{C}_A(\sigma_2', \omega_2) + \mathcal{C}_A(\sigma_1, \omega_1)$
$\mathcal{C}_A(\sigma_2, \omega_2) + \mathcal{C}_A(\sigma_1, \omega_1)$
$\mathcal{C}_A(\sigma_1, \omega_1)$
0
t

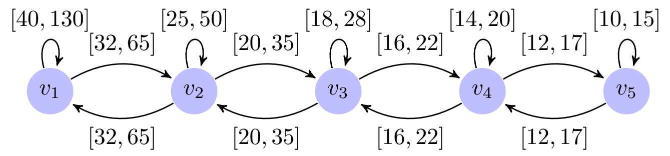
Fig. 1: Illustration of two job sequences $\mathcal{A}$ and $\mathcal{A}'$ released by a dAVR task, and the corresponding interference functions.

This dominance relationship can be generalized to two job sequences released at different angular speeds but still sharing the same sequence of job WCETs. Combining Property 1, it enables the concept of dominant speed, a speed that dominates a range of smaller speeds whenever they always produce sAVR job sequences with the same sequence of job WCETs [13]. Specifically, the dominant speed allows shorter inter-release times than the dominated ones while matching their sequence of job WCETs.

However, as in Eq. (11) the WCET of a dAVR job also depends on its actual release time. Hence, Eq. (15) no longer holds for dAVR tasks, and consequently the analysis developed for sAVR task systems [13], [25] is not directly applicable. An illustrative example is shown in the following.

Example 2: Figure 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 dynamic AVR 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 Eq. (15). Hence, the maximum job inter-release times are needed to correctly model the execution of dAVR tasks and facilitate the calculation of the interference function.

We now present our new analysis techniques. Specifically, to avoid enumerating the speed in the (continuous) speed space, we partition the speed space 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 (Section III-A). We prove the speed space partition (hence any transformation to dDRT task) is safe but sufficient only (Section III-B). Finally, to avoid exhaustive



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: The transformed dDRT task for the dAVR task in Table I with the speed partition $\mathcal{B} = \{500, 1500, 2500, 3500, 4500, 6500\}$ (top); and the WCET function of each vertex (bottom).

enumeration of the job release times, we study the dominance relationship between dDRT job sequences (Section III-C).

We note that the dominant speeds [13] 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 paper, we leverage the more explicit approach of speed partition and task transformation in [25], for its intuitive graphical representation.

A. dAVR to dDRT Transformation

The digraph real-time (DRT) task model [26] uses a directed graph to model a real-time task, where the set of vertices represents 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 that is 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 Eq. (15), this is sufficient.

However, as explained above, we cannot assume jobs are released with minimum inter-release times for dAVR tasks. Hence, the dDRT model defines the maximum inter-release time between vertices in addition to the minimum inter-release time. Also, the WCET of a dAVR task is a function of time to model the fact that the WCET of a dAVR task also depends on the release time. This is formalized in the definition below.

Definition 3 (Dynamic DRT): A dynamic digraph real-time task (**dDRT**) τ_D^* is characterized by a directed graph $(\mathbb{V}, \mathbb{E})$ where the set of vertices $\mathbb{V} = \{v_1, v_2, ...\}$ 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 represent possible flows of control, i.e., the release order of the jobs of τ_D^*. An 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 give an example below.

Example 3: Figure 2 illustrates an example dDRT task with five vertices. 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 $[12, 17]$ms. The WCET function of each vertex is shown in the table.

We now define a speed partition and the corresponding task transformation where (a) each speed interval is mapped to a distinct vertex in the dDRT task; (b) the 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

- they partition the complete speed range $(\omega^{\min}, \omega^{\max}]$ (hence $\beta_0 = \omega^{\min}$, $\beta_B = \omega^{\max}$);
- 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., $\mathcal{B} = \{\beta_0, \beta_1, \cdots, \beta_B\}$. 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

$$\mathcal{B} = \{\omega_{A,k}^m | \forall k = 1 \cdots T, \ \forall m = 1 \cdots M_{A,k}\} \cup \{\omega^{\min}, \omega^{\max}\} \tag{16}$$

Given a valid speed partition $\mathcal{B} = \{\beta_0, \beta_1, \cdots, \beta_B\}$, the dDRT task $\tau_D^* = (\mathbb{V}, \mathbb{E})$ can be constructed as follows.

- $\mathbb{V}$ is composed of a set of B vertices $\{v_1, \cdots, v_B\}$, 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 $(\beta_{i-1}, \beta_i]$.
- 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

$$\begin{cases} p^{\min}(v_i, v_j) = \min_{\forall \beta_i' \rightsquigarrow \beta_j'} \{T^{\min}(\beta_i', \beta_j')\}, \\ p^{\max}(v_i, v_j) = \max_{\forall \beta_i' \rightsquigarrow \beta_j'} \{T^{\max}(\beta_i', \beta_j')\} \end{cases} \tag{17}$$

$p^{\min}(v_i, v_j)$ and $p^{\max}(v_i, v_j)$ can be efficiently calculated by considering only three pairs of β_i' and β_j' as in [25].

Example 4: Considering the dAVR task in Table I and the speed partition $\mathcal{B} = \{500, 1500, 2500, 3500, 4500, 6500\}$ (rpm, revolutions per minute). $\mathcal{B}$ is the smallest valid speed partition, and Figure 2 gives the transformed dDRT task, where vertices $v_1 \cdots v_5$ represent the intervals $(500, 1500]$, $(1500, 2500]$, $(2500, 3500]$, $(3500, 4500]$, $(4500, 6500]$ respectively. The inter-release times are simplified for illustration purposes and may not match the actual rotational dynamics.

B. Safety and Pessimism of the Transformation

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

Definition 5 (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 6 (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.,

$$\forall t \geq 0, \;\; \mathcal{D}.I(t) = \nu_1.\mathcal{C}(\pi_1) + \sum_{l=2}^{n} \delta(\pi_1 + t, \pi_l)\; \nu_l.\mathcal{C}(\pi_l) \quad (18)$$

With these two definitions, similar to Eq. (13) and (14), 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

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

$$R(\tau_i, \mathcal{D}) = \min_{t>0} \left\{ t \mid C_i + \sum_{\tau_j \in hp(i)} \left\lceil \frac{t}{T_j} \right\rceil C_j + \mathcal{D}.I(t) \leq t \right\} \quad (19)$$

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

Definition 7 (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 8 (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.

Theorem 1: The task transformation with any valid speed partition is safe.

Proof. Consider any dAVR task τ_A^* and its transformed dDRT task τ_D^* with any valid speed partition. For any dAVR job sequence $\mathcal{A} = [(\sigma_1, \omega_1), \dots, (\sigma_n, \omega_n)]$ in τ_A^*, we construct a dDRT job sequence $\mathcal{D} = [(\pi_1, \nu_1), \dots, (\pi_n, \nu_n)]$ in τ_D^* where $\forall l \leq n$, $\pi_l = \sigma_l$ and ω_l is in the speed interval $(\beta_{l-1}, \beta_l]$ of ν_l. $\mathcal{D}$ satisfies Definition 5 since $\forall l < n, \omega_l \rightsquigarrow \omega_{l+1}$, and

$$\begin{aligned} p^{\min}(\nu_l, \nu_{l+1}) &\leq T^{\min}(\omega_l, \omega_{l+1}) \leq \pi_{l+1} - \pi_l = \sigma_{l+1} - \sigma_l \\ &\leq T^{\max}(\omega_l, \omega_{l+1}) \leq p^{\max}(\nu_l, \nu_{l+1}) \end{aligned}$$

Also, $\nu_l.\mathcal{C}(\pi_l) = \mathcal{C}_A(\sigma_l, \omega_l)$ since $\omega_l \in (\beta_{l-1}, \beta_l]$. This implies that $\forall t \geq 0, \mathcal{A}.I(t) = \mathcal{D}.I(t)$ and consequently $R(\tau_i, \mathcal{A}) = R(\tau_i, \mathcal{D})$. As $\mathcal{A}$ is an arbitrary dAVR job sequence of τ_A^*, we must have $R(\tau_i, \tau_A^*) \leq R(\tau_i, \tau_D^*)$. $\square$

Theorem 1 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, shown in the following theorem.

Theorem 2: The task transformation with any valid speed partition is not exact for any rotation source with $\omega^{\max} > \omega^t = \sqrt{\frac{25}{12} \cdot \frac{-\alpha^{\max}\alpha^{\min}\Theta}{\alpha^{\max} - \alpha^{\min}}}$.

Before proving the theorem, we first introduce a lemma on the minimum and maximum inter-release times of any two consecutive jobs released with the same speed.

Lemma 3: For any $\omega > \omega^t = \sqrt{\frac{25}{12} \cdot \frac{-\alpha^{\max}\alpha^{\min}\Theta}{\alpha^{\max} - \alpha^{\min}}}$, it always holds that $T^{\min}(\omega) < T^{\max}(\omega) < 2T^{\min}(\omega)$.

Proof. By the definitions of $T^{\min}(\omega)$ and $T^{\max}(\omega)$, obviously $T^{\min}(\omega) < T^{\max}(\omega)$.

Now we prove $T^{\max}(\omega) < 2T^{\min}(\omega)$. For convenience, let $a = \alpha^{\max}\omega^2 - \alpha^{\min}\omega^2, b = -\alpha^{\min}\alpha^{\max}\Theta$. Note that $\alpha^{\min} < 0$, hence $a > 0, b > 0$. Also, since $\omega > \sqrt{\frac{25}{12} \cdot \frac{-\alpha^{\max}\alpha^{\min}\Theta}{\alpha^{\max} - \alpha^{\min}}}$, $a - 2b > a - \frac{25}{12}b > 0$. Thus, in Eq. (4), $x = a - 2b > 0$, $\omega^l > 0$, and $\omega^L = \max(\omega^l, \omega^{\min})$.

`Case 1`: We first consider the case when $\omega^u \leq \omega^{\max}$ in Eq. (3) and $\omega^l \geq \omega^{\min}$ in Eq. (4). Thus, $\omega^U = \omega^u$ and $t_2^u = 0$ in Eq. (3), $\omega^L = \omega^l$ and $t_2^l = 0$ in Eq. (4). In this case, $T^{\min}(\omega)$ and $T^{\max}(\omega)$ can be rewritten as

$$\begin{aligned} & T^{\min}(\omega) \\ = \;& \left(\tfrac{1}{\alpha^{\max}} - \tfrac{1}{\alpha^{\min}}\right)\left(\sqrt{\tfrac{\alpha^{\max}\omega^2 - \alpha^{\min}\omega^2 - 2\alpha^{\min}\alpha^{\max}\Theta}{\alpha^{\max} - \alpha^{\min}}} - \omega\right) \\ = \;& \sqrt{\tfrac{1}{\alpha^{\max} - \alpha^{\min}}}\left(\tfrac{1}{\alpha^{\max}} - \tfrac{1}{\alpha^{\min}}\right)\left(\sqrt{a + 2b} - \sqrt{a}\right) \\ & T^{\max}(\omega) \\ = \;& \left(\tfrac{1}{\alpha^{\max}} - \tfrac{1}{\alpha^{\min}}\right)\left(\omega - \sqrt{\tfrac{\alpha^{\max}\omega^2 - \alpha^{\min}\omega^2 + 2\alpha^{\min}\alpha^{\max}\Theta}{\alpha^{\max} - \alpha^{\min}}}\right) \\ = \;& \sqrt{\tfrac{1}{\alpha^{\max} - \alpha^{\min}}}\left(\tfrac{1}{\alpha^{\max}} - \tfrac{1}{\alpha^{\min}}\right)\left(\sqrt{a} - \sqrt{a - 2b}\right) \end{aligned}$$

Hence,

$$\begin{aligned} & T^{\max}(\omega) < 2T^{\min}(\omega) \\ \Leftrightarrow \;& 2\sqrt{a + 2b} + \sqrt{a - 2b} > 3\sqrt{a} \\ \Leftrightarrow \;& 4(a + 2b) + (a - 2b) + 4\sqrt{(a + 2b)(a - 2b)} > 9a \\ \Leftrightarrow \;& 4\sqrt{(a + 2b)(a - 2b)} > 4a - 6b \\ \Leftrightarrow \;& 16a^2 - 64b^2 > 16a^2 + 36b^2 - 48ab \\ \Leftrightarrow \;& a > \tfrac{25}{12}b \;\Leftrightarrow\; \omega > \sqrt{\tfrac{25}{12} \cdot \tfrac{-\alpha^{\max}\alpha^{\min}\Theta}{\alpha^{\max} - \alpha^{\min}}} \end{aligned}$$

`Case 2`: If $\omega^u > \omega^{\max}$ in Eq. (3), by the definition of $T^{\min}(\omega)$ [25], it must be no smaller than the one as if there was no limit on the maximum angular speed (denoted as $\tilde{T}^{\min}(\omega)$). Similarly, if $0 < \omega^l < \omega^{\min}$ in Eq. (4), $T^{\max}(\omega)$ must be no larger than the one as if there was no limit on the minimum speed (denoted as $\tilde{T}^{\max}(\omega)$). By the proof of `Case 1`, $\tilde{T}^{\max}(\omega) < 2\tilde{T}^{\min}(\omega)$. Hence, we have $T^{\max}(\omega) \leq \tilde{T}^{\max}(\omega) < 2\tilde{T}^{\min}(\omega) \leq 2T^{\min}(\omega)$. $\square$

We now prove Theorem 2. Our intuition is that an inter-release time of an edge (v_i, v_j) in dDRT may not be achievable by all angular speed pairs associated with v_i and v_j. Consequently, *for some dDRT job sequence, there may not exist a valid dAVR job sequence corresponding to it.*

Proof of Theorem 2. Let $\mathcal{B} = \{\beta_0, \cdots, \beta_{B-1}, \omega^{\max}\}$ denote any valid speed partition. Since $\omega^{\max} > \omega^t$, by Lemma 3 and Property 2, there must exist a sufficiently small $\epsilon > 0$ such that $\omega^{\max} - \epsilon \in (\beta_{B-1}, \omega^{\max}]$ and $T^{\max}(\omega^{\max}) < T^{\max}(\omega^{\max} - \epsilon) < 2T^{\min}(\omega^{\max})$. For simplicity, let $p_1 = T^{\min}(\omega^{\max})$, $p_2 = T^{\max}(\omega^{\max} - \epsilon)$. Hence $p_1 < p_2 < 2p_1$.

We now construct an example task system to show the task transformation is pessimistic. Consider a task system containing a dAVR task τ_A^* and a periodic task τ_i with priority lower than τ_A^*. The parameters of τ_i are set as $C_i = 2p_1 + p_2 - 5\kappa$, $T_i = 2p_1 + p_2 + 3\kappa$, $D_i = 2p_1 + p_2 + 2\kappa$, where $0 < \kappa < p_1 - \frac{1}{2}p_2$. The WCET function of τ_A^* is

$$\forall \omega, \; \mathcal{C}_A(t, \omega) = \begin{cases} 2\kappa, & \text{if } t \in [0, p_1] \cup [p_1 + p_2, 2p_1 + p_2] \\ \kappa, & \text{otherwise.} \end{cases}$$

Let τ_D^* denote the transformed dDRT task from τ_A^* with speed partition $\mathcal{B}$. Figure 3 illustrates τ_D^*, where v_B is the vertex corresponding to the speed interval $(\beta_{B-1}, \omega^{\max}]$, and v_k represents the speed interval $(\beta_{k-1}, \beta_k]$ ($k = 0, \dots, B-1$). Thus, the WCET function of v_B can be calculated by $v_B.\mathcal{C}(t) = \mathcal{C}_A(t, \omega^{\max})$. Moreover, $p^{\min}(v_B, v_B) =$

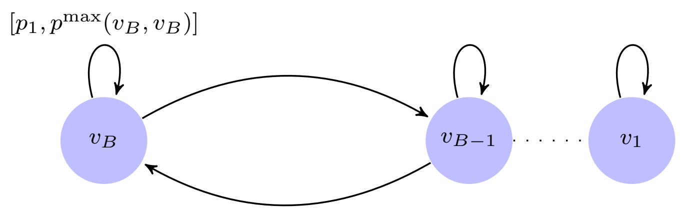


Fig. 3: The transformed dDRT from τ_A^* with speed partition $\mathcal{B}$. It only shows the label over the edge (v_B, v_B) as is needed by the proof of Theorem 2, and omits those for all other edges.

$T^{\min}(\omega^{\max}) = p_1$ and $p^{\max}(v_B, v_B) \geq T^{\max}(\omega^{\max} - \epsilon) = p_2$. This dDRT task permits a job sequence $\mathcal{D} = [(0, v_B), (p_1, v_B), (p_1 + p_2, v_B), (2p_1 + p_2, v_B)]$. The total workload from τ_i and $\mathcal{D}$ satisfies

$$\begin{cases} C_i + \mathcal{D}.I(0) & = 2p_1 + p_2 - 3\kappa \\ & > 2p_1 + p_2 - 3(p_1 - \frac{1}{2}p_2) \\ & = \frac{5}{2}p_2 - p_1 > p_1 \\ C_i + \mathcal{D}.I(p_1) & = 2p_1 + p_2 - \kappa \\ & > 2p_1 + p_2 - (p_1 - \frac{1}{2}p_2) \\ & = p_1 + \frac{3}{2}p_2 > p_1 + p_2 \\ C_i + \mathcal{D}.I(p_1 + p_2) & = 2p_1 + p_2 + \kappa > 2p_1 + p_2 \\ C_i + \mathcal{D}.I(2p_1 + p_2) & = 2p_1 + p_2 + 3\kappa \end{cases}$$

Hence, the response time of τ_i interfered by $\mathcal{D}$ must be $R(\tau_i, \mathcal{D}) \geq 2p_1 + p_2 + 3\kappa$. This means $R(\tau_i, \tau_D^*) \geq 2p_1 + p_2 + 3\kappa > D_i$, and τ_i is deemed to be unschedulable.

However, the accumulative interference generated by τ_A^* within any time window of length D_i must be less than 8κ (hence at most 7κ). Thus, the WCRT of τ_i must be no more than $2p_1 + p_2 + 2\kappa = D_i$, and τ_i is actually schedulable. We prove this by contradiction.

By Property 1, $p_1 = T^{\min}(\omega^{\max})$ is the minimum inter-release time between any two jobs of τ_A^*. Since $D_i = 2p_1 + p_2 + 2\kappa < 2p_1 + p_2 + 2p_1 - p_2 = 4p_1$, within any time window of length D_i, τ_A^* can release at most four jobs. If the total interference of these four jobs is 8κ, each of them must have a WCET of 2κ and consequently be released in $[0, p_1] \cup [p_1 + p_2, 2p_1 + p_2]$. Hence, there are two jobs released in $[0, p_1]$ and two in $[p_1 + p_2, 2p_1 + p_2]$. This requires all of them be released with speed $\omega^{\max}$, as $p_1 = T^{\min}(\omega^{\max})$ is only achievable between two jobs both released with speed $\omega^{\max}$. The only such dAVR job sequence is $\mathcal{A} = [(0, \omega^{\max}), (p_1, \omega^{\max}), (p_1+p_2, \omega^{\max}), (2p_1+p_2, \omega^{\max})]$. However, in $\mathcal{A}$ the second and third jobs are separated by p_2, while both released with speed $\omega^{\max}$. This is impossible as $p_2 = T^{\max}(\omega^{\max} - \epsilon) > T^{\max}(\omega^{\max})$ (Property 2). $\square$

Remark 1: We remark that the condition $\omega^{\max} > \omega^t = \sqrt{\frac{25}{12} \cdot \frac{-\alpha^{\max}\alpha^{\min}\Theta}{\alpha^{\max}-\alpha^{\min}}}$ in Theorem 2 is satisfied by typical engine dynamics. According to [14], the maximum acceleration/deceleration are typically selected to be able to accelerate/decelerate between the minimum and maximum speeds in about 35 revolutions. By Eq. (1), $\omega^{\max} = \sqrt{(\omega^{\min})^2 + 70\alpha^{\max}\Theta} \geq \sqrt{70\alpha^{\max}\Theta}$. Meanwhile, $\omega^t = \sqrt{\frac{25}{12} \cdot \frac{-\alpha^{\max}\alpha^{\min}\Theta}{\alpha^{\max}-\alpha^{\min}}} \leq \sqrt{\frac{25}{12}\alpha^{\max}\Theta}$. Hence, $\omega^{\max} > 5.79\ \omega^t$.

Although Theorem 2 proves that it is typically impossible to find an exact task transformation (and an exact speed partition), we note that the analysis on the transformed dDRT task system has a higher accuracy (but a longer runtime) with a finer speed partition. In our experiments (Section V) we use the speed partition in [13], [25] that is exact for sAVR tasks. Specifically, the speed partition is obtained by applying their generation procedures to one sAVR task, whose switching speeds are set as $\omega_{A,k}^m$ ($\forall k = 1 \cdots T,\ \forall m = 1 \cdots M_{A,k}$).

C. 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 Eq. (19) 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 9 (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 $\mathbb{C} = [(c_1^-, c_1^+, \nu_1), \ldots, (c_n^-, c_n^+, \nu_n)]$, is a sequence of release time ranges $[c_l^-, c_l^+]$ and vertices ν_l that satisfies

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

In Definition 9, 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 (ν_l, ν_{l+1}) is an edge of τ_D^*, and $[c_{l+1}^-, c_{l+1}^+]$ shall be reachable from $[c_l^-, c_l^+]$.

Definition 10 (Critical Job Sequence of $\mathbb{C}$): $\mathcal{D} = [(\pi_1, \nu_1), \ldots, (\pi_n, \nu_n)]$ is called a job sequence of $\mathbb{C} = [(c_1^-, c_1^+, \nu_1), \ldots, (c_n^-, c_n^+, \nu_n)]$, 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 $\mathbb{C}$;

$\mathcal{D}^c = [(\pi_1^c, \nu_1), \ldots, (\pi_n^c, \nu_n)] \in \mathbb{C}$ 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 5.
7) $\forall \mathcal{D} \in \mathbb{C} \cap \tau_D^*, \forall 1 \leq l \leq n,\ \pi_l^c - \pi_1^c \leq \pi_l - \pi_1 \quad (20)$

Definition 11 (Job Sequence Subset): For two job sequence sets $\mathbb{C}$ and $\mathbb{C}'$, $\mathbb{C}'$ is called a subset of $\mathbb{C}$, denoted as $\mathbb{C}' \subseteq \mathbb{C}$, if $\forall \mathcal{D} \in \mathbb{C}'$, it is $\mathcal{D} \in \mathbb{C}$.

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

Algorithm 1 Constructing a collection of common-WCET dDRT job sequence sets for a given path $(\nu_1, \ldots, \nu_n)$.

```
1:  procedure ConstructJSSets($\nu_1, \ldots, \nu_n$)
2:      for $k = 1$ to $T-1$ do
3:          $[c_1^-, c_1^+] \leftarrow [\gamma_k, \gamma_{k+1})$;
4:          $\mathfrak{C}_{k,1} \leftarrow \{[(c_1^-, c_1^+, \nu_1)]\}$;
5:          for $l = 2$ to $n$ do
6:              $\mathfrak{C}_{k,l} \leftarrow \emptyset$;
7:              for all $\mathbb{C} \in \mathfrak{C}_{k,l-1}$ do
8:                  $(c_{l-1}^-, c_{l-1}^+, \nu_{l-1}) \leftarrow$ last tuple of $\mathbb{C}$;
9:                  $c_{next}^- \leftarrow c_{l-1}^- + p^{\min}(\nu_{l-1}, \nu_l)$;
10:                 $c_{next}^+ \leftarrow c_{l-1}^+ + p^{\max}(\nu_{l-1}, \nu_l)$;
11:                 for $j = 1$ to $T-1$ do
12:                     if $c_{next}^- < \gamma_j < \gamma_{j+1} \leq c_{next}^+$ then
13:                         $[c_l^-, c_l^+] \leftarrow [\gamma_j, \gamma_{j+1})$;
14:                     if $\gamma_j \leq c_{next}^- < c_{next}^+ < \gamma_{j+1}$ then
15:                         $[c_l^-, c_l^+] \leftarrow [c_{next}^-, c_{next}^+]$;
16:                     if $\gamma_j \leq c_{next}^- < \gamma_{j+1} \leq c_{next}^+$ then
17:                         $[c_l^-, c_l^+] \leftarrow [c_{next}^-, \gamma_{j+1})$;
18:                     if $c_{next}^- < \gamma_j < c_{next}^+ < \gamma_{j+1}$ then
19:                         $[c_l^-, c_l^+] \leftarrow [\gamma_j, c_{next}^+]$;
20:                     $\mathfrak{C}_{k,l} \leftarrow \mathfrak{C}_{k,l} \cup \{[\mathbb{C}, (c_l^-, c_l^+, \nu_l)]\}$;
21:     return $\mathfrak{C}_{1,n} \cup \cdots \cup \mathfrak{C}_{T-1,n}$;
```

$\mathbb{C} = [(c_1^-, c_1^+, \nu_1), \ldots, (c_n^-, c_n^+, \nu_n)]$ 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.

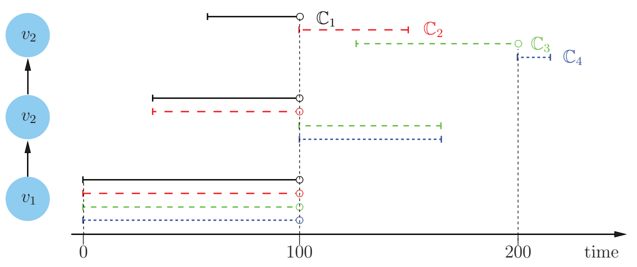
We now explain how to enumerate all necessary job sequence sets. By Definition 9, this involves (i) generating all valid paths (i.e., valid sequences of types of jobs) in the dDRT with limited length n, and (ii) finding all suitable release time ranges. The former follows that of a generic digraph such as the DRT tasks [27]. Given a path $(\nu_1, \cdots, \nu_n)$, Algorithm 1 generates all the related common-WCET job sequence sets.

Specifically, for the interval $[\gamma_k, \gamma_{k+1}), \forall k = 1, \cdots, T-1$ between two consecutive reconfigurations, we use $\mathfrak{C}_{k,l}$ to store the collection of job sequence sets until vertex ν_l. The first tuple (c_1^-, c_1^+, ν_1) sets the release time range as the interval (Line 3), i.e., $[c_1^-, c_1^+] = [\gamma_k, \gamma_{k+1})$ [1]. Then the possible ranges of the following jobs are sequentially set as the maximum possible range reachable from the previous one (Lines 8-10). However, such a range may not satisfy Condition 2 in Definition 9. Hence, it is split by the reconfiguration times to enforce that ν_l has the same WCET in each interval (Lines 11-20). At Line 20, the job sequence set $[\mathbb{C}, (c_l^-, c_l^+, \nu_l)]$ satisfies Definition 9, which is added to the collection $\mathfrak{C}_{k,l}$. Finally, Algorithm 1 returns a collection of the common-WCET dDRT job sequence sets as the union of $\mathfrak{C}_{k,n}, \forall k$. Let $|\mathfrak{C}_{k,l}|$ denote the size of $\mathfrak{C}_{k,l}$. For any k, l, $|\mathfrak{C}_{k,l}| \leq (T-k) \cdot |\mathfrak{C}_{k,l-1}|$. Hence, Algorithm 1 has $O(T^n)$ time complexity.

We illustrate Algorithm 1 with an example below.

Example 5: Given the dDRT task τ_D^* in Figure 2, we consider one path (v_1, v_2, v_2) with an initial tuple $(0, 100, v_1)$.

[1]Note that the range $[c_1^-, c_1^+]$ is closed, but $[\gamma_k, \gamma_{k+1})$ is right-open. In practice, this can be resolved by setting $c_1^+ = \gamma_{k+1} - \epsilon$ where ϵ is a small enough positive number.



(a) Partitioning release time ranges by the reconfiguration times $\gamma_1 = 0, \gamma_2 = 100, \gamma_3 = 200$.

$\mathbb{C}$	(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) Resultant four common-WCET dDRT job sequence sets.

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

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($\mathbb{C}$)
2:      $\tilde{\mathbb{C}} \leftarrow [(\tilde{c}_1^-, \tilde{c}_1^+, \nu_1), \ldots, (\tilde{c}_n^-, \tilde{c}_n^+, \nu_n)]$;
3:      $\tilde{c}_n^- \leftarrow c_n^-$, $\tilde{c}_n^+ \leftarrow c_n^+$;
4:      for $l = n$ to 2 do // Backward Pass
5:          $\tilde{c}_{l-1}^- \leftarrow \max(\tilde{c}_l^- - p^{\max}(\nu_{l-1}, \nu_l), c_{l-1}^-)$;
6:          $\tilde{c}_{l-1}^+ \leftarrow \min(\tilde{c}_l^+ - p^{\min}(\nu_{l-1}, \nu_l), c_{l-1}^+)$;
7:      $\mathcal{D}^c \leftarrow [(\pi_1^c, \nu_1), \ldots, (\pi_n^c, \nu_n)]$;
8:      $\pi_1^c \leftarrow \tilde{c}_1^+$;
9:      for $l = 1$ to $n-1$ do // Forward Pass
10:         $\pi_{l+1}^c \leftarrow \max(\pi_l^c + p^{\min}(\nu_l, \nu_{l+1}), \tilde{c}_{l+1}^-)$;
11:     return $\mathcal{D}^c$;
```

The release time constraint may branch to the next possible vertex v_2 where the edge (v_1, v_2) is labeled as $[32, 65]$. Hence, 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 according the reconfiguration times ($\gamma_2 = 100$ and $\gamma_3 = 200$). Hence, $[57, 150)$ is split into $[57, 100)$ and $[100, 150)$, and $[125, 215]$ is divided into $[125, 200)$ and $[200, 215]$. As a result, we obtain four common-WCET dDRT job sequence sets, shown in Figure 4(b).

In Definition 10, Eq. (20) 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

[2]In Examples 5 and 6, for simplicity we *treat closed and half-open intervals the same*.

$\mathbb{C}$	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

TABLE II: An illustrative example on Algorithm 2: construction of critical job sequences for the four sets in Figure 4(b).

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 call this process as legalization, as we show in Lemma 7 that 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). We note that the necessary splitting of release time ranges by the configuration times generates $\mathbb{C}$ with such illegal regions.

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 ν_{l-1} by satisfying the min/max inter-release times and the time constraint on ν_{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 9–10) to release other jobs as early as possible, subject to the requirement that $\mathcal{D}^c \in \widetilde{\mathbb{C}}$.

Before proving the correctness of Algorithm 2, we first give an example on how it works.

Example 6: Consider the third common-WCET job sequence set in Figure 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

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

(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 II illustrates the generated critical job sequences for the four common-WCET job sequence sets in Figure 4(b).

We now provide a few lemmas on the properties of $\widetilde{\mathbb{C}}$ and $\mathcal{D}^c$ generated by Algorithm 2. The following lemma shows that the resultant $\widetilde{\mathbb{C}}$ is a common-WCET dDRT job sequence set that is also a subset of $\mathbb{C}$.

Lemma 4: $\widetilde{\mathbb{C}}$ satisfies Definition 9, and $\widetilde{\mathbb{C}} \subseteq \mathbb{C}$.

Proof. $\widetilde{\mathbb{C}}$ obviously satisfies Condition 3 of Definition 9, as it contains the same sequence of vertices as $\mathbb{C}$, and $\forall 2 \le l \le n$, $\tilde{c}_{l-1}^- = \max(\tilde{c}_l^- - p^{\max}(\nu_{l-1}, \nu_l), c_{l-1}^-) \ge \tilde{c}_l^- - p^{\max}(\nu_{l-1}, \nu_l)$, $\tilde{c}_{l-1}^+ = \min(\tilde{c}_l^+ - p^{\min}(\nu_{l-1}, \nu_l), c_{l-1}^+) \le \tilde{c}_l^+ - p^{\min}(\nu_{l-1}, \nu_l)$.

Hence, it is sufficient to show that $\forall l \le n, c_l^- \le \tilde{c}_l^- \le \tilde{c}_l^+ \le c_l^+$, which implies that $\widetilde{\mathbb{C}}$ meets Conditions 1 and 2 of Definition 9. We prove it by induction.

Initial Step ($l = n$). By Line 3, $\tilde{c}_n^- = c_n^-$ and $\tilde{c}_n^+ = c_n^+$. Obviously, this means that $c_n^- = \tilde{c}_n^- \le \tilde{c}_n^+ = c_n^+$ since $\mathbb{C}$ satisfies Condition 1 of Definition 9.

Inductive Step. Assume $c_l^- \le \tilde{c}_l^- \le \tilde{c}_l^+ \le c_l^+$ where $l \ge 2$. We consider $\tilde{c}_{l-1}^-$ and $\tilde{c}_{l-1}^+$, which are set to $\max(\tilde{c}_l^- - p^{\max}(\nu_{l-1}, \nu_l), c_{l-1}^-)$ and $\min(\tilde{c}_l^+ - p^{\min}(\nu_{l-1}, \nu_l), c_{l-1}^+)$ respectively. Clearly, $c_{l-1}^- \le \tilde{c}_{l-1}^-$ and $\tilde{c}_{l-1}^+ \le c_{l-1}^+$. We now prove $\tilde{c}_{l-1}^- \le \tilde{c}_{l-1}^+$, by considering four cases.

`Case a`: $\tilde{c}_{l-1}^- = c_{l-1}^-$ and $\tilde{c}_{l-1}^+ = c_{l-1}^+$. This implies that $\tilde{c}_{l-1}^- \le \tilde{c}_{l-1}^+$ since $\mathbb{C}$ satisfies Condition 1 of Definition 9.

`Case b`: $\tilde{c}_{l-1}^- = c_{l-1}^-$ and $\tilde{c}_{l-1}^+ = \tilde{c}_l^+ - p^{\min}(\nu_{l-1}, \nu_l)$. In this case, we have

$$\tilde{c}_{l-1}^- = c_{l-1}^- \overset{(a)}{\le} c_l^- - p^{\min}(\nu_{l-1}, \nu_l) \overset{(b)}{\le} \tilde{c}_l^+ - p^{\min}(\nu_{l-1}, \nu_l) = \tilde{c}_{l-1}^+$$

where inequality (a) is from Condition 3 of Definition 9, and (b) is the inductive assumption.

`Case c`: $\tilde{c}_{l-1}^- = \tilde{c}_l^- - p^{\max}(\nu_{l-1}, \nu_l)$ and $\tilde{c}_{l-1}^+ = c_{l-1}^+$. Similar to `Case b`, it is

$$\tilde{c}_{l-1}^- = \tilde{c}_l^- - p^{\max}(\nu_{l-1}, \nu_l) \overset{(a)}{\le} c_l^+ - p^{\max}(\nu_{l-1}, \nu_l) \overset{(b)}{\le} c_{l-1}^+ = \tilde{c}_{l-1}^+$$

where inequality (a) is the inductive assumption, and (b) is from Condition 3 of Definition 9.

`Case d`: $\tilde{c}_{l-1}^- = \tilde{c}_l^- - p^{\max}(\nu_{l-1}, \nu_l)$ and $\tilde{c}_{l-1}^+ = \tilde{c}_l^+ - p^{\min}(\nu_{l-1}, \nu_l)$. We have

$$\begin{aligned}
\tilde{c}_{l-1}^- &= \tilde{c}_l^- - p^{\max}(\nu_{l-1}, \nu_l)\\
&\overset{(a)}{\le} \tilde{c}_l^- - p^{\min}(\nu_{l-1}, \nu_l) \overset{(b)}{\le} \tilde{c}_l^+ - p^{\min}(\nu_{l-1}, \nu_l) = \tilde{c}_{l-1}^+
\end{aligned}$$

where (a) is due to $p^{\min}(\nu_{l-1}, \nu_l) \le p^{\max}(\nu_{l-1}, \nu_l)$, and (b) is the inductive assumption. □

Furthermore, the following property proves that the resultant $\mathcal{D}^c$ is a job sequence of $\widetilde{\mathbb{C}}$.

Lemma 5: $\mathcal{D}^c \in \widetilde{\mathbb{C}}$.

Proof. Condition 4 of Definition 10 is satisfied by Line 7. We prove by induction that $\forall l \le n, \pi_l^c \in [\tilde{c}_l^-, \tilde{c}_l^+]$.

Initial Step ($l = 1$). π_1^c, set to $\tilde{c}_1^+$, is obviously in $[\tilde{c}_1^-, \tilde{c}_1^+]$.

Inductive Step. Assume $\pi_l^c \in [\tilde{c}_l^-, \tilde{c}_l^+]$ where $l < n$. We note that $\pi_{l+1}^c = \max(\pi_l^c + p^{\min}(\nu_l, \nu_{l+1}), \tilde{c}_{l+1}^-)$ by Line 10. $\pi_{l+1}^c \in [\tilde{c}_{l+1}^-, \tilde{c}_{l+1}^+]$ if $\pi_{l+1}^c = \tilde{c}_{l+1}^-$. Consider the case $\pi_{l+1}^c = \pi_l^c + p^{\min}(\nu_l, \nu_{l+1}) > \tilde{c}_{l+1}^-$. This implies $\pi_{l+1}^c \overset{(a)}{\le}$

$\tilde{c}_l^+ + p^{\min}(\nu_l, \nu_{l+1}) \overset{(b)}{\leq} \tilde{c}_{l+1}^+$, where inequality (a) is the inductive assumption, and inequality (b) is from Line 6. □

We now show by Definition 5, $\mathcal{D}^c$ is a job sequence of τ_D^*.

Lemma 6: $\mathcal{D}^c \in \tau_D^*$.

Proof. It suffices to show that $\forall l < n, p^{\min}(\nu_l, \nu_{l+1}) \leq \pi_{l+1}^c - \pi_l^c \leq p^{\max}(\nu_l, \nu_{l+1})$.

As the final value of π_{l+1}^c is updated in Line 10, we have two cases. The case that $\pi_{l+1}^c = \pi_l^c + p^{\min}(\nu_l, \nu_{l+1})$ is trivial. If $\pi_{l+1}^c = \tilde{c}_{l+1}^- > \pi_l^c + p^{\min}(\nu_l, \nu_{l+1})$, then $\pi_{l+1}^c - \pi_l^c > p^{\min}(\nu_l, \nu_{l+1})$. Moreover, it must be $\pi_{l+1}^c - \pi_l^c = \tilde{c}_{l+1}^- - \pi_l^c \overset{(a)}{\leq} \tilde{c}_{l+1}^- - \tilde{c}_l^- \overset{(b)}{\leq} p^{\max}(\nu_l, \nu_{l+1})$, where inequality (a) is because of Lemma 5, and (b) is correct due to Line 5. □

In addition, the following lemma presents the relationship between the two job sequence sets $\tau_D^* \cap \mathbb{C}$ and $\widetilde{\mathbb{C}}$.

Lemma 7: For any $\mathcal{D} = [(\pi_1, \nu_1), \cdots, (\pi_n, \nu_n)]$ such that $\mathcal{D} \in \tau_D^* \cap \mathbb{C}$, it must be $\mathcal{D} \in \widetilde{\mathbb{C}}$.

Proof. We show by induction that $\forall l \leq n$, $\pi_l \in [\tilde{c}_n^-, \tilde{c}_n^+]$.

Initial Step ($l = n$). Since $\mathcal{D} \in \mathbb{C}$ and $\tilde{c}_n^- = c_n^-$, $\tilde{c}_n^+ = c_n^+$ (Line 3), we have $\pi_n \in [\tilde{c}_n^-, \tilde{c}_n^+]$.

Inductive Step. Assume $\pi_l \in [\tilde{c}_l^-, \tilde{c}_l^+]$ where $l \geq 2$. By Lemma 4 $c_{l-1}^- \leq \tilde{c}_{l-1}^- \leq \tilde{c}_{l-1}^+ \leq c_{l-1}^+$, hence there are only three cases as below.

`Case a`: $\tilde{c}_{l-1}^+ < \pi_{l-1} \leq c_{l-1}^+$. It implies that $\pi_{l-1} > \tilde{c}_{l-1}^+ \overset{(a)}{=} \tilde{c}_l^+ - p^{\min}(\nu_{l-1}, \nu_l) \overset{(b)}{\geq} \pi_l - p^{\min}(\nu_{l-1}, \nu_l)$, which conflicts with $\mathcal{D} \in \tau_D^*$ by Definition 5. Here equality (a) is from Line 6 (note that $\tilde{c}_{l-1}^+ < c_{l-1}^+$), and inequality (b) is the inductive assumption.

`Case b`: $c_{l-1}^- \leq \pi_{l-1} < \tilde{c}_{l-1}^-$. It implies that $\pi_{l-1} < \tilde{c}_{l-1}^- \overset{(a)}{=} \tilde{c}_l^- - p^{\max}(\nu_{l-1}, \nu_l) \overset{(b)}{\leq} \pi_l - p^{\max}(\nu_{l-1}, \nu_l)$, which conflicts with $\mathcal{D} \in \tau_D^*$ by Definition 5. Here equality (a) is from Line 5 (note that $\tilde{c}_{l-1}^- > c_{l-1}^-$), and inequality (b) is the inductive assumption.

`Case c`: $\tilde{c}_{l-1}^- \leq \pi_{l-1} \leq \tilde{c}_{l-1}^+$. It is the only possibility as cases `a` and `b` are impossible. □

With the above four lemmas, we are now ready to prove the correctness of Algorithm 2.

Theorem 8: Given $\mathbb{C} = [(c_1^-, c_1^+, \nu_1), \ldots, (c_n^-, c_n^+, \nu_n)]$, Algorithm 2 generates a critical job sequence $\mathcal{D}^c = [(\pi_1^c, \nu_1), \ldots, (\pi_n^c, \nu_n)]$ according to Definition 10.

Proof. By Lemmas 4–6, Conditions 4–6 of Definition 10 are all satisfied. Also, due to Lemma 7, Condition 7 only requires to prove that $\forall \mathcal{D} = [(\pi_1, \nu_1), \ldots, (\pi_n, \nu_n)] \in \widetilde{\mathbb{C}} \cap \tau_D^*$, $\forall 1 \leq l \leq n, \pi_l^c - \pi_1^c \leq \pi_l - \pi_1$. Before proving it by induction, we note that $\pi_1 \leq \tilde{c}_1^+ = \pi_1^c$.

Initial Step ($l = 1$). This is trivially true since $\pi_1^c - \pi_1^c = \pi_1 - \pi_1 = 0$.

Inductive Step. Assume $\pi_l^c - \pi_1^c \leq \pi_l - \pi_1$ where $l < n$. We consider π_{l+1}^c, which is set to $\max(\pi_l^c + p^{\min}(\nu_l, \nu_{l+1}), \tilde{c}_{l+1}^-)$.

`Case a`: $\pi_{l+1}^c = \tilde{c}_{l+1}^-$. This satisfies $\pi_{l+1}^c - \pi_1^c \leq \pi_{l+1} - \pi_1$ since $\pi_{l+1}^c = \tilde{c}_{l+1}^- \leq \pi_{l+1}$ and $\pi_1^c \geq \pi_1$.

`Case b`: $\pi_{l+1}^c = \pi_l^c + p^{\min}(\nu_l, \nu_{l+1}) > \tilde{c}_{l+1}^-$. Hence,

$$\begin{aligned}\pi_{l+1}^c - \pi_1^c &= \pi_l^c + p^{\min}(\nu_l, \nu_{l+1}) - \pi_1^c \\ &\overset{(a)}{\leq} \pi_l + p^{\min}(\nu_l, \nu_{l+1}) - \pi_1 \overset{(b)}{\leq} \pi_{l+1} - \pi_1\end{aligned}$$

Here inequality (a) is due to $\pi_1^c \geq \pi_1$, and (b) is because $\mathcal{D} \in \tau_D^*$. □

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

$$\begin{aligned}R(\tau_i^*, \sigma, \omega) = \min_{t>0}\{t \mid \mathcal{C}_i(\sigma, \omega) + \sum_{\tau_j^* \in hp^*(i)} \mathcal{C}_j(\sigma, \omega) \\ + \sum_{\tau_j \in hp(i)} \left\lceil \frac{t}{T_j} \right\rceil C_j \leq t\}\end{aligned} \quad (21)$$

We now narrow down the set of jobs of τ_i^* that we need to check for its schedulability. First, by Eq. (11) the WCET of any dAVR job for a given speed ω only changes at the set of reconfiguration times $\mathcal{T} = \{\gamma_1, \cdots, \gamma_T\}$. Hence, it is enough to only consider $\mathcal{T}$ as the set of representative release times for dAVR jobs

$$R(\tau_i^*, \omega) = \max_{\sigma \in \mathcal{T}} R(\tau_i^*, \sigma, \omega) \quad (22)$$

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

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

and consider two consecutive switching speeds ω_l and ω_{l+1} in $\mathbb{W}_i$. By Eq. (11) 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 possible angular speeds in $(\omega_l, \omega_{l+1}]$. It is sufficient to only check the schedulability of jobs of τ_i^* released with a angular speed in $\mathbb{W}_i$. That is, τ_i^* is schedulable if the following condition is satisfied

$$\forall \omega \in \mathbb{W}_i,\ R(\tau_i^*, \omega) \leq D_i(\omega) \quad (23)$$

In the end, the schedulability analysis of τ_i^* in Eq. (21)-(23) 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 [16], [17]: (i) the minimum/maximum rotational speed is 500/6500 rpm; and (ii) the maximum acceleration/deceleration is 1.62×10^{-4} rev/msec^2. Hence, the engine needs 35 revolutions to accelerate/decelerate between the minimum and maximum speeds.

We compare three methods as follows for schedulability analysis of dAVR task systems.

• dAVR: The analysis proposed in this paper, based on the transformation of dAVR tasks to dDRT tasks.

• 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 [14], [15].

• UB: The necessary-only analysis from [16], [17]. It overestimates the workload from a dAVR task τ_i^* in the k-th configuration as the maximum over a series of virtual execution mode sets $\{(C_{i,k}^m, \omega_{i,k}^m), (C_{i,k}^{M_{i,k}}, \omega^{\max})\}, \forall m = 1, \cdots, M_{i,k} - 1$, i.e., one is the m-th mode with WCET $C_{i,k}^m$ executed in the speed interval $(\omega^{\min}, \omega_{i,k}^m]$, the other is the simplest control strategy with WCET $C_{i,k}^{M_{i,k}}$ effective in $(\omega_{i,k}^m, \omega^{\max}]$.

For the methods dAVR and dAVR2sAVR, we use the speed partition in [13], [25] that is exact for schedulability analysis of sAVR tasks. This results in, for example, on average 1122 speed intervals for the systems in Figures 5–6.

Random Task Systems. The random task systems are generated following [14]. 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 $U_A = \max_t \max_\omega \frac{\mathcal{C}_A(t,\omega)}{T^{\min}(\omega)}$. 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 [28] and the period is uniformly distributed between 3 and 100ms. 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 $\gamma_k = (k-1) \cdot T_{sample}$ for $1 \le k \le T$. We set $T_{sample} = 500$ms, which is the sample period in most standard driving cycles [29], [30]. 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 to the product of its utilization and minimum inter-release time $T^{\min}(\omega^{m+1})$.

Results. In the following four experiments, 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. In this sense, the schedulability ratio shown in Figures 5–8 is a normalization w.r.t. UB (hence UB is omitted). Each data point in the figures is the average over 1000 random task sets.

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 the three methods

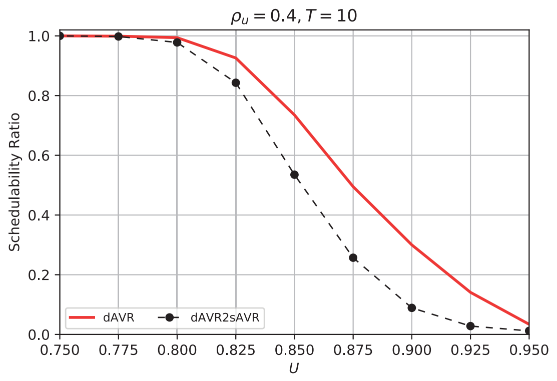


Fig. 5: Schedulability ratio vs. system utilization U.

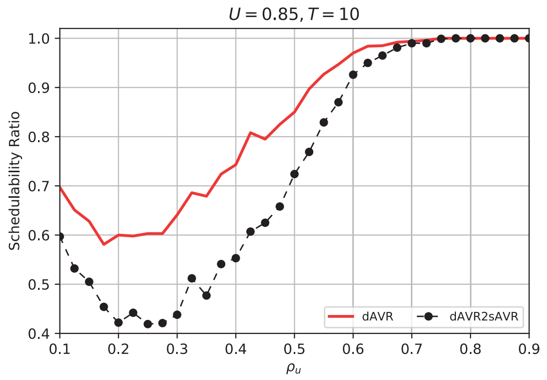


Fig. 6: Schedulability ratio vs. dAVR utilization fraction ρ_u.

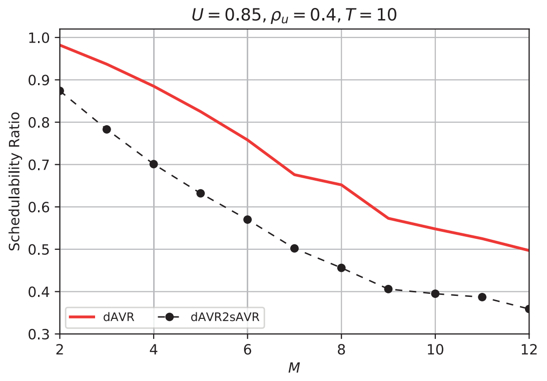


Fig. 7: Schedulability ratio vs. number of modes M.

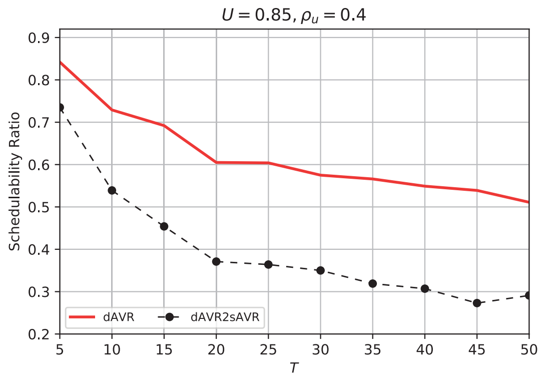


Fig. 8: Schedulability ratio vs. number of configurations T.

are always 1 when $U < 0.75$, hence, Figure 5 only reports the results for $U \geq 0.75$. As shown in the figure, dAVR always has a higher schedulability ratio (hence better analysis accuracy) than dAVR2sAVR. For example, at $U = 0.875$, the schedulability ratio of dAVR is around 24% higher than that of dAVR2sAVR. Finally, dAVR has substantially lower schedulability ratio compared to the necessary-only analysis UB, which indicates that dAVR may be significantly pessimistic. However, we show in [31] that their experimental control performances are always indistinguishable.

The relative comparison between the two methods dAVR and dAVR2sAVR is also consistent 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 Figures 6–8 respectively. As in these figures, dAVR is always more precise than dAVR2sAVR. Also, the difference between between dAVR and dAVR2sAVR is typically around $10\% - 26\%$. The only exception is when $\rho_u \geq 0.6$ in Figure 6, where these two methods all have a schedulability ratio close to 1 (i.e., close to that of UB). This is because in scenarios with a high workload from the dAVR task, the task system is mostly either easily unschedulable (such that UB also deems the system to be unschedulable), or easily schedulable (most periodic tasks have a higher priority than the dAVR task and are not affected by its high workload).

VI. Conclusion

In this paper, we propose the dynamic AVR task model that reconfigure the switching speeds at runtime. To verify the schedulability for such systems, we provided a sufficient response time analysis by partitioning the speed space and approximating the workload of dynamic AVR task with a new type of digraph real-time tasks called dDRT tasks. We also present efficient algorithms for analyzing dDRT task systems that use one critical job sequence to represent the worst case workload of a set of job sequences. Experiments demonstrate that our analysis is substantially more accurate than simple extensions of those for static AVR systems.

Acknowledgment

We would like to thank Alessandro Biondi for sharing the source code of the algorithms presented in [16]. This work is partially funded by NSF Grant No. 1812963.

References

[1] T. Feld, A. Biondi, R. Davis, G. Buttazzo, and F. Slomka, "A survey of schedulability analysis techniques for rate-dependent tasks," *Journal of Systems and Software*, vol. 138, pp. 100–107, 2017.

[2] D. Buttle, "Keynote speech: Real-time in the prime-time," in *Euromicro Conference on Real-Time Systems*, 2012.

[3] J. Guanetti, Y. Kim, and F. Borrelli, "Control of connected and automated vehicles: State of the art and future challenges," *Annual Reviews in Control*, 2018.

[4] J. Lin et al., "A real-time en-route route guidance decision scheme for transportation-based cyber-physical systems," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 3, pp. 2551–2566, 2017.

[5] W. Schwarting, J. Alonso-Mora, and D. Rus, "Planning and decision-making for autonomous vehicles," *Annual Review of Control, Robotics, and Autonomous Systems*, no. 0, 2018.

[6] S. Li et al., "Dynamical modeling and distributed control of connected and automated vehicles: Challenges and opportunities," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 3, pp. 46–58, 2017.

[7] B. Chen et al., "Connected vehicles and powertrain optimization," *Mechanical Engineering Magazine Select Articles*, 139(9):12–18, 2017.

[8] J. Kim, K. Lakshmanan, and R. Rajkumar, "Rhythmic tasks: A new task model with continually varying periods for cyber-physical systems," in *IEEE/ACM Conference on Cyber-Physical Systems*, 2012.

[9] 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 *Conf. Design, Automation Test in Europe*, 2013.

[10] ——, "Sufficient real-time analysis for an engine control unit," in *International Conference on Real-Time Networks and Systems*, 2013.

[11] T. Feld and F. Slomka, "Sufficient response time analysis considering dependencies between rate-dependent tasks," in *Conference on Design, Automation Test in Europe*, 2015.

[12] R. Davis, T. Feld, V. Pollex, and F. Slomka, "Schedulability tests for tasks with variable rate-dependent behaviour under fixed priority scheduling," in *RTAS*, 2014.

[13] A. Biondi, A. Melani, M. Marinoni, M. D. Natale, and G. Buttazzo, "Exact interference of adaptive variable-rate tasks under fixed-priority scheduling," in *Euromicro Conference on Real-Time Systems*, 2014.

[14] A. Biondi, M. D. Natale, and G. Buttazzo, "Response-time analysis for real-time tasks in engine control applications," in *IEEE/ACM Conference on Cyber-Physical Systems*, 2015.

[15] A. Biondi, M. Di Natale, and G. Buttazzo, "Response-time analysis of engine control applications under fixed-priority scheduling," *IEEE Transactions on Computers*, 2017.

[16] A. Biondi, M. D. Natale, and G. Buttazzo, "Performance-driven design of engine control tasks," in *IEEE/ACM Conference on Cyber-Physical Systems*, 2016.

[17] A. Biondi, M. D. Natale, G. Buttazzo, and P. Pazzaglia, "Selecting the transition speeds of engine control tasks to optimize the performance," *ACM Transactions on Cyber-Physical Systems*, vol. 2, no. 1, 2018.

[18] T. Feld and F. Slomka, "Exact interference of tasks with variable rate-dependent behaviour," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, 2017.

[19] W. Huang and J. Chen, "Techniques for schedulability analysis in mode change systems under fixed-priority scheduling," in *IEEE Conference on Embedded and Real-Time Computing Systems and Applications*, 2015.

[20] G. Buttazzo, E. Bini, and D. Buttle, "Rate-adaptive tasks: Model, analysis, and design issues," in *Conference on Design, Automation and Test in Europe*, 2014.

[21] Z. Guo and S. Baruah, "Uniprocessor edf scheduling of avr task systems," in *ACM/IEEE Conference on Cyber-Physical Systems*, 2015.

[22] A. Biondi and G. Buttazzo, "Engine control: Task modeling and analysis," in *Conference on Design, Automation Test in Europe*, 2015.

[23] A. Biondi, G. Buttazzo, and S. Simoncelli, "Feasibility analysis of engine control tasks under edf scheduling," in *Euromicro Conference on Real-Time Systems*, 2015.

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

[25] M. Mohaqeqi, J. Abdullah, P. Ekberg, and W. Yi, "Refinement of Workload Models for Engine Controllers by State Space Partitioning," in *Euromicro Conference on Real-Time Systems*, 2017.

[26] M. Stigge, P. Ekberg, N. Guan, and W. Yi, "The digraph real-time task model," in *IEEE RTAS*, 2011.

[27] N. Guan, C. Gu, M. Stigge, Q. Deng, and W. Yi, "Approximate response time analysis of real-time task graphs," in *IEEE Real-Time Systems Symposium*, 2014.

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

[29] T. Barlow, S. Latham, I. McCrae, and P. Boulter, "A reference book of driving cycles for use in the measurement of road vehicle emissions," *TRL Published Project Report*, 2009.

[30] P. Adak, R. Sahu, and S. Elumalai, "Development of emission factors for motorcycles and shared auto-rickshaws using real-world driving cycle for a typical indian city," *Science of the Total Environment*, vol. 544, pp. 299–308, 2016.

[31] C. Peng, Y. Zhao, and H. Zeng, "Trip-based performance optimization of engine control tasks with dynamic adaptation," Tech. Rep., 2018. [Online] https://github.com/ChaoPeng13/EvaluateDynamicAVR

APPENDIX

Proof of Property 2. We demonstrate the property with ω_k, while the property with ω_{k+1} can be similarly proved. Hence, in Eq. (4) we regard x, ω^l, and ω^L as functions of ω_k, and denote them as $x(\omega_k)$, $\omega^l(\omega_k)$, and $\omega^L(\omega_k)$, respectively. Since $x(\omega_k)$ and $\omega^l(\omega_k)$ are both strictly increasing with ω_k, $\omega^L(\omega_k)$ is increasing with ω_k. Consider any pair of $\omega_{k,1}$ and $\omega_{k,2}$ such that $\omega^{\min} \leq \omega_{k,1} < \omega_{k,2} \leq \omega^{\max}$. Due to the monotonicity of $\omega^L(\omega_k)$ with respect to ω_k, we only need to discuss three cases.

`Case a`: $\omega^L(\omega_{k,1}) = \omega^L(\omega_{k,2}) = \omega^{\min}$. In this case, we have

$$\begin{aligned}&T^{\max}(\omega_{k,2},\omega_{k+1}) - T^{\max}(\omega_{k,1},\omega_{k+1})\\ =&\frac{\omega_{k,2}-\omega_{k,1}}{-\alpha^{\min}} - \frac{\omega_{k,2}^2-\omega_{k,1}^2}{-2\omega^{\min}\alpha^{\min}}\\ =&\frac{\omega_{k,2}-\omega_{k,1}}{-\alpha^{\min}}\cdot(1-\frac{\omega_{k,2}+\omega_{k,1}}{2\omega^{\min}})\\ <&\ 0\end{aligned}$$

`Case b`: $\omega^{\min} < \omega^L(\omega_{k,1}) < \omega^L(\omega_{k,2})$. Thus, $\forall\omega_k \in [\omega_{k,1},\omega_{k,2}]$, we have $\omega^L(\omega_k) = \omega^l(\omega_k)$. Also, $\omega^l(\omega_k)$ is a differentiable function of ω_k for any $\omega_k \in [\omega_{k,1},\omega_{k,2}]$. Since $(\alpha^{\max}-\alpha^{\min})(\omega^l(\omega_k))^2 = x(\omega_k)$, it is

$$\frac{\partial\omega^l}{\partial\omega_k} = \frac{1}{2\omega^l(\alpha^{\max}-\alpha^{\min})}\cdot\frac{\partial x}{\partial\omega_k} = \frac{\alpha^{\max}\omega_k}{\omega^l(\alpha^{\max}-\alpha^{\min})}$$

Hence, $\forall\omega_k \in [\omega_{k,1},\omega_{k,2}]$,

$$\begin{aligned}&\frac{\partial}{\partial\omega_k}T^{\max}(\omega_k,\omega_{k+1})\\ =&\frac{1}{-\alpha^{\min}} - \frac{\omega_k}{-\alpha^{\min}\omega^{\min}}\\ &+\frac{\partial\omega^l}{\partial\omega_k}\cdot(-\frac{1}{-\alpha^{\min}}+\frac{\omega^l}{-\alpha^{\min}\omega^{\min}}+\frac{\omega^l}{\alpha^{\max}\omega^{\min}}-\frac{1}{\alpha^{\max}})\\ =&\frac{\omega^l-\omega_k}{-\alpha^{\min}\omega^l}\\ <&\ 0\end{aligned} \tag{24}$$

Note the last inequality holds with the intuition that ω^l is achieved by decelerating from ω_k with $|\alpha^{\min}|$. (24) implies that $T^{\max}(\omega_{k,2},\omega_{k+1}) < T^{\max}(\omega_{k,1},\omega_{k+1})$ as $\omega_{k,2} > \omega_{k,1}$.

`Case c`: $\omega^L(\omega_{k,1}) = \omega^{\min} < \omega^L(\omega_{k,2})$. We note that the function $\omega^l(\omega_k)$ is continuous and monotonically increasing with ω_k. Also, $\omega^l(\omega_{k,1}) \leq \omega^{\min}$, $\omega^l(\omega_{k,2}) > \omega^{\min}$. Hence, there must exist $\omega_{k,*}$ such that $\omega_{k,1} \leq \omega_{k,*} < \omega_{k,2}$ and $\omega^l(\omega_{k,*}) = \omega^{\min}$. From the proof of `Case b`, it is $T^{\max}(\omega_{k,2},\omega_{k+1}) < T^{\max}(\omega_{k,*},\omega_{k+1})$. From the proof of `Case a`, we have $T^{\max}(\omega_{k,*},\omega_{k+1}) \leq T^{\max}(\omega_{k,1},\omega_{k+1})$. Hence, $T^{\max}(\omega_{k,2},\omega_{k+1}) < T^{\max}(\omega_{k,1},\omega_{k+1})$.

Combining the above three cases, it is always true that $T^{\max}(\omega_k,\omega_{k+1})$ is strictly decreasing with ω_k. □