Trip-Based Performance Optimization of Engine Control Tasks with Dynamic Adaptation

ABSTRACT

The different control strategy implementations integrated in the engine control tasks have the different impact on the overall performance and system workload. Hence, the design problem is how to optimize the performance by selecting the transition speeds under schedulability constraint. The previous work focus on the static performance optimization which adopts the unchanged transition speed setting during the whole driving process and hence cannot adjust to the actual situations (e.g., driving cycles). In this work, we define the trip-based performance optimization problem and then propose one optimization framework using the concept of engine control tasks with dynamic adaptation. In order to reduce the optimization runtime, we design an efficient and sufficient four-step schedulability test. The experimental results demonstrate that our approach can obtain the performance close to the upper bound and has the potential as an online algorithm due to the low runtime.

1. INTRODUCTION

The design of engine management systems (EMS) is a challenging example of Cyber-Physical Systems (CPS) design problems and plays a key role in car manufacturers. In order to determine the injection time and amount of fuel, the physical characteristics of the engine has to be considered in the EMS design. This leads to the existence of some engine control applications which are triggered at predefined angles of the engine crankshaft. As a result, the release frequencies of these applications will vary with respect to the engine speed.

As highlighted in the keynote [16], today's EMS integrates a (finite) set of operation modes and control strategies with different amounts of computational demand. In general, the computational demand is quantified by the *worst case execution time (WCET)*. The more advanced control strategy will result in the better performance at the price of the larger computational demand. For example, compared with the simple single-injection strategy, the strategy based on multiple fuel injections can reduce the emission or improve the fuel savings but needs more CPU occupation time. To cope with the CPU overload on the hosting Electronic Control Unit (ECU) at high engine rotation speeds ω (in RPM, Revolutions per Minutes), the EMS will adaptively adopt the more simplified control strategies at the higher engine speeds. Hence, the EMS will execute different modes for different speed intervals. For this reason, angular tasks are often referred to as adaptive variable-rate (AVR) tasks in the literature [19].

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235


```
#define ω4 1000
#define ω3 2000
#define ω2 4000
#define ω1 6000

task EC_task {
      ω = read_engine_speed();
      f0();
      if (ω ≤ ω1)      f1();
      if (ω ≤ ω2)      f2();
      if (ω ≤ ω3)      f3();
      if (ω ≤ ω4)      f4();
}
```

Figure 1: Typical software implementation of vehicle engine control.

The engine control software is typically realized as a sequence of conditional `if` statements [16, 11]. As in Fig. 1, the control strategy at speeds higher than ω_1 (i.e., $\omega > \omega_1$) only executes function `f0()`. This is only a small subset of the functions (`f0()`, `f1()`, `f2()`, `f3()`, and `f4()`) executed by the control strategy at ω_4. Hence, the WCET is dependent on the engine rotation speed.

Generally, in the current EMS design, the engine configuration is determined by conducting a series of experiments at the test bench or collecting the calibration data from the actual driving over standard driving cycle, and then fixed at runtime. Clearly, the fixed configuration may lead to the sub-optimal since the environmental differences. In this paper, we focus on how to optimize the control performance based on the more realistic cases. We first review the related work below.

1.1 Related Work

Kim et al. [24] propose the rhythmic task model under a few of restrictive assumptions to allow continually varying the periods and WCETs according to the current physical attributes of the engine system. Pollex et al. [30] present a sufficient real-time analysis assuming that the angular velocity is arbitrary but fixed. And later in [29] they extend the analysis with the changing of the engine speed. [30, 29] both perform the sufficient test by deriving the upper bound on the interference with the maximum execution time and minimum inter-arrival time for certain intervals. To further improve

the accuracy, Feld and Slomka [20] provide an analysis with the consideration of dependencies between these angular tasks.

The AVR task model was first introduced by Buttle in his keynote [16] address at 2012. This model is characterized by a set of the finite execution modes with the different computational demand (see Fig 1). With respect to the fixed priority, Davis et al. [18] first present a number of sufficient analysis techniques to upper bound the worst case interference for AVR tasks. Biondi et al. [11] introduce the concept of dominant speeds to efficiently capture the interference of angular tasks, and subsequently [12, 10] derive exact response time analysis. Feld and Sloma [21] further improve the schedulability analysis [11, 12, 10] by omitting some searching directions, which is exact if the maximum acceleration and deceleration have the same absolute values.

In the recent work [13, 14], Biondi et al. maximize the performance of the task set comprising of a few of periodic tasks and multiple AVR tasks triggered by the common rotation source scheduling with the fixed priority by selecting the transition speed for each mode. The engine performance is assumed to be related to the performance function and the length of the speed interval for each mode.

With respect to EDF scheduled systems, a number of sufficient utilization-based schedulability tests are presented [15, 22, 7]. Differently, Biondi et al. [9, 6] propose an exact schedulability analysis based on the concept of dominant speed as in [11]. Mohaqeqi et al. [26] propose to partition the speed space and transform angular tasks to digraph real-time (DRT) tasks [31].

1.2 Our Contributions and Structure

Overall, all the previous work assumes that the engine switching speeds are fixed offline. Differently, in our previous work [2], we propose the concept of engine control with dynamic adaptation, where the engine switching speeds are dynamically adjusted according to the driving cycle, and present the scheduability analysis for such systems. In this paper, we will focus on the optimization design adopting the dAVR task model for the purpose of the high performance and low runtime. This paper has the following contributions:

- We formalize the trip-based performance optimization problem and present an optimization framework, which greedily assigns the optimal configuration for the upcoming speed profile while guaranteeing the schedulability.
- We propose one sufficient but efficient mixed schedulability test by integrating a set of the analysis methods with different accuracies.
- We conduct a number of experiments to evaluate the benefits of the proposed approach in terms of engine control performance and optimization runtime.

The rest of the paper is organized as follows. Section 2 describes the system model considered in this work and some existing results for the engine dynamical characteristics. Section 3 defines the trip-based performance optimization problem and presents a simple optimization framework. Section 4 designs an efficient schedulability test combining some existed schedulability tests with different precisions and complexities, and presents the methods by using the predefined precision length for reducing the search space. Section 5 evaluates the benefits of our approach. Finally, Section 6 concludes the paper.

2. SYSTEM MODEL

We now present the system model, following the study on AVR tasks (e.g., [11, 12, 26]). The engine, i.e., the rotation source, is described by its current rotation angle θ, angular velocity ω, and angular acceleration α. Due to the engine physical attributes, the angular velocity 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 an engine control system Γ consisting of a set of real-time tasks scheduled with *fixed priority* on a uni-processor. Each task is either a periodic task 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), $D_i \leq T_i$ is the constrained deadline, and P_i is the priority. These parameters are all fixed, i.e., they are independent from the engine rotation.

An AVR task τ_i^* models a task triggered at predefined engine crankshaft angles $\theta_i = \Psi_i + k\Theta_i, \quad k \in \mathbb{N}$, where $\mathbb{N}$ is the set of non-negative integers, Ψ_i is the angular phase, and Θ_i is the angular period. The angular deadline is $\Delta_i = \lambda_i \cdot \Theta_i$ where $\lambda_i \in [0, 1]$ (hence τ_i^* also has a constrained deadline). Similar to [12], we assume the AVR tasks *share a common rotation source, and they have the same angular period and phase*. Hence, the angular period and phase are denoted as Θ and Ψ, i.e., the task index is dropped for them. The AVR task parameters (WCET, inter-release time, deadline) all depend on the engine dynamics.

Engine Dynamics. We assume that *instantaneous angular speed at the time of the job release can be used* [11, 12, 26]. The speed after rotating θ angles given an initial speed ω and a constant acceleration α is calculated as [11]

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

Consider two engine 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 $\omega_k \rightsquigarrow \omega_{k+1}$. ω_k and ω_{k+1} shall satisfy

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

The minimum inter-release time between them, denoted as $T^{\min}(\omega_k, \omega_{k+1})$, is given as [26]

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

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 that $t_2^u = 0$ 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 con-

stant speed, and a maximum acceleration

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

For convenience, we also use the following simplified notations

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

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

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

In addition, the minimum inter-release time between an AVR job (σ, ω) and its subsequent job is calculated as:

$$\begin{aligned} & \widehat{T}(\omega) = t_1^r + t_2^r, \\ \text{where} \quad & t_1^r = \tfrac{\omega^R - \omega}{\alpha^{\max}}, \\ & t_2^r = \tfrac{1}{\omega^{\max}}(\Theta - \tfrac{(\omega^R)^2 - \omega^2}{2\alpha^{\max}}), \\ & \omega^R = \min\{\omega^{\max}, \Omega(\omega, \alpha^{\max}, \lambda_i\Theta)\} \end{aligned} \tag{7}$$

Obviously, it always holds that $\forall \omega \in [\omega^{\min}, \omega^{\max}], \widehat{T}(\omega) \leq T^{\min}(\omega)$.

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

$$\begin{aligned} & D_i(\omega) = t_1^d + t_2^d, \\ \text{where} \quad & t_1^d = \tfrac{\omega^D - \omega}{\alpha^{\max}}, \\ & t_2^d = \tfrac{1}{\omega^{\max}}(\lambda_i\Theta - \tfrac{(\omega^D)^2 - \omega^2}{2\alpha^{\max}}), \\ & \omega^D = \min\{\omega^{\max}, \Omega(\omega, \alpha^{\max}, \lambda_i\Theta)\} \end{aligned} \tag{8}$$

sAVR Task Model. The **static AVR** (or **sAVR**) task model, as proposed in the literature, assumes a fixed engine configuration including the engine 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 engine 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

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

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

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

The **steady-state** utilization for a fixed engine speed ω is defined as [8]:

$$u_i(\omega) = \frac{\mathcal{SC}_i(\omega)\Theta}{\omega} \tag{10}$$

Similarly, the utilization for one fixed execution mode $m \leq SM_i$, i.e., there does not exist any mode switch, is defined as:

$$U_{i,m} = \frac{SC_i^m}{T^{\min}(\varsigma\omega_i^m)} \tag{11}$$

Time Interval (ms)						
$[\gamma_1, \gamma_2) = [0, 100)$	$\mathcal{M}_{i,1}$	m-th mode	1	2	3	
		$\omega_{i,1}^m$ (rpm)	500	2500	4500	
		$C_{i,1}^m$ (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)	500	1500	2500	4500
		$C_{i,3}^m$ (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)	500	3500		
		$C_{i,3}^m$ (μs)	600	300		

Table 1: An illustrative example of a dAVR task τ_i^* (rpm = revolutions per minute).

In addition, let U_i denote the utilization of the sAVR task τ_i^*. Obviously, we have the following relationship:

$$\max_{\omega \in [\omega^{\min}, \omega^{\max}]} u_i(\omega) \leq \max_{m=1,\ldots,SM_i} U_{i,m} \leq U_i \tag{12}$$

dAVR Task Model. In this paper, like [2] we use the concept of engine control with dynamic execution modes, where the switching speeds are adjusted at runtime. The static AVR task model can also be specified by the dynamic dAVR task (dAVR) model [2]. We assume that the reconfiguration happens at predefined times $\mathcal{T} = \{\gamma_1, \cdots, \gamma_T\}$. The associated dAVR task τ_i^* has a series of execution mode sets defined as

$$\begin{aligned} & \mathcal{Q}_i = \{(\mathcal{M}_{i,k}, \gamma_k), k = 1, \ldots, T\}, \\ \text{where} \quad & \mathcal{M}_{i,k} = \{(C_{i,k}^m, \omega_{i,k}^m), m = 1, \ldots, M_{i,k}\} \end{aligned} \tag{13}$$

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

$$\mathcal{C}_i(t, \omega) = C_{i,k}^m \quad \text{if } t \in [\gamma_k, \gamma_{k+1}) \textbf{ and } \omega \in [\omega_{i,k}^m, \omega_{i,k}^{m+1}) \tag{14}$$

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 1 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.*

3. PROBLEM DEFINITION AND OPTIMIZATION FRAMEWORK

3.1 Problem Definition

In this paper, we discuss how to maximize the engine performance with respect to the engine speed profile by dynamically selecting the control strategies and the corresponding switching speeds while guaranteeing the schedulability. Similar to [13, 14], the optimization problem is restricted to one real-time task system consisting of a set of periodic tasks and multiple engine control implementations, which will be integrated into one single AVR task. These systems under analysis are scheduled with the fixed priority running on a uni-processor. The optimization problem is formalized as follows.

Inputs: a set of real-time tasks consisting of periodic tasks, an unset AVR task integrated with multiple engine control implementations, and the priority ordering, as well as one engine speed profile.

There are multiple control implementations with different computational load, which will be appropriately integrated into the AVR task. Similar to [13, 14], we assume the implementation with the larger WCET has the better performance, and consider two types of engine control performance rate functions. Such functions may be

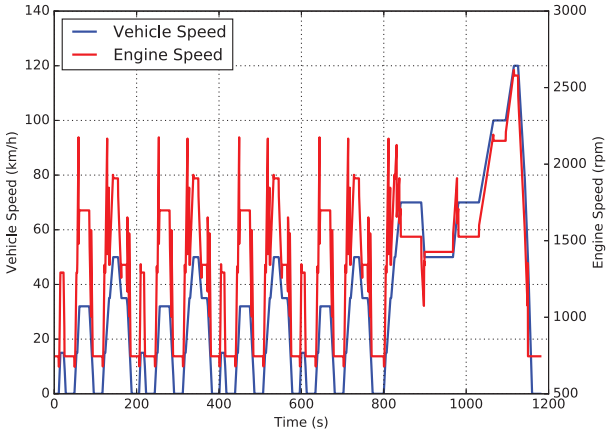

Figure 2: Vehicle and engine speeds in the new European driving cycle (NEDC).

used to approximate a set of engine control performance metrics, such as power, fuel consumption, and emissions [13]:

- The first type of performance function is a constant function that is independent from the engine speed $\mathbf{P}_c(\omega) = j$, where j is a constant.
- The second is an exponential function of the engine speed $\mathbf{P}_e(\omega) = k_1 \cdot e^{-\frac{k_2}{\omega}}$, where k_1 and k_2 are two constant. coefficients.

In this paper, the larger the function is, the better the engine control performance.

Without loss of generality, we denote the engine speed profile as $\mathcal{W}$ and use function $\mathcal{W}(t)$ to represent the engine speed at time t. Following the concept of the driving cycles (vehicle speed with respect to time), we assume function $\mathcal{W}(t)$ can also be expressed by a set of sampling times, i.e., $\{t_1, \ldots, t_W\}$. And during each sampling period, the engine speed is fixed, i.e.,

$$\forall t, t' \in [t_k, t_{k+1}), \;\; \mathcal{W}(t) = \mathcal{W}(t')$$

Hence, we can use the simplified notation $\mathcal{W}_k = \mathcal{W}(t_k)$ for referring to $\mathcal{W}(t)$, $\forall t \in [t_k, t_{k+1})$. Moreover, let t_{W+1} be the finish time of this profile. As an example, Figure 2 illustrates the NEDC driving cycle [4] and the corresponding engine speed profile obtained by the commercial vehicle simulator AVL Cruise [3].

Constraint: the reconfiguration times can only happen at t_k ($k = 2, \ldots, W$) and the system must satisfy the time constraint. The task system is schedulable if and only if any job instance released by the task system cannot miss the deadline with the predefined fixed priority ordering.

Variables: the concrete engine configuration for each time interval. That is to say, the optimization key is how to appropriately design the unset AVR task consisting of multiple control strategies.

Performance Metric: given one engine speed profile $\mathcal{W}$ and one concrete AVR task τ_A^*, the overall performance metric can be expressed by

$$\begin{aligned} \mathcal{P}_A &= \int_{t_1}^{t_{W+1}} \mathcal{P}_A(t, \mathcal{W}(t)) dt \\ &= \sum_{k=1}^{W} \mathcal{P}_A(t_k, \mathcal{W}_k) \cdot (t_{k+1} - t_k) \end{aligned} \tag{15}$$

where $\mathcal{P}_A(t, \omega)$ denotes the performance rate of the control implementation adopted at release time t and speed ω from τ_A^*. Actually, given the release time t and speed ω, we can determine which implementation is chosen, then the corresponding performance function ($\mathbf{P}_c(\omega)$ or $\mathbf{P}_e(\omega)$), and finally the performance rate with respect to this speed. The overall performance is the integration of the performance rate over the time. Such overall performance also equals to the summation of the product of the performance rate and its time duration (or the sample period).

Optimization Objective: the maximum engine control performance, i.e., $\max_{\tau_A^*} \mathcal{P}_A$.

Outputs: the concrete parameter setting for τ_A^*.

Algorithm 1 Pseudo-code for the optimization procedure.

```
1: procedure OPTIMIZATION(C_1, ..., C_Q, W)
2:     for k ← 1 to W do
3:         for m ← 1 to Q do
4:             M_k ← {(C_m, W_k), (C_Q, ω^max)};
5:             if SCHEDULABLE({(M_j, t_j), ∀j ≤ k}) then
6:                 break;
7:             if m = Q then
8:                 return Unfeasible;
9:     return {(M_j, t_j), ∀j ≤ W};
```

3.2 Optimization Framework

In this subsection, we propose an optimization framework for the problem mentioned above. Intuitively, thanks to the concept of the dAVR task model which allows to dynamically adjust the engine configurations, it is possible to greedily assign the best configuration for each engine speed without the consideration of the following engine speeds. The detailed optimization procedure is illustrated in Algorithm 1.

The algorithm will search over the whole profile and design the engine configuration for each sampling speed (Lines 2-8). The basic idea is to select the optimal implementation for speed $\mathcal{W}_k$, i.e, $\max \mathcal{P}_A(t_k, \mathcal{W}_k)$. The iteration in Lines 3-8 models this progress and will try to construct one engine configuration such that speed $\mathcal{W}_k$ can correspond to the highest performance function. At this point, in order to introduce the smallest workload, we construct one simple two-mode set composed of the m-th implementation and the smallest-WCET one and switching modes at speed $\mathcal{W}_k$ (Line 4). The schedulability can be verified by function **Schedulable** which is capable of entailing the schedulability of the system with the current parameter setting (Line 5). Moreover, if the condition at Line 7 is satisfied, this means there is no feasible solution such that the system is schedulable and will returns the unfeasible result (Line 8).

Properties: Now we discuss a few of key properties for the framework. Firstly, each solution returned by this framework (Line 9) is feasible since function **Schedulable** is able to provide the sufficient schedulability test. Secondly, the framework must obtain the feasible solution if the system only configured with the least-WCET implementation is schedulable (i.e., the AVR task containing one single execution mode with the simplest control strategy can lead to a feasible solution). Thirdly, the framework assigns the switching speeds according to the engine speed profile while the state-of-the-art approaches in [13, 14] examine them based on heuristic/constant searching steps. Finally, each engine configuration within the time interval $[t_k, t_{k+1})$ must contain the simplest implementation and the mode number cannot exceed two.

4. DESIGNING EFFICIENT SCHEDULABLITY TEST

In this section, we first briefly recap the results of the analysis method for the dAVR task model (which is generalized from the sAVR task model), and then propose an efficient schedulability test. The schedulability analysis of periodic tasks interfered by the dAVR task is considered in this work, since it is the main reason for the runtime complexity. With respect to the schedulability analysis of the dAVR task, please refer Section 4 in [2].

Our previous work [2] provides the sufficient schedulability analysis of the dAVR task model by explicitly considering the engine reconfigurations. To achieve this, we note that the key is to derive the valid (finite) speed partitions for dAVR tasks [12, 10, 26, 2], which are defined as follows [2].

DEFINITION 1 (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 partitions 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 is 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., $\mathcal{B} = \{\beta_0, \beta_1, \cdots, \beta_B\}$.

Given a valid speed partition $\mathcal{B}$, based on the transformation method proposed in [2], we can construct one dynamic digraph real-time task and then drive the sufficient schedulability analysis.

DEFINITION 2 (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 $p^{\min}(v_i, v_j)$ and $p^{\max}(v_i, v_j)$ respectively denote the minimum and maximum separation times between the releases of v_i and v_j.*

Now the schedulability analysis can be performed with the transformed dDRT task.

DEFINITION 3 (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 $\forall l < n$,*

$$(\nu_l, \nu_{l+1}) \in \mathbb{E} \textbf{ and } p^{\min}(\nu_l, \nu_{l+1}) \leq \pi_{l+1} - \pi_l \leq p^{\max}(\nu_l, \nu_{l+1}) \tag{16}$$

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

DEFINITION 4 (INTERFERENCE FUNCTION OF $\mathcal{D}$). *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 $\mathcal{D}.I(t)$, calculated by*

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

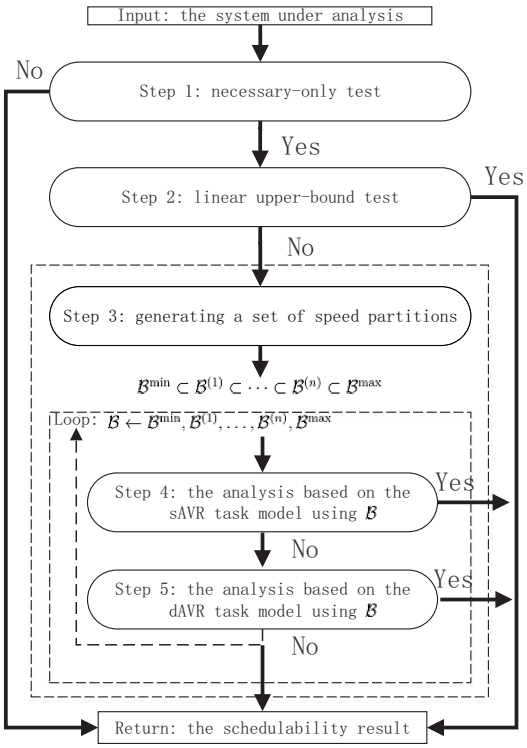

Figure 3: The efficient mixed schedulability test.

With these two definitions, 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 [2]

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

$$\text{where } R(\tau_i, \mathcal{D}) = \min_{t>0} \left\{ t \mid C_i + \sum_{\tau_j \in hp(i)} \left\lceil \frac{t}{T_j} \right\rceil C_j + \mathcal{D}.I(t) \leq t \right\} \tag{17}$$

Further, the analysis based on this equation can be accelerated by searching over the common-WCET dDRT job sequence sets and finding the critical job sequences [2].

In order to further reduce the complexity, we present an approach for efficiently verifying the schedulability of the system with the dAVR task τ_A^* setting as $\{(\mathcal{M}_j, t_j), \forall j \leq k\}$, which will be used as function **Schedulable** in the framework. Our schedulability test is designed by combining a set of analysis methods with different precisions and summarized in Fig. 3.

4.1 Step 1: necessary-only test

First of all, our hybrid schedulability test will perform one **necessary-only** analysis for one sAVR task with the execution mode set $\mathcal{M}_k$. **Note that for the dAVR task under analysis $\mathcal{M}_k$ will be preserved from t_k to $+\infty$.** Based on this observation, if the system with the sAVR task is unschedulable, this means the system with the dAVR task is unschedulable. To efficiently capture the unschedulability, we model each mode of the sAVR task as one periodic task without the consideration of dynamic conditions (similar to the UB-NX test proposed in [18]) and these transformed periodic tasks inherit the priority of the original dAVR task. The worst-case execution time of each periodic task equals to the execution time of the original mode and the period is set as the minimum inter-release time such that any two consecutive jobs are released in this mode.

4.2 Step 2: linear upper-bound test

Due to the nature of the necessary-only analysis, the schedulability verified by **Step 2** cannot imply that the system is actually schedulable. Hence, we present one **linear upper-bound** analysis to entail the schedulability. Intuitively, we can approximate a dAVR task τ_A^* with an sAVR task $\tau_{\widetilde{A}}^*$ where the WCET of $\tau_{\widetilde{A}}^*$ released at speed ω is set as the **maximum** WCET of τ_A^* released at speed ω over all the engine configurations which can contrib to the schedulability. Now let us discuss how to derive the schedulability analysis of periodic task τ_i interfered the periodic tasks with higher priorities and this dAVR task τ_A^*. Since $\{(\mathcal{M}_j, t_j), \forall j \leq k-1\}$ has been verified to be schedulable and the engine configuration may be changed at time t_k, this means that any job of τ_i released within the interval $[\max(0, t_k - D_i), +\infty)$ must be schedulable. At this point, the involved engine configurations are restricted within this interval and then abstracted as the sAVR task $\tau_{\widetilde{A}}^*$.

Given the resultant sAVR task, we can apply a set of analysis methods proposed in the prior work. Among these methods, Davis et al. [18] present an upper bound on the maximum amount of workload of the sAVR task, formalized as:

$$\tau_{\widetilde{A}}^*.I^{UB}(t) = U_{\widetilde{A}} \cdot t + C_{\widetilde{A}}^{\max}(1 - U_{\widetilde{A}}) \tag{18}$$

where $U_{\widetilde{A}}$ and $C_{\widetilde{A}}^{\max}$ denote as the maximum utilization and execution time of $\tau_{\widetilde{A}}^*$ respectively.

However, [8] demonstrates that calculating the utilization $U_{\widetilde{A}}$ is difficult and then provides a utilization bound as:

$$U'_{\widetilde{A}} = \max_m \frac{SC_{\widetilde{A}}^m}{\widehat{T}(\varsigma\omega_{\widetilde{A}}^m)} \tag{19}$$

In order to provide a sufficient-only analysis with the linear-time complexity, we further upper bound the maximum amount of the interference of the sAVR task in the following equation:

$$\tau_{\widetilde{A}}^*.I^{LUB}(t) = U'_{\widetilde{A}} \cdot t + C_{\widetilde{A}}^{\max}(1 - \max_{m=1,\ldots,SM_{\widetilde{A}}} U_{\widetilde{A},m}) \tag{20}$$

It can be easily shown that $\forall t \geq 0, \tau_A^*.I^{UB}(t) \leq \tau_A^*.I^{LUB}(t)$.

By this equation, the worst case response time of the periodic task τ_i interfered by $\tau_{\widetilde{A}}^*$ can be safely bounded by the following equation:

$$R^{LUB}(\tau_i, \tau_{\widetilde{A}}^*) = \min_{t>0} \left\{ t \mid C_i + \sum_{\tau_j \in hp(i)} \left\lceil \frac{t}{T_j} \right\rceil C_j + \tau_{\widetilde{A}}^*.I^{LUB}(t) \leq t \right\} \tag{21}$$

where $hp(i)$ denotes the set of periodic tasks with higher priority than τ_i. If $R^{LUB}(\tau_i, \tau_{\widetilde{A}}^*) \leq D_i$, this means τ_i is schedulable.

4.3 Step 3: generating speed partitions

However, the experimental results in [18] demonstrates the accuracy of the analysis based on Equation (21) deteriorates with the utilization increase. Motivated by the observation, in order to obtain the higher overall performance we adopt the analysis techniques presented in [12, 10, 26, 2].

We note that by the second condition in Definition 1, the smallest valid speed partition for τ_A^* is composed of all the switching speeds and the two speed limits, i.e.,

$$\mathcal{B}^{\min} = \{\omega_{A,k}^m \mid \forall k = 1, \cdots, T, \forall m = 1, \cdots, M_{A,k}\} \cup \{\omega^{\min}, \omega^{\max}\} \tag{22}$$

Thus, the smallest valid speed partition is $\mathcal{B}^{\min} = \{500, 1500, 2500, 3500, 4500, 6500\}$ (rpm) for the dAVR task in Table 1.

In addition, from Theorem 2 in [2], it is typically impossible to find an exact task transformation (and an exact speed partition). At this point, let $\mathcal{B}^{\max}$ denote the finest valid speed partition, which is generated in the similar way as the speed partitions in [11, 26] but the involved switching speeds are all the elements in $\mathcal{B}^{\min}$. Each valid speed partition $\mathcal{B}^{(i)}$ is generated such that $\mathcal{B}^{(i-1)} \subset \mathcal{B}^{(i)}$ ($\mathcal{B}^{(0)} = \mathcal{B}^{\min}$) and $\mathcal{B}^{(i)}$ includes at least half of elements in $\mathcal{B}^{\max} - \mathcal{B}^{(i-1)}$.

Each valid speed partition $\widetilde{\mathcal{B}}$ shall be generated such that satisfies $\mathcal{B} \subset \widetilde{\mathcal{B}}$ and includes nearly half of elements in $\{\beta_{i,1}, \ldots, \beta_{i,b_i}\}$.

4.4 Step 4: sAVR-based analysis using $\mathcal{B}$

In this step, we still use the sAVR task $\tau_{\widetilde{A}}^*$ constructed in the same way as **Step 3** to verify the schedulability. Thus, given the speed partition $\mathcal{B}$, the analysis methods for the sAVR task model presented in [12, 10, 26] can be applied.

We note that the number of the considered critical sAVR job sequences in the above analyses significantly increases with the length of the time window. To reduce the complexity, we define an auxiliary function $\mathcal{A}.\widehat{I}(t, \Delta)$ for a job sequence $\mathcal{A} \in \tau_{\widetilde{A}}^*$ with one predefined precision length $\Delta \geq 0$ as:

$$\mathcal{A}.\widehat{I}(t, \Delta) = \begin{cases} \mathcal{A}.I(t), & \text{for } t \leq \Delta \\ \mathcal{A}.I(\Delta) + \tau_{\widetilde{A}}^*.I^{LUB}(t - \Delta), & \text{for } t > \Delta \end{cases} \tag{23}$$

By this definition, it is easy to verify that $\forall t \geq 0, \mathcal{A}.I(t) \leq \mathcal{A}.\widehat{I}(t, \Delta)$. Hence, the WCRT of the periodic task τ_i interfered by $\tau_{\widetilde{A}}^*$ can be safely bounded by:

$$\widehat{R}(\tau_i, \tau_{\widetilde{A}}^*, \Delta) = \max_{\mathcal{A} \in \mathcal{CS}(\Delta)} \widehat{R}(\tau_i, \mathcal{A}, \Delta) \tag{24}$$

where

$$\widehat{R}(\tau_i, \mathcal{A}, \Delta) = \min_{t>0} \left\{ t \mid C_i + \sum_{\tau_j \in hp(i)} \left\lceil \frac{t}{T_j} \right\rceil C_j + \mathcal{A}.\widehat{I}(t, \Delta) \leq t \right\} \tag{25}$$

And $\mathcal{CS}(\Delta)$ denotes the set of critical job sequences of $\tau_{\widetilde{A}}^*$ within the time length Δ. As a result, the number of the considered job sequences is restricted in this equation.

4.5 Step 5: dAVR-based analysis using $\mathcal{B}$

Given the speed partition $\mathcal{B}$, the analysis methods for the dAVR task model proposed in [2] can be used. However, we have to figure out the complexity with the large time windows. Similar to **Step 4**, for reducing the number of the considered dDRT job sequences, we define an auxiliary function $\mathcal{D}.\widehat{I}(t, \Delta)$ for a job sequence $\mathcal{D} \in \tau_D^*$ ($\Delta \geq 0$) as:

$$\mathcal{D}.\widehat{I}(t, \Delta) = \begin{cases} \mathcal{D}.I(t), & \text{for } t \leq \Delta \\ \mathcal{D}.I(\Delta) + \tau_{\widetilde{A}}^*.I^{LUB}(t - \Delta), & \text{for } t > \Delta \end{cases} \tag{26}$$

where $\tau_{\widetilde{A}}^*$ is constructed in the same way as **Step 4**.

By this definition, we can easily verify that $\forall t \geq 0, \mathcal{D}.I(t) \leq \mathcal{D}.\widehat{I}(t, \Delta)$. Hence, the WCRT of the periodic task τ_i interfered by τ_D^* can be safely bounded by:

$$\widehat{R}(\tau_i, \tau_D^*, \Delta) = \max_{\mathcal{D} \in \tau_D^*} \widehat{R}(\tau_i, \mathcal{D}, \Delta) \tag{27}$$

where

$$\widehat{R}(\tau_i, \mathcal{D}, \Delta) = \min_{t>0} \left\{ t \mid C_i + \sum_{\tau_j \in hp(i)} \left\lceil \frac{t}{T_j} \right\rceil C_j + \mathcal{D}.\widehat{I}(t, \Delta) \leq t \right\} \tag{28}$$

Consequently, it is sufficient to calculate $\widehat{R}(\tau_i, \mathcal{D}, \Delta)$ over a bounded number of (critical) dDRT job sequences. The complexity is significantly reduced by safely ignoring any common-WCET dDRT sequence sets whose critical dDRT job sequences have been dominated by other dDRT job sequences.

5. EXPERIMENTAL EVALUATION

In this section, we evaluate the benefits of the proposed optimization framework on engine control performance, using randomly generated synthetic task systems. We adopt the engine parameters in [13, 14]: (i) the minimum/maximum engine speed is 500/6500 rpm; and (ii) the maximum acceleration/deceleration is 1.62×10^{-4}rev/ms Hence, the engine needs 35 revolutions to accelerate/decelerate between the minimum and maximum speeds. We compare a list of task models and optimization algorithms as follows:

- dAVR-exact: An exact optimization algorithm [27], which will dynamically adjust the switching speeds to maximize the control performance for each data point in the engine speed profiles, while using dAVR for schedulability analysis.
- dAVR2sAVR-exact: Similar to dAVR-exact but using dAVR2sAVR for schedulability analysis.
- sAVR-heuristic: The backward search optimization algorithm from [13, 14]. The schedulability analysis is from [12, 10], which is exact for sAVR tasks. The original algorithm [13, 14] simply returns failure when a calculated switching speed is less than the minimum engine speed. In such cases, we set the performance to be zero.
- sAVR-heuristic-improved: A small improvement over sAVR-heuristic. Specifically, when the original algorithm [13, 14] fails to find a set of schedulable switching speeds, we modify the algorithm to use a single execution mode that has the best performance while keeping the system schedulable.
- PERF-UB: The performance upper-bound [13], achieved with the necessary-only analysis UB as explained in [2]. Specifically, the switching speed of an execution mode is calculated as the largest speed that maintains the system schedulability assuming there are only two execution modes: the mode under consideration and the simplest one.

We note that there is another optimization algorithm (plain branch-and-bound) for sAVR task model in [13, 14]. We compare with sAVR-heuristic only since (i) it is very close to branch-and-bound in the optimized performance; (ii) branch-and-bound is reported to take a long time (on average 10 minutes on an Intel i7 3.2GHz processor), which is ill-suited for online adjustment of engine switching speeds (or even the extensive experiments below).

Engine Speed Profiles. In our experiments, we use ten standard driving cycles to evaluate the performance: IDC, NEDC, FTP-75, HWFET, ECE 15, JA 10-15, EUDC, US SC03, ARTEMIS ROAD, and ARTEMIS URBAN. IDC is available from [1], while the rest is from [4]. Given a driving cycle (the vehicle speed over time), we use the commercial vehicle simulator AVL Cruise [3] to get the corresponding engine speed profile.

Engine Control Performance Functions. As in [13], we consider two types of engine control performance functions. Such functions may be used to approximate a set of engine control performance metrics, such as power, fuel consumption, and emissions [13]. In this paper, the larger the function is, the better the engine control performance.

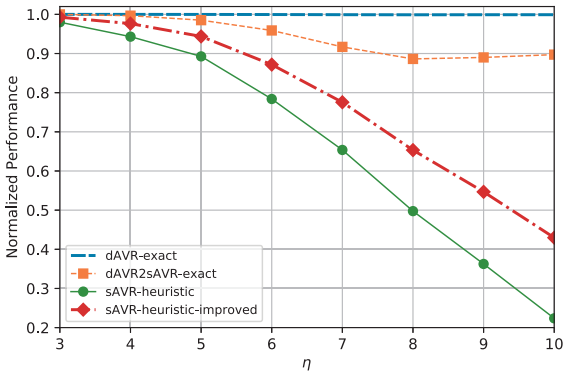

Figure 4: Normalized performance vs. scaling factor η for constant performance functions.

The first type of performance function is a constant function that is independent from the engine speed $\mathbf{P}_c(\omega) = j$, where j is a constant. For each control strategy, j is selected uniformly from $[j^{\min}, j^{\max}]$ with a minimum separation of 1.

The second is an exponential function of the engine speed $\mathbf{P}_e(\omega) = k_1 \cdot e^{-\frac{k_2}{\omega}}$, where k_1 and k_2 are two constant coefficients. k_1 is always set to 1, while k_2 is generated as follows: the seed value $k_{2,sed}$ of k_2 is randomly selected following a uniform distribution $[\log k_2^{\min}, \log k_2^{\max}]$, and the actual performance coefficient k_2 is computed as $k_2 = e^{k_{2,sed}}$.

5.1 Small-Size Task Systems

We generate the random task systems following [13]. Specifically, each task system contains five periodic tasks and an AVR task. The task priority order is assigned with the deadline monotonic policy. The total utilization of periodic tasks is set to $U_P = 75\%$. For each periodic task, the utilization is then calculated by the UUnifast algorithm [5], the period is randomly selected from the set $\{5, 10, 20, 50, 80, 100\}$ms, the deadline is equal to the period, and the WCET is the product of the utilization and period.

The AVR task implements six different control strategies. Since the switching speeds and consequently the inter-release times of the AVR task are design variables, its utilization is also unknown. Hence, we generate the WCET of each control strategy as a product of a base WCET and a scaling factor η. The base WCETs of these strategies are uniformly selected from $[100, 1000]\mu s$ with a minimum step of $100\mu s$. The scaling factor η is used to control the computational requirement from the AVR task. The AVR task has an angular period/deadline of one full revolution of the crankshaft.

We generate 500 random task sets, 30 performance functions, and apply the methods to these systems over the aforementioned 10 driving cycles. Hence, each point in the following figures is averaged over $500 \times 30 \times 10 = 150,000$ different combinations of task set, performance function, and driving cycle. Let $\Delta = 100$ ms, which means the schedulability test designed in Section 4 and the analysis proposed in [2] have the same precision since the maximum period/deadline is only 100 ms. We normalize the average control performance by the upper bound PERF-UB, hence PERF-UB is omitted from the figures.

5.1.1 *Performance Comparisons*

We first compare the methods with constant performance functions, where $j^{\min} = 1, j^{\max} = 50$. As in Figure 4, dAVR-exact is

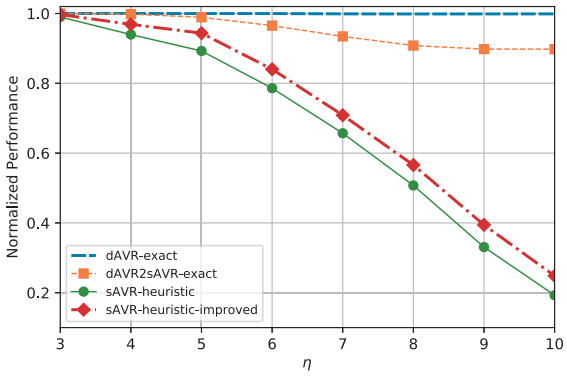

Figure 5: Normalized performance vs. scaling factor η for exponential performance functions.

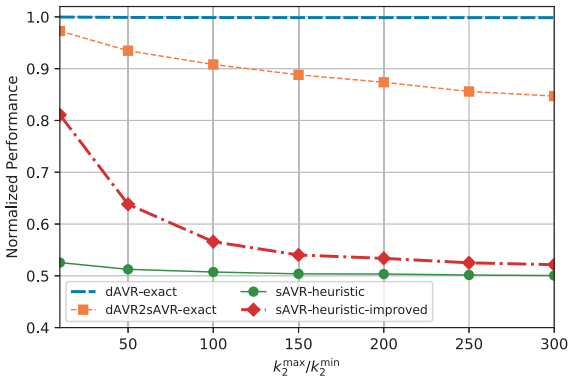

Figure 6: Normalized performance vs. ratio $k_2^{\max}/k_2^{\min}$ for exponential performance functions.

always approaching the performance upper bound PERF-UB. This indicates that although the analysis dAVR may be inaccurate based on the results in [2], it does not have noticeable negative impact on the engine control performance. On the other hand, dAVR2sAVR-exact is apparently suboptimal compared to dAVR-exact (e.g., 10% at $\eta = 10$) due to its substantially pessimistic schedulability analysis, which demonstrates the necessity for the analysis techniques proposed in this paper. With respect to the state-of-the-art approach sAVR-heuristic and its improvement sAVR-heuristic-improved, both are significantly inferior to dAVR-exact, due to their inability to dynamically adjust the engine configurations.

We then compare the methods with exponential performance functions. We fix $k_2^{\min} = 50$ rpm, $k_2^{\max} = 100 \cdot k_2^{\min}$, and vary the scaling factor η from 3 to 10 with a step of 1. Figure 5 illustrates the results. We also study how the methods perform under different ratios of $k_2^{\max}/k_2^{\min}$, by fixing the scaling factor $\eta = 8$ and vary $k_2^{\max}/k_2^{\min}$ from 10 to 300 (the higher the ratio, the larger the difference between the performance functions). The results are summarized in Figure 6. As in both figures, dAVR-exact has a normalized performance metric almost always equal to 1, or essentially the same as the performance upper bound from PERF-UB. Also, dAVR-exact is consistently outperforming dAVR2sAVR-exact (by as much as 15%), as well as sAVR-heuristic and sAVR-heuristic-improved (by as much as 80%).

In summary, from Figures 4–6, the experimental results confirm the potential benefits of dynamic adjustment of engine configuration under various settings of task sets, performance functions, and driving profiles.

5.1.2 *Runtime Comparisons*

Finally, we measure the runtime of our optimization algorithm dAVR-exact, which are implemented in the C++ language. We use a low-cost Raspberry Pi 3 [28] as the experiment platform, which has one 1.2GHz 64-bit quad-core CPU and 1GB memory. The choice is motivated by the recent adoption of Raspberry Pi based embedded microcontrollers in automotive due to their low price and good computational capabilities, such as Carberry [17], iCarus [23], and online algorithms to reduce engine emissions [32]. The average runtime for our approach dAVR-exact is 90ms, while the maximum runtime is close to one second. This is generally shorter than or comparable to the typical sample time of driving cycles (500ms) [4, 1]. We note that the engine reconfiguration is rather a soft real-time task, in that missing a data point or two in the driving profile will only affect the control performance, but the system schedulability is always guaranteed.

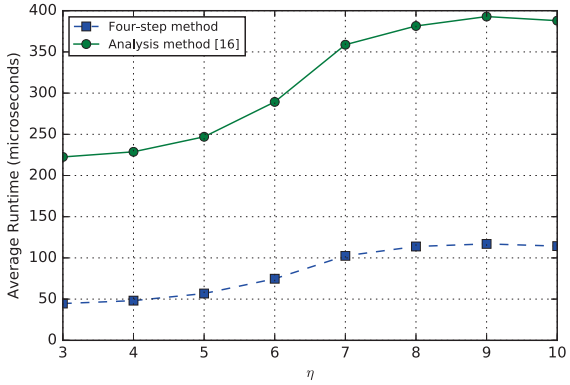

Figure 7: Runtime vs. scaling factor η.

Since the optimization runtimes are only related to the tested task systems and the upcoming engine speeds not the performance functions, the runtime results with the constant and exponential performance functions are consistent. In Fig. 7, we report the runtime results for evaluating the improvement of applying the four-step schedulability test over the analysis method proposed in [2]. As observed in this graph, the average runtimes of the two methods increase with the increase of the scaling factor η and always are less than one millisecond. This is generally much shorter than to the typical sample time of driving cycles (500ms) [4, 1], and hence our approach can be applied into the on-line optimization. The proposed four-step method can reduce the complexity compared with the method in [2] and the largest improvement is around 4 times at $\eta = 3$. The observed maximum runtimes of the two methods with the same parameter settings are similar and close to 40 milliseconds, since their worst case complexities are same.

5.2 Large-Size Task Systems

In the set of experiments, we also evaluate the applicability of our methods in the 500 randomly generated task systems with the constant and exponential performance functions over the ten driving cycles, which are similar to Section 5.1 but the periods are set as [25]. Specifically, for each tested system there are 20 periodic tasks where the periods are randomly selected from the set $\{1, 2, 5, 10, 20, 50, 100, 200, 1000\}$ms. Δ is also set as 100 ms.

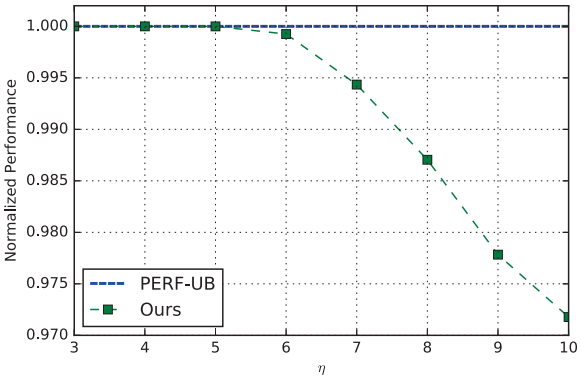

Figure 8: Normalized performance vs. scaling factor η for constant performance functions.

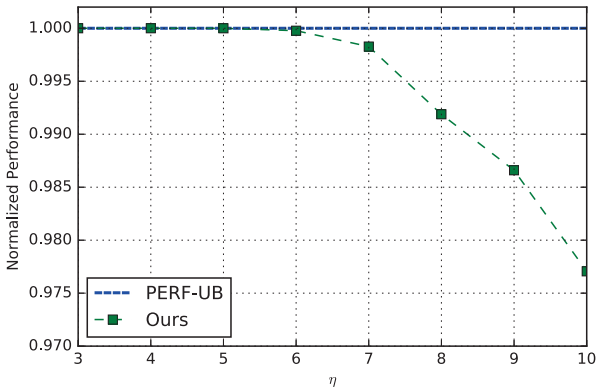

Figure 9: Normalized performance vs. scaling factor η for exponential performance functions.

We compare two optimization algorithms as follows.

- Ours: Our proposed optimization framework, which will dynamically adjust the switching speeds to maximize the control performance for each data point in the engine speed profiles, while using the schedulability test designed in Section 4.
- PERF-UB: The performance upper-bound, achieved with the necessary-only analysis similar to **dAVR2sAVR-NO**, which assumes each execution mode m as one sporadic task without any mode switch. The minimum inter-release time of this sporadic task is set as the minimum inter-arrival time for two jobs released in this execution mode. Since the engine may be reconfigured at time $k \cdot 500$ ms ($k \in \mathbb{N}^+$), the WCET of any job released by this sporadic task equals to the execution time of this mode if the job released within the time interval [0,500] ms; the execution time of the simplest control strategy otherwise.

We note that the performance upper bound [13, 14] is more precise (and less) than PERF-UB since it is obtained by additionally counting for the mode switches between the mode under consideration and the simplest one. However, the calculation of this performance upper bound cannot be finished in 48 hours (even on a machine with an Intel Core i7 3.4Ghz CPU) for adopting the time consuming exact analysis [12, 10]. At this point, we normalize the average control performance by the upper bound PERF-UB due to its much lower complexity.

5.2.1 *Performance Comparisons*

Figure 8 illustrates the average normalized performance as a function of the scaling factor η for the constant performance functions. The performance derived by Ours decreases with the increase of η and is always larger than the 97% of the performance upper bound PERF-UB. When $\eta \in [3, 6]$, Ours is extremely close to the upper-bound.

Figure 9 illustrates the average normalized performance as a function of the scaling factor η for the exponential performance functions. The observed results are consistent with Fig. 8. At $\eta = 10$, the performance of Ours is around 97.7%. Further, the performance results under different ratios of $k_2^{\max}/k_2^{\min}$ with the fixed scaling factor $\eta = 8$ is reported in Fig. 10. As in this figure, Ours has a normalized performance always larger than 99%. The performance results shown in Figures 8 , 9 and 10 confirm the potential benefits of our optimization framework for the large-size systems

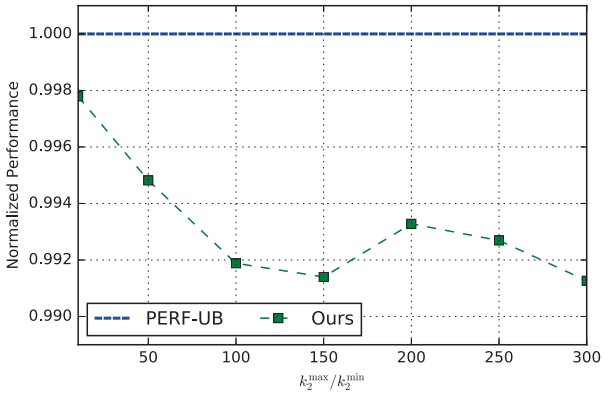

Figure 10: Normalized performance vs. ratio $k_2^{\max}/k_2^{\min}$ for exponential performance functions.

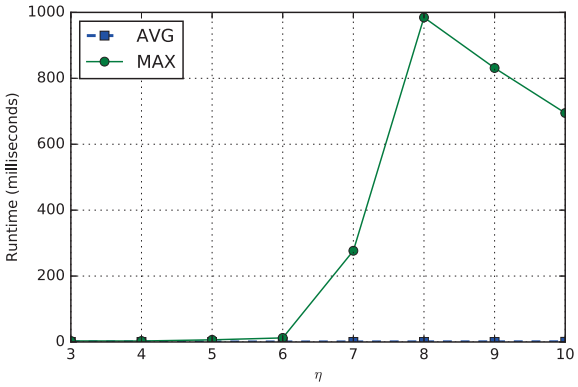

Figure 11: Runtime vs. scaling factor η.

and at the worse case the performance can reach more than the 97% of the upper-bound.

5.2.2 *Runtime Comparisons*

The runtimes as a function of the scaling factor η, varying from 3 to 10, are plotted in Fig. 11. As can be observed from this graph, all the collected maximum runtimes are below one second, while the average runtime is always less than one milliseconds (e.g., at $\eta = 10$ the average runtime is only 547 microseconds). This is generally shorter than or comparable to the typical sample time of driving cycles (500ms) [4, 1]. We note that the engine reconfiguration is

rather a soft real-time task, in that missing a data point or two in the driving profile will only affect the control performance, but the system schedulability is always guaranteed.

6. CONCLUSION

In this paper, we have defined the trip-based performance optimization problem for the engine control task system scheduling with fixed priority and proposed an approach to dynamically optimize the transition speeds. The schedulability test in our approach combines some sufficient or necessary analyses to speed up the optimization runtime. Our approach can obtain the performance, which is close to the upper bound, within small runtimes. For the large-size systems, the observed runtime is only around 500 μs. As a future work, we aim to study the schedulability test for the large-size systems with the better precision and more efficiency.

Acknowledgements

We would like to thank Alessandro Biondi for sharing the source code of the optimization algorithms presented in [13].

7. REFERENCES

[1] 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*, 544:299–308, 2016.

[2] Anonymous. Schedulability analysis of engine control tasks with dynamic switching speeds.

[3] AVL. Avl cruise vehicle driveline simulation. Website http://www.avl.com/cruise.

[4] 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.

[5] E. Bini and G. Buttazzo. Measuring the performance of schedulability tests. *Real-Time Systems*, 30(1-2):129–154, 2005.

[6] A. Biondi. *Analysis and Design Optimization of Engine Control Software*. PhD thesis, Scuola Superiore Sant Anna, Pisa, Italy, 2017.

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

[8] A. Biondi and G. Buttazzo. Modeling and analysis of engine control tasks under dynamic priority scheduling. *IEEE Transactions on Industrial Informatics*, PP(99):1–1, 2018.

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

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

[11] 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.

[12] 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.

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

[14] 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*, 2(1), 2018.

[15] 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.

[16] D. Buttle. Real-time in the prime-time. In *Keynote speech at Euromicro Conference on Real-Time Systems*, 2012.

[17] Carberry. Website http://www.carberry.it.

[18] R. Davis, T. Feld, V. Pollex, and F. Slomka. Schedulability tests for tasks with variable rate-dependent behaviour under fixed priority scheduling. In *IEEE Real-Time and Embedded Technology and Applications Symposium*, 2014.

[19] 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*, 138:100–107, 2017.

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

[21] 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.

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

[23] iCarus. Website http://www.i-carus.com.

[24] 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.

[25] S. Kramer, D. Ziegenbein, and A. Hamann. Real world automotive benchmark for free. In *Workshop on Analysis Tools and Methodologies for Embedded and Real-Time Systems (WATERS)*, 2015.

[26] 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.

[27] C. Peng, Y. Zhao, and H. Zeng. Trip-based performance optimization of engine control tasks with dynamic adaptation. Technical report, Feb 2018.

[28] R. Pi. Website https://www.raspberrypi.org.

[29] V. Pollex, T. Feld, F. Slomka, U. Margull, R. Mader, and G. Wirrer. Sufficient real-time analysis for an engine control unit. In *International Conference on Real-Time Networks and Systems*, 2013.

[30] 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 *Conference on Design, Automation Test in Europe*, 2013.

[31] M. Stigge, P. Ekberg, N. Guan, and W. Yi. The digraph real-time task model. In *IEEE Real-Time and Embedded Technology and Applications Symposium*, 2011.

[32] A. Vaughan and S. Bohac. An extreme learning machine approach to predicting near chaotic hcci combustion phasing in real-time. *arXiv preprint arXiv:1310.3567*, 2013.