Dynamic Switching Speed Reconfiguration for Engine Performance Optimization

Chao Peng
National Univ. Defense Tech., China
Email:pengchao06@gmail.com

Yecheng Zhao
Virginia Tech, USA
Email:zyecheng@vt.edu

Haibo Zeng
Virginia Tech, USA
Email:hbzeng@vt.edu

ABSTRACT

Today's automotive engine control systems adopt several control strategies that come with tradeoffs between computational load and performance. The current practice is that the switching speeds at which the engine control system changes control strategy is fixed offline, typically based on the average driving need in a standard driving cycle (i.e., vehicle speed profile over time). This is clearly suboptimal since it fails to capture the variation in the driving cycle, and the actual driving cycle may be considerably different from the standard one. In this paper, we propose to dynamically adjust switching speeds based on the predicted driving cycle. We develop a hybrid set of schedulability analysis techniques to tame the complexity of ensuring the real-time schedulability of engine control tasks. We design an effective and efficient optimization algorithm that provides close-to-optimal solutions. Experimental results demonstrate that our approach efficiently finds dynamic switching speeds that significantly improve engine performance over static ones.

KEYWORDS

Automotive Engine Performance Optimization, Dynamic Switching Speeds, Schedulability Analysis, Adaptive Variable-Rate Tasks

1 INTRODUCTION

In vehicles powered by the internal combustion engine, the engine control system determines the timing and amount of fuel injected in the engine, where the execution of certain software tasks (called *angular tasks*) is triggered at predefined rotation angles of the engine crankshaft. Due to its great impact on our environment and economy, any improvement on engine control performance (in terms of emission and fuel efficiency) may have significant benefits. For example, a mere one percent improvement in fuel economy for light-duty vehicles could save over $2.8 billion per year in U.S. in their fuel costs, based on the gas usage and price in 2017 [26].

Today's engine control systems adopt different control strategies at different engine rotation speed intervals [10]. The most sophisticated control strategy (e.g., with multiple fuel injections during one revolution) has the best performance. However, its excessively large computational demand will result in CPU overload on the hosting microcontroller at high engine speeds. Hence, engine control systems are designed to be self-adaptive in that they switch to simpler control strategies (such as single fuel injection) at higher engine speeds. Hence, angular tasks are often referred to as *Adaptive Variable-Rate (AVR)* tasks in the literature [25]. The behavior of AVR tasks strongly depends on the dynamics of the engine, making its analysis and optimization uniquely challenging.

Currently the configuration and calibration of the engine control system are performed *offline* using standard driving cycles. The engine configuration is then fixed at runtime (i.e., during vehicle operation) even if the driving route demands a completely different vehicle speed profile. This includes the optimization of the *switching speeds* (at which engine speeds engine control switches control strategies) [2]. However, this is clearly suboptimal due to the significant difference between the standard driving cycle and the actual ones, not to mention the variation within the same driving cycle.

This inefficiency will hopefully be mitigated in the upcoming era of Connected and Automated Vehicles (CAVs). With various sensing (e.g., camera, radar, lidar) and communication (such as vehicle-to-vehicle and vehicle-to-infrastructure) capabilities in place, valuable real-time information about the driving environment will become readily available. By carefully leveraging such information, it is possible to significantly improve the vehicle operations including path and speed planning [17], vehicle dynamics control [22], and engine control [6]. With the CAV techniques making the driving profile and engine speed more predictable than ever, we propose the concept of engine control with dynamic switching speeds to improve engine performance, where the switching speeds are dynamically adjusted according to the upcoming (expected) engine speed. The corresponding AVR tasks are called 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.

Optimizing engine control performance requires to address two problems [2]. One is to select the right set of control strategies and the corresponding switching speeds. The other is the schedulability analysis to make sure that the time-critical software tasks finish before their deadlines. Below we first review related work.

Related Work. The schedulability analysis of engine control systems is extensively studied in recent years. Kim et al. [16] describe AVR tasks with the rhythmic task model, but it comes with a few assumptions that may not match the engine dynamics, e.g., the inter-release time is shortened by a fixed ratio during any acceleration. Buttle introduces the typical software implementation of engine control systems [10], which is adopted by most of the studies thereafter.

Below we provide a *selective* review on the schedulability analysis techniques for studies consistent with the model by Buttle [10], and refer the readers to [25] for a more comprehensive discussion. For systems scheduled with fixed priority, Davis et al. [11] use quantization to discretize the continuous engine speed space, and present a sufficient analysis on the worst-case interference from AVR tasks. Instead, Biondi et al. [1] discover that a finite set of speeds is sufficient for calculating the exact worst-case interference, each of which dominates a range of speeds. This allows to develop exact analyses for systems scheduled with fixed priority [9] or Earliest Deadline First (EDF) [8]. Guo and Baruah [14] present a sufficient utilization-based schedulability test for EDF scheduled systems. Mohaqeqi

DAC '19, June 2–6, 2019, Las Vegas, NV, USA

ACM ISBN 978-1-4503-6725-7/19/06…$15.00
https://doi.org/10.1145/3316781.3317806

et al. [18] transform AVR tasks to digraph real-time tasks [19, 23] where each vertex represents a disjoint speed interval. Feld et al. [13] improve the analysis efficiency, whose approach may be inaccurate if the maximum acceleration and deceleration are different.

With respect to engine performance optimization, the lone work is by Biondi et al. [2], which proposes to maximize the engine performance by configuring the switching speeds at design time.

Our Contributions. All the above studies assume that the engine switching speeds are *fixed offline*. Differently, we propose the dAVR task model to allow dynamic adjustment to the switching speeds, and study the real-time schedulability analysis of such systems [20]. In this paper, we develop optimization algorithms that can efficiently configure engine switching speeds *at runtime*. However, adjusting switching speeds at runtime is extraordinarily difficult due to its high complexity. The studies on sAVR task systems (a special case of dAVR) take hours to find an engine configuration [2]. Also, existing analysis on systems with dAVR tasks is too complex to be scalable for direct online usage [20].

To address the above challenges, we develop a set of new analysis and optimization techniques. Specifically, we present a fast heuristic which avoids exploring the infinite space of switching speeds but still achieves performance that is close to the performance upper bound. Also, to mitigate the complexity of existing schedulability analysis, we combine a hybrid set of necessary-only and sufficient-only analyses to gain 10× speedup without losing any accuracy.

2 PROBLEM DEFINITION

2.1 System Model

Engine Dynamics. The engine is described by its current rotation angle θ, angular speed ω, and angular acceleration α. Due to the engine physical attributes, the angular speed and acceleration are restricted in certain ranges, i.e., $\omega \in [\omega^{\min}, \omega^{\max}]$ and $\alpha \in [\alpha^{\min}, \alpha^{\max}]$.

Engine Control System. The engine control system consists of a set of periodic tasks $\{\tau_1, \ldots, \tau_p\}$ and a set of AVR tasks. It is scheduled with fixed priority as specified by the automotive standard AUTOSAR and OSEK on operating systems. Since the AVR tasks share the same angular phase and period, for the purpose of analysis and optimization of switching speeds, *we can use one AVR task τ_A to represent the AVR tasks* [2]. The AVR task τ_A implements multiple engine control strategies $\mathbf{\Lambda} = \{\Lambda^1, \ldots, \Lambda^M\}$. Each strategy Λ^j is characterized by a WCET C^j and a performance function $\mathbf{P}^j(\omega)$. Similar to [2], we assume these implementations satisfy

$$\forall i > j, \forall \omega, \quad C^i < C^j \text{ and } \mathbf{P}^i(\omega) < \mathbf{P}^j(\omega) \tag{1}$$

This is based on the rationale that a more complex control implementation only makes sense if it improves the performance [2].

Periodic Task Model. 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 WCET, $D_i \le T_i$ is the constrained deadline, and P_i is the priority. The execution of periodic tasks, and thus their parameters are all *independent from the engine dynamics as well as the events triggering engine reconfiguration*.

AVR Task Model. The AVR task τ_A is triggered at predefined crankshaft angles $\theta = \Psi + k\Theta, \forall k \in \mathbb{N}$, where $\mathbb{N}$ is the set of non-negative integers, Ψ is the angular phase, and Θ is the angular period. Its angular deadline is $\rho = \lambda \cdot \Theta$ where $\lambda \le 1$ (hence τ_A also has a constrained deadline). Clearly the AVR task parameters WCET, inter-release time, and deadline all depend on the engine dynamics.

The **static AVR** (or **sAVR**) task model [10] assumes a fixed configuration including the switching speeds. In this model, an sAVR task τ_A contains a set $\mathcal{SM}$ of SM execution modes. Each mode m implements a control strategy $\Upsilon^m = \Lambda^j$, and is executed when the angular speed at the task release time is in the range $(\varsigma^{m-1}, \varsigma^m]$. Here $\varsigma^0 = \omega^{\min}$, $\varsigma^{SM} = \omega^{\max}$. Also $\forall m < SM$, let $\Upsilon^m = \Lambda^i$, $\Upsilon^{m+1} = \Lambda^j$, it must be $\varsigma^m < \varsigma^{m+1}$ and $i < j$ (hence $C^i > C^j$). Thus, the set of execution modes of an sAVR task τ_A can be described as

$$\mathcal{SM} = \{(\Upsilon^m, \varsigma^m), m = 1, \ldots, SM\} \tag{2}$$

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

$$\mathcal{SC}(\omega) = C^j \quad \text{if } \omega \in (\varsigma^{m-1}, \varsigma^m] \text{ and } \Upsilon^m = \Lambda^j \tag{3}$$

The **dynamic AVR** (or **dAVR**) task model [20] allows to dynamically adjust the switching speeds. The reconfiguration happens at times $\mathcal{T} = \{\gamma_1, \cdots, \gamma_T\}$, and may be triggered by events independent from those activating the periodic tasks. The dynamic AVR task τ_A has a series of execution mode sets defined as

$$\mathcal{Q} = \quad \{(\mathcal{M}_k, \gamma_k), k = 1, \ldots, T\}, \tag{4}$$

where $\mathcal{M}_k = \{(\Xi_k^m, \xi_k^m), m = 1, \ldots, M_k\}$ contains a set of M_k modes, within which each mode (Ξ_k^m, ξ_k^m) implements the strategy $\Xi_k^m \in \mathbf{\Lambda}$ in the speed interval $(\xi_k^{m-1}, \xi_k^m]$. The WCET of the dAVR job released at time t with instantaneous engine speed ω is

$$\mathcal{C}(t, \omega) = C^j \quad \text{if } t \in [\gamma_k, \gamma_{k+1}) \text{ and } \omega \in (\xi_k^{m-1}, \xi_k^m] \text{ and } \Xi_k^m = \Lambda^j.$$

2.2 Problem Formulation

Inputs. As part of the input, the engine control system will receive an updated prediction of the engine speed. We assume that the predicted engine speed $\omega(\gamma_k)$ will be effective at time γ_k. This can be done in two steps. First, the vehicle speed can be predicted using CAV techniques, as studied in a number of papers (see a recent review [12]). Second, the engine speed can be derived using, e.g., the relationship between vehicle cruise speed and engine speed [15].

The other inputs are the set of real-time tasks consisting of periodic tasks $\{\tau_1, \ldots, \tau_p\}$, and the dAVR task τ_A containing multiple engine control implementations $\{\Lambda^1, \ldots, \Lambda^M\}$. The task parameters are all fixed, including all parameters of periodic tasks, the angular period and deadline of the dAVR task, the WCET and performance function of the engine control implementations, and task priorities.

Variables. In this paper, we focus on the switching speed configurations as the variable to be dynamically adjusted. That is, once a prediction on the engine speed at γ_k is received, we seek to find a new execution mode set $\mathcal{M}_k$ to be effective at γ_k. Note that at time γ_k we do not assume any knowledge on the next reconfiguration time γ_{k+1}, nor its predicted engine speed $\omega(\gamma_{k+1})$.

Constraints. We must ensure that the engine control system is schedulable due to its time-criticality [2]. That is, all tasks in the system must finish no later than their deadlines. We remark that, due to the possible prediction errors or other uncertainties, we shall make pessimistic assumptions in the schedulability analysis: the system shall be schedulable under any possible engine speed profile in the future, as long as it is compliant with the engine dynamics.

Objective. We seek to optimize the engine control performance. We do not assume any particular form of the performance metric, as long as it satisfies the property in (1).

3 OPTIMIZATION ALGORITHM

Unlike design time configuration where the designer may spend days to optimize, dynamic switching speed reconfiguration has to be performed in a fraction of a second (to catch up with the dynamics of typical driving cycle). There are two difficulties to make the reconfiguration algorithms scalable while providing good solution quality. One is to find the most performant yet schedulable set of control implementations and their switching speeds, where the space for switching speeds is continuous. The other is schedulability analysis, to check if the system is schedulable with a given solution. The existing analysis is of exponential complexity [20].

In this section, we discuss how to efficiently tackle the optimization problem. We first present a simple yet effective heuristic in Section 3.1 to select the control implementations and switching speeds. We then explain in Section 3.2 how to reduce the complexity of schedulability analysis without losing any accuracy.

3.1 Overall Optimization Algorithm

Our algorithm is based on Equation (1): a more complicated control implementation comes with the benefit of better performance. Hence, we consider a reconfiguration that only contains two control strategies. One may have high WCET but good control performance, the other is Λ^M which has the smallest WCET but bad performance, switching at the predicted engine speed $\omega(\gamma_k)$. Intuitively, such a solution provides the best average-case performance for the expected engine speed, while trying to accommodate possible worst-case computational demand as much as possible.

Algorithm 1 Pseudo-code for the optimization procedure.

```
1: procedure OPTIMIZATION({Λ^1, ..., Λ^M}, γ_k, ω(γ_k))
2:     for m ← 1 to M do
3:         M_k ← {(Λ^m, ω(γ_k)), (Λ^M, ω^max)};
4:         if SCHEDULABLE(Q ∪ {(M_k, γ_k)}) then
5:             Q ← Q ∪ {(M_k, γ_k)};
6:             return (M_k, γ_k);
7:     return INFEASIBLE;
```

The pseudo-code is illustrated in Algorithm 1. Specifically, given the speed prediction $\omega(\gamma_k)$ at time γ_k, the algorithm iterates from the most sophisticated implementation Λ^1 to the simplest one Λ^M, and tries to combine it with Λ^M to form a new configuration that switches at the predicted engine speed $\omega(\gamma_k)$ (Line 3). If one such configuration makes the system schedulable (as checked in Line 4 and detailed in Section 3.2), then it is chosen as the new configuration effective at time γ_k (Line 6) and added to the list of engine configurations stored in $\mathcal{Q}$ (Line 5). Otherwise, the system is deemed as infeasible (Line 7).

We now discuss the properties of the proposed algorithm.

PROPERTY 1. *If the task system is feasible, Algorithm 1 is able to find a feasible solution.*

PROPERTY 2. *Our approach directly assigns each switching speed according to the predicted engine speed and is independent from the performance function.*

PROPERTY 3. *Each engine configuration contains at most two execution modes.*

In Line 3 of Algorithm 1, a configuration containing only the simplest implementation Λ^M (thus with the smallest WCET) will be checked when $m = M$, before claiming the system is infeasible. Since any other configuration will require more computational demand and is thus more difficult to schedule than the configuration with a single implementation Λ^M, it must mean that Property 1 is true.

Property 2 is true since Line 3 of Algorithm 1 sets the switching speed according to the predictive speed, without using any information on the performance function (except that it satisfies Equation (1)). This is in sharp contrast with the state-of-the-art approach [2]. In the experiments, we will show that such a simple approach works very well, as it provides close-to-optimal solutions for a variety of performance functions.

Property 3 is straightforward (see Line 3). It may significantly reduce the analysis time due to the small number of execution modes.

3.2 Schedulability Analysis

The schedulability of the task systems containing the set of periodic tasks $\{\tau_1, \ldots, \tau_p\}$ and an AVR task τ_A can be verified by the analysis technique presented in [20]. In this section, we focus on the schedulability of each periodic task τ_i interfered by τ_A, which is the main contributor of the analysis runtime [20]. We first briefly recap the analysis in [20] and then provide an efficient hybrid approach that leverages a new set of schedulability analyses with different accuracy and complexity.

Before we detail our approach, we make three comments. First, unlike the optimization objective that aims to improve the average performance, the system schedulability shall provide worst-case guarantees. Hence, we make no assumptions on the engine speed at time γ_k or any time afterwards, since these predictions may not be accurate due to uncertainties. Second, since the next reconfiguration time γ_{k+1} and the engine speed $\omega(\gamma_{k+1})$ are unknown, we configure the engine at γ_k assuming it may be used for an undefined amount of time. Finally, for any task instance (i.e., job) with a deadline no larger than γ_k, its schedulability will not be affected by the new configuration, hence we exclude them from the analysis for time γ_k.

3.2.1 *Existing schedulability analysis (S4).*

Let τ_i be the periodic task under analysis, and $hp(i)$ be the set of periodic tasks with higher priority than τ_i. An exact analysis can be established by exhaustively enumerating all job sequences of τ_A, where τ_A contains the series of execution mode sets $\mathcal{Q}$ until γ_{k-1} and the new set $(\mathcal{M}_k, \gamma_k)$. We first define two useful concepts.

DEFINITION 1 (DAVR JOB SEQUENCE [20]). *A job (σ_l, ω_l) of the dAVR task τ_A is denoted by its release time σ_l and the engine speed ω_l at time σ_l. 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 the engine dynamics.*

DEFINITION 2 (INTERFERENCE FUNCTION OF DAVR JOB SEQUENCE [20]). *$\forall t \geq 0$, 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,*

$$\mathcal{A}.I(t) = \mathcal{C}_A(\sigma_1, \omega_1) + \sum_{l=2}^{n} \delta(\sigma_1 + t, \sigma_l) \cdot \mathcal{C}_A(\sigma_l, \omega_l) \qquad (5)$$

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

By the two definitions, the worst case response time (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 [20]

$$R(\tau_i, \mathcal{A}) = \min_{t>0} \left\{ t \mid C_i + \sum_{\tau_j \in hp(i)} \left\lceil \frac{t}{T_j} \right\rceil C_j + \mathcal{A}.I(t) \leq t \right\} \quad (6)$$

The above equation is based on the assumption that the periodic tasks and the dAVR task are triggered by independent sources, thus the worst-case scenario is that τ_i is released simultaneously with all its interfering tasks (including τ_A). Further, the WCRT $R(\tau_i, \tau_A)$ of τ_i is the maximum over all possible dAVR job sequences of τ_A

$$R(\tau_i, \tau_A) = \max_{\mathcal{A} \in \tau_A \cap \Delta_i} R(\tau_i, \mathcal{A}) \quad (7)$$

Here $\Delta_i = [\max(0, \gamma_k - D_i), \gamma_k]$ defines the analysis window for task τ_i, and $\mathcal{A} \in \Delta_i$ denotes that the first job release time of $\mathcal{A}$ is contained in Δ_i, i.e., $\sigma_1 \in \Delta_i$. Any schedulable job of τ_i with a deadline earlier than γ_k (hence with a release time smaller than $\max(0, \gamma_k - D_i)$) must have finished. Also, any job of τ_i released at or after γ_k is interfered in the same way by τ_A, since the analysis only concerns the engine configurations until γ_k (Line 4, Algorithm 1).

Obviously, the analysis in (7) is impractical, as it enumerates all dAVR job sequences which are infinitely many due to the continuous spaces of both job release time and engine speed. In [20] a set of techniques are developed to discretize them, but the complexity is still prohibitively high. In this paper, we propose to judiciously combine a set of necessary-only and sufficient-only analyses. This may improve analysis efficiency by an order of magnitude without losing any accuracy.

3.2.2 An efficient hybrid approach for schedulability analysis. Our approach is detailed in Figure 1. Besides the existing analysis S4 as the last sanity check, we additionally propose three analyses S1-S3. These analysis techniques are very efficient with good accuracy. They are called before S4, either to quickly confirm the schedulability of the solution (in the case of sufficient-only analysis S2 and S3), or to promptly rule out many unschedulable solutions (in the case of necessary-only analysis S1).

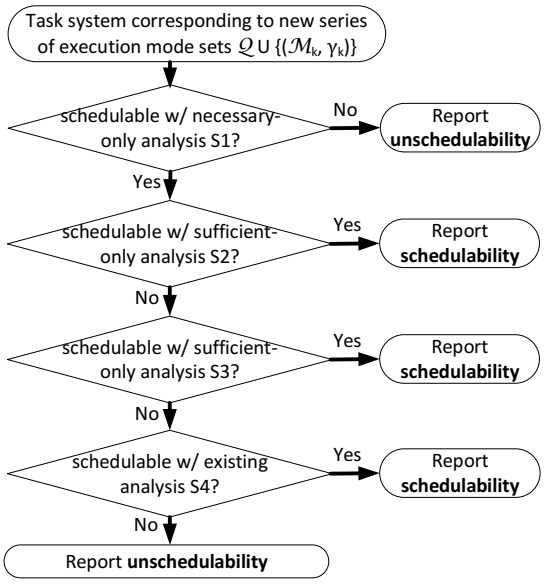

Figure 1: A hybrid approach for schedulability analysis.

This procedure preserves the same accuracy as S4, since (i) any solution that is deemed schedulable by a sufficient-only analysis must be schedulable by S4 as well; (ii) any solution that is deemed unschedulable by a necessary-only analysis must be unschedulable by S4 as well. Below we detail each of the new analyses S1-S3, but in a different order than that of Figure 1 for better readability.

3.2.3 Necessary-only analysis S1. The necessary-only analysis S1 (and the analysis S3) is based on the observation that the analysis for systems with sAVR tasks is substantially simpler than those with dAVR tasks. Specifically, an sAVR task has a unique characteristic that dAVR tasks do not satisfy: its WCET is independent from its release time. This makes two efficiency improvements possible. (i) Consider two job sequences $\mathcal{A}$ and $\mathcal{A}'$ which release jobs at the same sequence of angular speeds, but jobs in $\mathcal{A}$ are always released no later than $\mathcal{A}'$. It is safe to ignore $\mathcal{A}'$ and only consider $\mathcal{A}$ in Equation (7), since the interference function of $\mathcal{A}$ is always no smaller than that of $\mathcal{A}'$. (ii) There exist a small number of dominant speeds [1]. Each dominant speed dominates a range of smaller speeds which always produce sAVR job sequences with the same sequence of job WCETs as that of the dominant speed. That is, the dominant speed allows shorter inter-release times than the dominated ones while matching their sequence of job WCETs.

The analysis S1 is derived by creating an sAVR task τ_A^{S1} with an execution mode set $\mathcal{SM}^{S1} = \mathcal{M}_k = \{(\Lambda^m, \omega(\gamma_k)), (\Lambda^M, \omega^{\max})\}$, and then applying the existing analysis method on sAVR task systems [1, 18]. Hence, the WCET function $\mathcal{SC}^{S1}(\omega)$ of τ_A^{S1} is defined as C^m if $\omega \leq \omega(\gamma_k)$, and C^M otherwise.

The following theorem shows that S1 is necessary-only.

THEOREM 1. *If τ_i interfered by the dAVR task τ_A is schedulable, then τ_i interfered by the sAVR task τ_A^{S1} must be schedulable too.*

Proof. We prove its contrapositive. If τ_i interfered by τ_A^{S1} is unschedulable, then there must exist a job sequence of τ_A^{S1}, written as $\mathcal{A} = [(\sigma_1, \omega_1), \ldots, (\sigma_n, \omega_n)]$, such that $R(\tau_i, \mathcal{A}) > D_i$. We can construct another job sequence $\mathcal{A}' = [(\sigma_1', \omega_1), \ldots, (\sigma_n', \omega_n)]$ by delaying the release time of each job by γ_k, i.e., $\forall j \leq n: \sigma_j' = \sigma_j + t_k$. Obviously, $\mathcal{A}'$ is a valid job sequence of τ_A since the engine configuration of τ_A after γ_k is $\mathcal{M}_k$. Moreover, $\forall t \geq 0$, $\mathcal{A}'.I(t) = \mathcal{A}.I(t)$. Hence, $R(\tau_i, \mathcal{A}') = R(\tau_i, \mathcal{A}) > D_i$, and $R(\tau_i, \tau_A) > D_i$. □

3.2.4 sAVR task based sufficient-only analysis S3. We now generate another sAVR task τ_A^{S3} which permits a sufficient-only analysis. Specifically, for τ_i under analysis with an analysis window Δ_i, the WCET of the job of τ_A^{S3} released at speed ω is set as the *maximum* WCET of τ_A released at ω over Δ_i

$$\mathcal{SC}^{S3}(\omega) = \max_{l:\gamma_l \in \Delta_i} \{C^j \mid \omega \in (\omega_l^{m-1}, \omega_l^m] \text{ and } \Xi_l^m = \Lambda^j\} \quad (8)$$

Obviously the interference function of τ_A^{S3} upper bounds that of τ_A: for any job sequence $\mathcal{A} = [(\sigma_1, \omega_1), \ldots, (\sigma_n, \omega_n)] \in \tau_A$, we can construct another one $\mathcal{A}' \in \tau_A^{S3}$ that has the same job release times and engine speeds as $\mathcal{A}$. It is easy to see that $\forall t \geq 0$, $\mathcal{A}.I(t) \leq \mathcal{A}'.I(t)$. Hence, S3 must be a sufficient-only analysis.

3.2.5 Utilization-based sufficient-only analysis S2. Given the sAVR task τ_A^{S3}, we present a utilization-based schedulability test. Specifically, Davis et al. [11] propose an upper bound on the interference function of a job sequence for an sAVR task τ_A^{S3} as

$$\forall t \geq 0, \forall \mathcal{A} \in \tau_A^{S3}: \mathcal{A}.I(t) \leq U_A \cdot t + C_A^{\max}(1 - U_A) \quad (9)$$

where U_A and $C_A^{\max}$ denote the maximum utilization and maximum WCET of τ_A^{S3}, respectively. However, the direct calculation of U_A is complicated, and an upper-bound on U_A can be calculated as [7]

$$U_A^{ub} = \max_{\omega} \frac{\mathcal{C}_{\max}(\omega)}{T(\omega, \alpha^{\max})} \geq U_A \tag{10}$$

where $T(\omega, \alpha^{\max})$ is the minimum time to rotate Θ angles with the initial speed ω and maximum acceleration $\alpha^{\max}$. They also show that the lower-bound utilization U_A^{lb} can be achieved by the so-called steady-state utilization, i.e.,

$$U_A^{lb} = \max_{\omega} \frac{\mathcal{SC}^{S3}(\omega)}{T^{\min}(\omega)} \leq U_A \tag{11}$$

where $T^{\min}(\omega)$ is the minimum inter-release time between two consecutive jobs both of which are released at speed ω [18, 20].

Since $U_A^{lb} \leq U_{\max} \leq U_A^{ub}$, by Equation (9), the interference function of any job sequence of τ_A^{S3}, and consequently that of τ_A, is bounded by $U_A^{ub} \cdot t + C_A^{\max}(1 - U_A^{lb})$. Hence, using this bound on the interference of τ_A^{S3} provides a sufficient-only analysis.

4 EXPERIMENTAL EVALUATION

In this section, we evaluate the benefits of dynamic switching speed reconfiguration on engine control performance, as well as the efficiency of the proposed optimization algorithm. We adopt the engine parameters in [2]: (i) the minimum/maximum engine speed is 500/6500 rpm; (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 a list of optimization algorithms as follows:

- Ours-hybrid (resp. Ours-dAVR): Our proposed optimization algorithm to dynamically adjust the switching speeds, while using the proposed hybrid approach (resp. the one in [20]) for schedulability analysis.
- Heuristic: The backward search optimization algorithm from [2], which assumes full knowledge on the driving cycle. The schedulability analysis is from [9], which is exact for sAVR tasks. The original algorithm [2] simply returns failure when a calculated switching speed is less than the minimum engine speed. In such cases, we set its performance to be zero.
- Heuristic-improved: A small improvement over Heuristic. That is, when the original algorithm [2] fails to find a schedulable solution, we try to use a single execution mode that has the best performance while keeping the system schedulable.
- PERF-UB: The performance upper bound [2], which provides an upper bound on the optimal performance.

We note that there is another optimization algorithm (plain branch-and-bound) for sAVR task model in [2]. We compare with Heuristic only, since (i) Heuristic is very close to branch-and-bound in the optimized performance; (ii) branch-and-bound takes 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.

Engine Speed Profiles. 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 [4], while the rest is from [24]. Given a driving cycle, we use the commercial vehicle simulator AVL Cruise [5] to get the corresponding engine speed profile.

Engine Control Performance Functions. Like [2], we consider two types of engine control performance rate functions, which may be used to approximate a set of engine control performance metrics, such as power, fuel consumption, and emissions. The first type of performance functions is a constant function 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 $\mathbf{P}_e(\omega) = k_1 \cdot e^{-k_2/\omega}$, where k_1 and k_2 are two constant coefficients.

For constant functions, j is selected uniformly from $[j^{\min}, j^{\max}]$ with a minimum separation of 1 for each control strategy. For exponential functions, 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 $k_2 = e^{k_{2,sed}}$.

Task System Generation. We leverage an industrial case study [3], which contains an AVR task and 19 periodic tasks with period ranging from $700\mu s$ to $200ms$. The task priorities are given, where the AVR task has an overall priority order of 13 (hence with a higher priority than 7 periodic tasks). The task WCETs are unknown, but the number of instructions of the periodic tasks is provided.

We then follow [2] to generate the rest of the task parameters. The WCETs of periodic tasks are assigned proportionally to the number of instructions, such that their total utilization is 75%. 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 where we filter out the sets that are unschedulable under any switching speeds configuration, 30 performance functions, and apply the methods over the aforementioned 10 driving cycles. Hence, each point in the following figures is the average over $500 \times 30 \times 10 = 150,000$ different combinations of task set, performance function, and driving cycle.

Performance Comparison. We first compare the average control performance of the optimization methods, normalized by the upper bound PERF-UB. Hence PERF-UB is omitted from the figures.

For constant performance functions, we vary the scaling factor η from 3 to 10 with a step of 1, and set $j^{\min} = 1, j^{\max} = 50$. The performances of the optimization algorithms are illustrated in Figure 2. For exponential functions, Figures 3 shows the results where we fix $k_2^{\min} = 50$ rpm, $k_2^{\max} = 100 \cdot k_2^{\min}$. Figure 4 shows 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).

As in the figures, Ours-hybrid is always approaching to the performance upper bound PERF-UB (hence is always close to optimal). In contrast, Heuristic and Heuristic-improved both are substantially suboptimal, even if they know the complete driving cycle a priori. This is because of their incapability to capture the dynamics in a driving cycle. Meanwhile, as in Figure 5 for systems with varying scaling factor η, Heuristic-improved improves upon Heuristic, and (like Ours-hybrid) always finds a schedulable solution.

Runtime Comparison. We now demonstrate how the hybrid analysis approach improve the efficiency while keeping the same solution

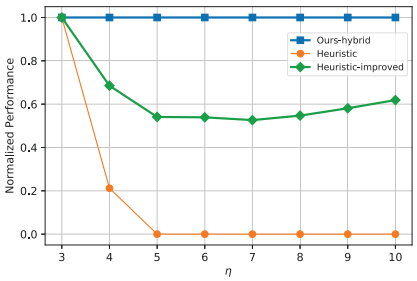

Figure 2: Normalized performance vs. scaling factor η for constant functions.

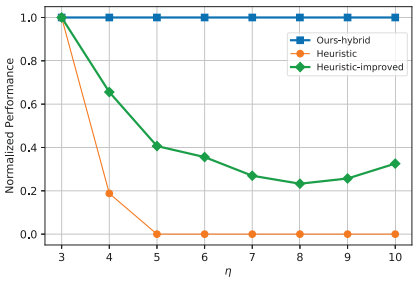

Figure 3: Normalized performance vs. scaling factor η for exponential functions.

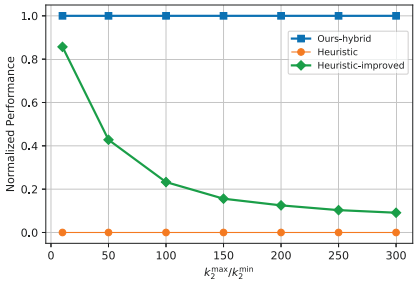

Figure 4: Normalized performance vs. ratio $k_2^{\max}/k_2^{\min}$ for exponential functions.

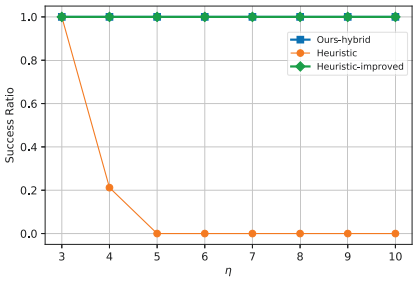

Figure 5: Ratio of schedulable solutions vs. scaling factor η for constant functions.

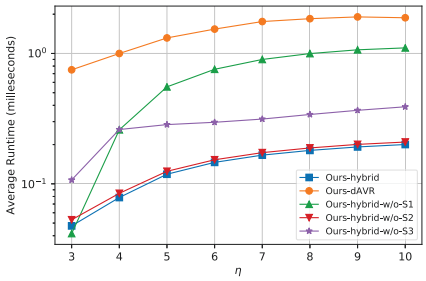

Figure 6: Average runtime vs. scaling factor η.

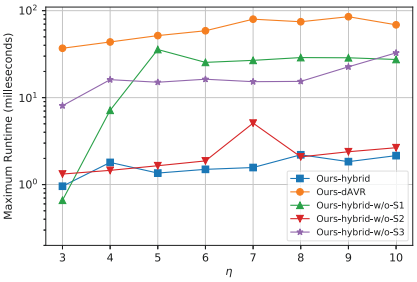

Figure 7: Maximum runtime vs. scaling factor η.

quality. We measure the runtime on a low-cost Raspberry Pi 3 [21] with a 1.2GHz 64-bit quad-core CPU and 1GB memory.

Figures 6 and 7 illustrate the average and maximum runtimes with varying scaling factor η. Overall, each of the analyses S1-S3 can effectively reduce the runtime and make our approach suitable for online usage. On average Ours-hybrid is 5.0× faster than Ours-hybrid-w/o-S1 (the analysis without S1), while the maximum runtime is 13.5× smaller. For S2, the speedups on average and maximum runtimes are 1.1× and 1.4× respectively. For S3, the speedups are 2.1× and 10.6× respectively. Hence, S1-S3 reduce the average and maximum runtimes of Ours-hybrid to 250μs and 2.2ms respectively, much smaller than the typical driving cycle sample time (500ms).

5 CONCLUSION

In this paper, we propose an approach to dynamically adjust the switching speeds in engine control systems. Our approach contains an effective heuristic that can obtain control performances close to the upper bound, and a hybrid set of schedulability analyses that significantly reduce the runtime while keeping the same solution quality. Experimental results demonstrate that our approach provides much better solutions than static configuration of switching speeds while being feasible for online usage.

ACKNOWLEDGEMENTS

This work is partially supported by NSF Grant No. 1812963.

REFERENCES

[1] A. Biondi et al. Exact interference of adaptive variable-rate tasks under fixed-priority scheduling. In *Euromicro Conference on Real-Time Systems*, 2014.
[2] A. Biondi et al. Selecting the transition speeds of engine control tasks to optimize the performance. *ACM Trans. Cyber-Physical Systems*, 2018.
[3] A. Hamann et al. Industrial challenge 2017. In *Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS)*, 2017.
[4] 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.
[5] AVL. Avl cruise vehicle driveline simulation. http://www.avl.com/cruise.
[6] B. Chen et al. Connected vehicles and powertrain optimization. *Mechanical Engineering Magazine Select Articles*, 139(09):S12–S18, 2017.
[7] A. Biondi and G. Buttazzo. Modeling and analysis of engine control tasks under dynamic priority scheduling. *IEEE Trans. Industrial Informatics*, 2018.
[8] A. Biondi, G. Buttazzo, and S. Simoncelli. Feasibility analysis of engine control tasks under edf scheduling. In *Euromicro Conference on Real-Time Systems*, 2015.
[9] A. Biondi, M. Di Natale, and G. Buttazzo. Response-time analysis of engine control applications under fixed-priority scheduling. *IEEE Trans. Comput.*, 2017.
[10] D. Buttle. Keynote speech: Real-time in the prime-time. In *Euromicro Conference on Real-Time Systems*, 2012.
[11] 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.
[12] E. Vlahogianni et al. Short-term traffic forecasting: Where we are and where we're going. *Transportation Research Part C: Emerging Technologies*, 2014.
[13] T. Feld and F. Slomka. Exact interference of tasks with variable rate-dependent behaviour. *IEEE Trans. Computer-Aided Design of IC and Systems*, 2017.
[14] Z. Guo and S. Baruah. Uniprocessor edf scheduling of avr task systems. In *ACM/IEEE Conference on Cyber-Physical Systems*, 2015.
[15] J. Happian-Smith. *An introduction to modern vehicle design*. Elsevier, 2001.
[16] J. Kim et al. Rhythmic tasks: A new task model with continually varying periods for cyber-physical systems. In *Conference on Cyber-Physical Systems*, 2012.
[17] J. Lin et al. A real-time en-route route guidance decision scheme for transportation-based cyber-physical systems. *IEEE Trans. Vehicular Technology*, 2017.
[18] M. Mohaqeqi, J. Abdullah, P. Ekberg, and W. Yi. Refinement of Workload Models for Engine Controllers by State Space Partitioning. In *ECRTS*, 2017.
[19] C. Peng and H. Zeng. Response time analysis of digraph real-time tasks scheduled with static priority: generalization, approximation, and improvement. *Real-Time Systems*, 2018.
[20] C. Peng, Y. Zhao, and H. Zeng. Schedulability analysis of adaptive variable-rate tasks with dynamic switching speeds. In *RTSS*, 2018.
[21] Raspberry Pi. https://www.raspberrypi.org.
[22] S. Li et al. Dynamical modeling and distributed control of connected and automated vehicles: Challenges and opportunities. *IEEE Intelligent Transportation Systems Magazine*, 9(3):46–58, 2017.
[23] 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.
[24] T. Barlow et al. A reference book of driving cycles for use in the measurement of road vehicle emissions. *TRL Published Project Report*, 2009.
[25] T. Feld et al. A survey of schedulability analysis techniques for rate-dependent tasks. *Journal of Systems and Software*, 2017.
[26] U.S. Energy Information Administration. *Use of Gasoline*. https://www.eia.gov/.