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 and hence cannot adjust to the actual situations (e.g., driving cycles). In this work, we propose the trip-based performance optimization framework of engine control tasks with dynamic adaptation and present a near-optimal and efficient optimization algorithm. Our approach has shown the significant performance improvement with simulation over many standard driving cycles.

1. INTRODUCTION

1.1 Related Work

1.2 Our contributions

This paper has the following contributions:

- We define the trip-based performance optimization problem of the engine control tasks with dynamic adaptation.
- We provide a general optimization framework and derive an on-line algorithm by effectively combining a series of schedulability analysis techniques.
- A number of experiments are conducted to evaluate the proposed approach.

2. SYSTEM MODEL

We now present the system model, following the study on AVR tasks (e.g., [6, 7, 12]). The list of notations is summarized in Appendix **??**. 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), $D_i \le T_i$ 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 $\theta_i = \Psi_i + k\Theta_i, \;\; k \in \mathbb{N}$, 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 [7], 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* [6, 7, 12]. The speed after rotating θ angles given an initial speed ω and a constant acceleration α is calculated as [6]

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

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

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

The minimum inter-release time between them, denoted as $T^{\min}(\omega_k, \omega_{k+1})$, is given as [12]

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

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

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

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 constant speed, and a maximum acceleration

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

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

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

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,

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

dAVR Task Model. In this paper, like [13] 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 [13]. 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

$$\begin{array}{rl} & \mathcal{Q}_i = \{(\mathcal{M}_{i,k}, \gamma_k), k = 1, \ldots, T\}, \\ \text{where} & \mathcal{M}_{i,k} = \{(C_{i,k}^m, \omega_{i,k}^m), m = 1, \ldots, M_{i,k}\} \end{array} \tag{8}$$

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

$$\mathcal{C}_i(t, \omega) = C_{i,k}^m \quad \text{if } t \in [\gamma_k, \gamma_{k+1}) \textbf{ and } \omega \in [\omega_{i,k}^m, \omega_{i,k}^{m+1}) \tag{9}$$

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. TRIP-BASED PERFORMANCE OPTIMIZATION PROBLEM AND FRAMEWORK

Time Interval (ms)	Execution Modes					
$[\gamma_1, \gamma_2) = [0, 100)$	$\mathcal{M}_{i,1}$	m-th mode	1	2	3	
		$\omega_{i,1}^m$ (rpm)	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 this section, we first formally define the trip-based performance optimization problem, and then provide one general optimization framework.

3.1 Problem Definition

In order to maximize the engine performance with respect to the engine speed profile, we need to dynamically select the control strategies and the corresponding switching speeds while guaranteeing the schedulability. Next, the optimization problem can be formalized as the following.

Inputs: a set of real-time tasks consisting of periodic and AVR tasks under fixed priority scheduling on a uni-processor, where there is an unset AVR task integrated with multiple control implementations, as well as one engine speed profile.

There are multiple control implementations with different computational load, which will be appropriately integrated into the unset AVR task. Similar to [8, 9], 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 used to approximate a set of engine control performance metrics, such as power, fuel consumption, and emissions [8]:

- 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.

In addition, we also assume the engine speed profile $\mathcal{W}$ corresponds to one certain driving cycle (vehicle speed with respect to time) and can be obtained by the commercial vehicle simulator AVL Cruise [2]. At this point, let $\{\varpi_1, \ldots, \varpi_W\}$ denote the set of sampling times of the engine speed profile (or its original driving cycle), and $\mathcal{W}(t)$ represent the engine speed at time t. Moreover, let ϖ_{W+1} be the finish time of this profile. We note that the resultant speed profiles have the fixed engine speed during each sampling period, i.e., $\forall t_1, t_2 \in [\varpi_k, \varpi_{k+1})$, $\mathcal{W}(t_1) = \mathcal{W}(t_2)$. As a result, we can use the simplified notation: $\mathcal{W}_k = \mathcal{W}(\varpi_k)$.

Variables: a set of reconfiguration times and for each time interval selected control implementations, as well as the corresponding switching speeds. That is to say, the optimization key is how to appropriately design the unset AVR task consisting of multiple control strategies.

Constraint: the time constraint. The task system is schedulable if and only if any job instance released by the task system cannot miss the deadline under the current fixed priority assignment.

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

$$
\begin{aligned}
PER &= \int_{\varpi_1}^{\varpi_{W+1}} \mathcal{P}_A(t, \mathcal{W}(t))dt \\
&= \sum_{k=1}^{W} \mathcal{P}_A(\varpi_k, \mathcal{W}_k) \cdot (\varpi_{k+1} - \varpi_k)
\end{aligned} \tag{10}
$$

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, and finally the final 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 corresponding performance rate and its time duration (or the sample period).

Optimization Objective: the maximum engine control performance, i.e., $\max PER$.

Outputs: a concrete parameter setting for τ_A^*.

3.2 Optimization Framework

Intuitively, the reconfiguration times of the AVR task τ_A^* can be assigned according to the sample times of the engine speed profile, and during each time interval $[\varpi_k, \varpi_{k+1})$ we will determine the optimal execution mode setting for speed $\mathcal{W}_k$ while satisfying the schedulability. The detailed optimization procedure is illustrated in Algorithm 1.

Algorithm 1 Pseudo-code for the optimization procedure.

```
1: procedure OPTIMIZATION($C_1, \ldots, C_Q, \mathcal{W}$)
2:     for $k = 1$ to $W$ do
3:         for $m = 1$ to $Q$ do
4:             $\mathcal{M}_k \leftarrow \{(C_m, \mathcal{W}_k), (C_Q, \omega^{\max})\}$;
5:             if SCHEDULABLE($\{(\mathcal{M}_j, \varpi_j), j \le k\}$) then
6:                 break;
7:     return $\{(\mathcal{M}_j, \varpi_j), j \le W\}$;
```

The algorithm search over the whole profile and design the engine configuration for each sampling speed (Lines 2-6). The intuitive idea is to allocate the optimal implementation for speed $\mathcal{W}_k$ under the schedulability constraints, shown in Lines 3-6. Actually, we note that when considering the m-th implementation the smallest workload can be obtained by Line 4. Here we require that procedure **Schedulable** is capable of entailing the schedulability of the system containing a series of execution mode sets (Line 5).

4. EXPERIMENTAL EVALUATION

In this section, we evaluate the benefits of the proposed approach on engine control performance, using randomly generated synthetic task systems. We adopt the engine parameters in [8, 9]: (i) the minimum/maximum engine speed is 500/6500 rpm; and (ii) the maximum acceleration/deceleration is 1.62×10^{-4} rev/msec^2. Hence, the engine needs 35 revolutions to accelerate/decelerate between the minimum and maximum speeds.

We evaluate the benefit of engine control performance using the dAVR task model, and compare a list of task models and optimization algorithms as follows:

- dAVR-exact: our proposed optimization framework in Algorithm 1, 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 analysis presented in [13].
- dAVR2sAVR-exact: Similar to dAVR-exact but using dAVR2sAVR fo schedulability analysis.
- sAVR-heuristic: The backward search optimization algorithm from [8, 9]. The schedulability analysis is from [7, 5], which is exact for sAVR tasks. The original algorithm [8, 9] 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 [8, 9] 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 [8], achieved with the necessary-only analysis UB as explained in Section **??**. Specifically, the switching speed ω_a of an execution mode a is calculated assuming there are only two execution modes: ω_a and the simplest one. ω_a is set to be the largest switching speed that maintains the system schedulability.

We note that there is another optimization algorithm (plain branch-and-bound) for sAVR task model in [8, 9]. 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 [3]. Given a driving cycle (the vehicle speed over time), we use the commercial vehicle simulator AVL Cruise [2] to get the corresponding engine speed profile.

4.1 Small-Size Task Systems

We generate the random task systems following [8]. 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 [4], 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. We normalize the average control performance by the upper bound PERF-UB, hence PERF-UB is omitted from the figures.

We first compare the methods with constant performance functions, where for each control strategy j is selected uniformly from

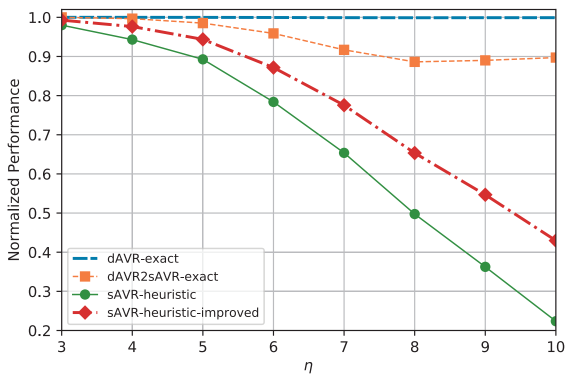

Figure 1: Normalized performance vs. scaling factor η for constant performance functions.

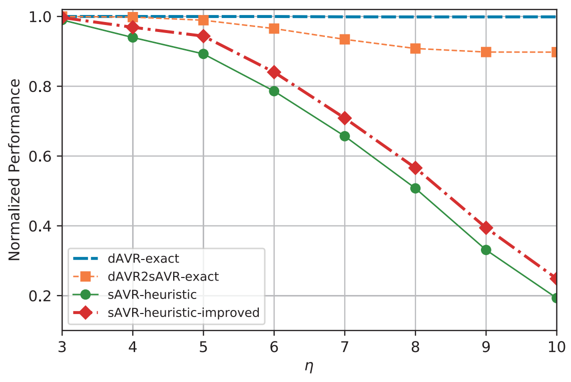

Figure 2: Normalized performance vs. scaling factor η for exponential performance functions.

$[1, 50]$ with a minimum separation of 1. As in Figure 1, dAVR-exact is always approaching the performance upper bound PERF-UB. This indicates that although the analysis dAVR may be inaccurate based on the results in Section **??**, 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. 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 $k_2 = e^{k_{2,sed}}$. 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 2 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 3. 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 1–3, the experimental results confirm the potential benefits of dynamic adjustment of engine configuration under various settings of task sets, performance functions, and driving profiles.

Finally, we measure the runtime of our optimization algorithm dAVR-exact. We use a low-cost Raspberry Pi 3 [14] 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 [10], iCarus [11], and the efforts for online adaptation algorithms to reduce engine emissions [15]. The average runtime for our approach dAVR-exact is 90ms, while the maximum runtime is

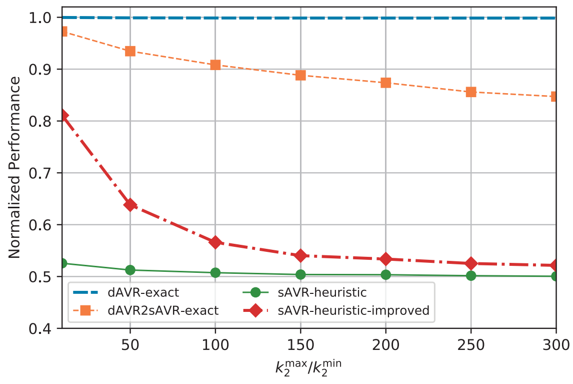

Figure 3: Normalized performance vs. ratio $k_2^{\max}/k_2^{\min}$ for exponential performance functions.

close to one second. This is generally shorter than or comparable to the typical sample time of driving cycles (500ms) [3, 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 not the system schedulability.

4.2 Real-World Benchmark

4.3 An Automative Case Study

5. CONCLUSION

In this paper, we defined the trip-based performance optimization problem for the engine control task system scheduling with fixed priority and proposed an efficient optimization algorithm to dynamically optimize the transition speeds. This new algorithm combines some sufficient-only or necessary-only analyses to speed up the runtime. In addition, a number of experiments have been conducted to evaluate our method over many driving cycles. The experimental results indicate that our method has significant performance improvement. As a future work, we aim to study the optimization problem for the engine control task system under EDF scheduling.

Acknowledgements

We would like to thank Alessandro Biondi for sharing the source code of the optimization algorithms presented in [8].

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

[3] 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.

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

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

[6] 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.

[7] 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.

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

[9] 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.

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

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

[12] 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.

[13] C. Peng, Y. Zhao, and H. Zeng. Schedulability analysis of engine control tasks with dynamic switching speeds. 2018.

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

[15] 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.