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

Introduction of the performance optimization based on the driving cycles.

1.1 Related Work

Optimization Problem.

In addition, Biondi et al. [8] maximize the performance of the task set comprising a few of periodic tasks and one AVR task scheduling with the fixed priority by selecting the transition speed for each mode. They assume that the implementation with the larger execution time shall result in the better performance. It is reasonable in the engine design since in general the more advanced control strategies require more complicate operation. For instance, some fuel injection control applications have triple injections with low engine speed and only single injection with high engine speed. In our work, we also hold this assumption.

Unfortunately, there are three main defects preventing the use of the proposed approaches in [8]. **Firstly, they have the implicit assumption that each engine speed has the same probability for computing the overall performance.** However in reality, the vehicle speed shall vary with the time during driving, which can be abstracted as the driving cycles. A driving cycle is a set of data points indicating the vehicle speed with respect to the time [11] and has been widely used in the analysis and design optimization of electric vehicles. The reference book [3] presents more than 200 standard driving cycles in the formal format. As a result, the engine speed also changes accordingly and the equal-probability assumption does not hold in the driving cycles. **Secondly, the transition speeds cannot be re-adjusted during driving.** That is why we call the existing approaches as the static optimization. **Thirdly, the time complexity is very high since of a large number of the schedulability checks using the exact analysis during the optimization procedure.** In [8], the search time of the branch and bound method (**BB**) is up to 4.5 hour and the backwards search algorithm (**BS**) also has the average runtime of 5 seconds. And in the experiments the maximum period of the periodic tasks is only 100 ms, while in the real world it can be up to 1000 ms [9] which significantly increases the analysis runtime.

In this work, we focus on the real-time applications consisting of a set of periodic tasks with the common activation timer and one single AVR task scheduled under fixed priority and study how to dynamically select the transition speeds for the control strategy implementations with different computation load to optimize the overall performance (e.g, fuel consumption, emission and energy) with respect to the driving cycles.

1.2 Our contributions

This paper has the following contributions:

- We describe the motivated example to show the defect of the previous work and present the performance optimization problem based on the engine profile.
- We propose the fast optimization framework and the corresponding response time analyses for the real-world task systems.
- A number of experimental results are reported to indicate our optimization approach can improve the runtime while guaranteeing the overall performance.

The rest of the paper is organized as follows. Section **??** describes the AVR task model and summarizes the existing response time analyses. Section **??** summarizes the existed exact and sufficient-only response time analyses, and presents the necessary-only response time analyses. Section 3 defines the performance optimization problem based on driving cycles and describes a running example. Section **??** presents an optimization algorithm to fast compute one suboptimal solution. Section 6 evaluates our approaches. Finally, Section 7 concludes the paper.

2. SYSTEM MODEL AND ENGINE DYNAMICAL CHARACTERISTICS

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

In this section, we present the system model considered in this work, as well as some useful dynamical characteristics for the rotation source. The rotation source can be described by the current rotation angle θ with the current angular velocity ω and acceleration α. Due to the physical attributes of the rotation source, the angular velocity and acceleration are restricted in the certain ranges, i.e., $\omega \in [\omega^{\min}, \omega^{\max}]$ and $\alpha \in [\alpha^{\min}, \alpha^{\max}]$. In order to deriving the general and precise analysis, we consider the changing of the acceleration during the revolution of the crankshaft, which on the contrary is assumed to be negligible in [6, 7, 5]. In addition, there is no relation between $\alpha^{\min}$ and $\alpha^{\max}$.

2.1 System Model

We consider an engine control system Γ consisting of a number of real-time tasks, where each task can be a periodic task or a dynamic AVR task, scheduled with fixed priorities running on a uni-processor. We denote the real-time task set as $\Gamma = \Gamma_P \cup \Gamma_A$ (satisfying $\Gamma_P \cap \Gamma_A = \emptyset$) where Γ_P and Γ_A respectively represent the sets of periodic and AVR tasks. For convenience, a periodic task is denoted as τ_i while an AVR task represented as τ_i^*.

A periodic task τ_i is characterized by a tuple $\langle T_i, o_i, C_i, D_i \rangle$, where T_i is the period, o_i is the release offset, C_i is the worst-case execution time (WCET), D_i is the deadline satisfying $D_i \leq T_i$ (therefore hold the constrained deadlines). The j-th instance of τ_i, denoted as $\tau_{i,j}$, is released at $r_{i,j} = o_i + (j-1) \cdot T_i$ and has a deadline of $r_{i,j} + D_i$.

Furthermore, an AVR task τ_i^* is activated at the specific crankshaft angles:

$$\theta_i = \Psi_i + k\Theta, \;\; k \in \mathbb{N}$$

where Ψ_i and Θ_i denote the angular phase and period respectively, and $\mathbb{N}$ is a set of the non-negative integers. We assume the angular deadline is expressed as $\lambda_i \cdot \Theta_i$ where $\lambda_i \in [0, 1]$ (hence enforce the constrained deadlines).

A **dynamic** AVR task τ_i^* has a series of configurations for the transition speeds with the configuration switching time, which is expressed as

$$\mathcal{Q}_i = \{(\mathcal{M}_{i,k}, \gamma_{i,k}), k = 1, \ldots, Q_i\}$$

where $\mathcal{M}_{i,k} = \{(C_{i,k}^m, \omega_{i,k}^m), m = 1, \ldots, M_{i,k}\}$ and Q_i denote the number of configurations. Without loss of generality, let $\gamma_{Q+1} = +\infty$. Each time inside the time window $[\gamma_{i,k}, \gamma_{i,k+1})$ corresponds to the specific execution modes $\mathcal{M}_{i,k}$, formally as

$$\mathcal{M}_i(t) = \mathcal{M}_{i,k} \text{ if } t \in [\gamma_{i,k}, \gamma_{i,k+1})$$

Further, the execution time of one job instance from the dynamic AVR task τ_i^* released at time t and speed ω is determined by the following.

$$\mathcal{C}_i(t, \omega) = C_{i,k}^j \text{ if } t \in [\gamma_{i,k}, \gamma_{i,k+1}) \textbf{ and } \omega \in (\omega_{i,k}^{j+1}, \omega_{i,k}^j]$$

The calculation of $\mathcal{C}_i(t, \omega)$ is implemented by two steps: first determine which configuration is used at time t; then based on this configuration the execution time is selected with respect to the release speed ω, illustrated in Fig. 1.

Table 1 shows an illustrative example for the switching configurations of the dynamic AVR task τ_i^*. The engine configuration will dynamically change with the time. For example, within $[0, 1000)$ (ms) the engine uses the execution modes $\mathcal{M}_{i,1}$ containing the three modes, while at time 1000 ms the engine is switched to the configuration with the execution modes $\mathcal{M}_{i,2}$ with six modes.

Following [7], we assume these AVR tasks have the common rotation source with the same angular period and phase. To simplify the notation, the task index for the angular period and phase is omitted, i.e., Θ and Ψ.

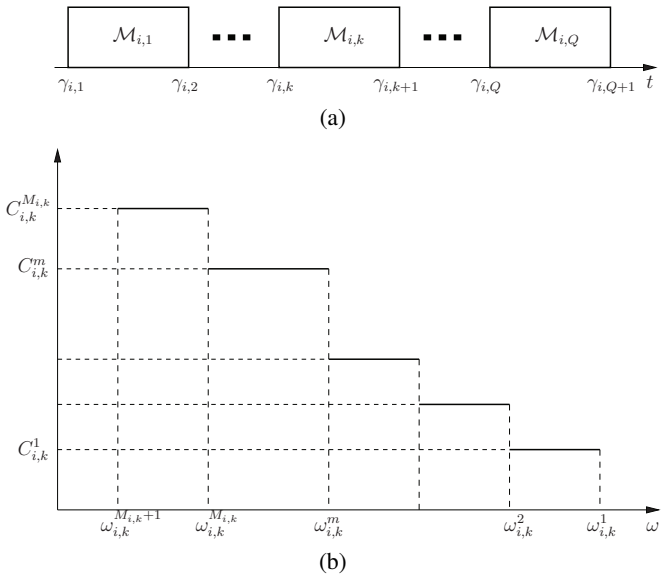

Figure 1: (a) First step: determining the configuration for the release time, (b) second step: determining the execution time for the release speed under the configuration obtained from (a).

Table 1: An illustrative example of the dynamic AVR task.

Time Interval (ms)		Engine Configuration						
$[0, 1)$	$\mathcal{M}_{i,1}$	j mode	1	2	3			
		$\omega_{i,1}^j$ (rpm)	6500	4500	2500			
		$C_{i,1}^j$ (μs)	150	400	600			
$[1, 2)$	$\mathcal{M}_{i,2}$	j mode	1	2	3	4	5	6
		$\omega_{i,2}^j$ (rpm)	6500	5500	4500	3500	2500	1500
		$C_{i,3}^j$ (μs)	150	278	344	425	576	966
$[2, +\infty)$	$\mathcal{M}_{i,3}$	j mode	1	2	3	4		
		$\omega_{i,3}^j$ (rpm)	6500	5000	3500	2000		
		$C_{i,3}^j$ (μs)	150	300	450	600		

2.2 Engine Dynamical Characteristics

To specify the interference from the AVR task, it is necessary to establish the relations between the rotation source and the AVR task parameters. In this subsection, we restate the engine dynamical characteristics.

Given an instantaneous speed ω and the constant acceleration α, the next instantaneous speed after θ' angles is

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

Now considering two speeds ω_k and ω_{k+1} satisfying ω_{k+1} can be reached from ω_k after one angular period Θ under the engine limitations, the elapsed time $T(\omega_k, \omega_{k+1})$ between the two speeds is restricted within $\left[T^{\min}(\omega_k, \omega_{k+1}), T^{\max}(\omega_k, \omega_{k+1})\right]$. Thanks to [10], the minimum elapsed time between ω_k and ω_{k+1} within one angular period, $T^{\min}(\omega_k, \omega_{k+1})$, can be calculated as the following procedure based on the observation in Fig. 2.

- We first compute ω' which denotes the maximum speed such that the rotation angular coming from accelerating ω_k to ω' with $\alpha^{\max}$ and then decelerating ω' to ω_{k+1} with $\alpha^{\min}$ equals

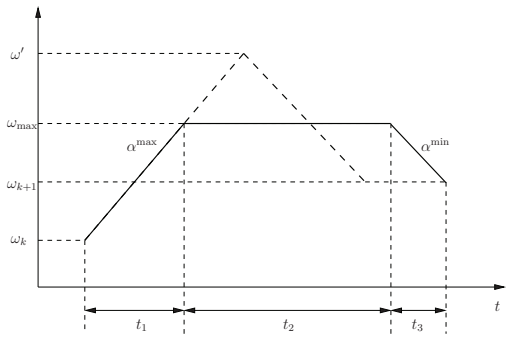

Figure 2: Illustration of calculating $T^{\min}(\omega_k, \omega_{k+1})$.

to the angular period Θ and is expressed as

$$\omega' = \sqrt{\frac{\alpha^{\max}\omega_{k+1}^2 - \alpha^{\min}\omega_k^2 - 2\alpha^{\min}\alpha^{\max}\Theta}{\alpha^{\max} - \alpha^{\min}}} \quad (2)$$

- However, due to the limitation of the maximum engine speed, ω' is invalid if $\omega' > \omega^{\max}$. At this case, there must exist one time interval with the maximum angular speed $\omega^{\max}$ where the acceleration is 0. Thus for convenience, let t_1, t_2 and t_3 respectively denote the time intervals with the acceleration of $\alpha^{\max}$,0, and $\alpha^{\min}$, computed by

$$\begin{cases} t_1 = \frac{\omega^* - \omega_k}{\alpha^{\max}} \\ t_2 = \frac{\Theta - \frac{(\omega^*)^2 - \omega_k^2}{2\alpha^{\max}} - \frac{(\omega^*)^2 - \omega_{k+1}^2}{-2\alpha^{\min}}}{\omega^{\max}} \\ t_3 = \frac{\omega^* - \omega_{k+1}}{-\alpha^{\min}} \\ \text{where } \omega^* = \min(\omega^{\max}, \omega') \end{cases} \quad (3)$$

Note that $t_2 = 0$ if $\omega' \leq \omega^{\max}$.

- Finally, it holds

$$T^{\min}(\omega_k, \omega_{k+1}) = t_1 + t_2 + t_3 \quad (4)$$

Similarly, we can also derive the maximum elapsed time between the speeds ω_k and ω_{k+1} within Θ as

$$T^{\max}(\omega_k, \omega_{k+1}) = t_1 + t_2 + t_3 \quad (5)$$

where

$$\begin{cases} \omega' = \sqrt{\frac{\alpha^{\max}\omega_k^2 - \alpha^{\min}\omega_{k+1}^2 + 2\alpha^{\min}\alpha^{\max}\Theta}{\alpha^{\max} - \alpha^{\min}}} \\ t_1 = \frac{\omega_k - \omega^*}{-\alpha^{\min}} \\ t_2 = \frac{\Theta - \frac{(\omega^*)^2 - \omega_k^2}{2\alpha^{\min}} + \frac{(\omega^*)^2 - \omega_{k+1}^2}{2\alpha^{\max}}}{\omega^{\min}} \\ t_3 = \frac{\omega_{k+1} - \omega^*}{\alpha^{\max}} \\ \text{where } \omega^* = \max(\omega^{\min}, \omega') \end{cases} \quad (6)$$

Hence, the minimum (or maximum) inter-arrival time between one job released at speed ω_k and the following at speed ω_{k+1} equals to $T^{\min}(\omega_k, \omega_{k+1})$ (or $T^{\max}(\omega_k, \omega_{k+1})$).

Although the angular deadline of an AVR task τ_i^* is fixed as $\lambda_i\Theta$, the actual time deadline of a job is related to the concrete release speed ω. Thus, we denote $D_i(\omega)$ as the time deadline of a job released at speed ω and for the safe analysis $D_i(\omega)$ is assigned as the minimum time for the engine rotates $\lambda_i\Theta$ angles, i.e.,

$$D_i(\omega) = t_1 + t_2 \quad (7)$$

where

$$\begin{cases} \omega' = \Omega(\omega, \alpha^{\max}, \lambda_i\Theta) \\ t_1 = \frac{\omega^* - \omega}{\alpha^{\max}} \\ t_2 = \frac{\lambda_i\Theta - \frac{(\omega^*)^2 - \omega^2}{2\alpha^{\max}}}{\omega^{\max}} \\ \text{where } \omega^* = \min(\omega^{\max}, \omega') \end{cases} \quad (8)$$

3. PROBLEM DEFINITION

In this work, we focus on the optimization problem: given a task system mixed with a set of periodic tasks and multiple dynamic AVR tasks with the common rotation source, how to optimize performance with a specific driving cycle by assigning the priority ordering, determining the switching times $\{\gamma_k\}$ of the engine configurations, and selecting the concrete transition speeds for the interval between the two consecutive switching times while satisfying the time constraint. The optimization problem can be formalized as the following.

Inputs: a set of periodic tasks with one timer, one single AVR task and a driving cycle. The AVR task has a set of the control implementations where each one has the various computational load.

To link the driving cycle and the transition speed, we need to preprocess the driving cycle to the engine speed cycle. Figure 3 is an example of the NEDC driving cycle [3], which lasts 1180 seconds. And this figure also illustrates the corresponding engine speed cycle with respect to the time, which is achieved by the simulation tool AVL/CRUISE [1] with the calibration of the real data.

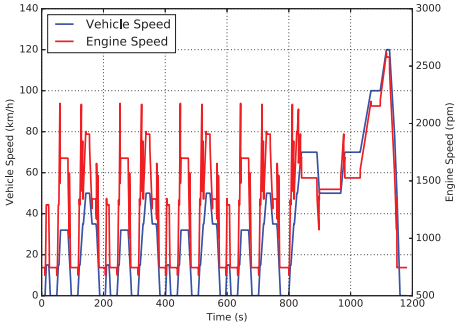

Figure 3: Vehicle and engine speeds in the new European driving cycle (NEDC).

Variables: a set of transition speeds and the fixed priority assignment.

Constraint: the schedulability constraint. The task system is schedulable if and only if there exists one fixed priority assignment under which any job instance released by the task system cannot miss the deadline. Thus, we can use Audsley's algorithm [2] to efficiently verify the schedulability.

Performance Metric: the overall performance metric based on a driving cycle can be expressed by

$$\mathcal{P}_{DC} = \sum \int_{T_{DC_i}^-}^{T_{DC_i}^+} PR_{\vec{\omega}_i}(\omega_{DC_i}(t))dt \quad (9)$$

where DC is a driving cycle and contains a series of sub driving cycles DC_i where $T_{DC_i}^-$ and $T_{DC_i}^+$ are the start and end time of the sub driving cycle, $\omega_{DC_i}(t)$ indicates the engine speed at time t of the sub driving cycle DC_i and $PR_{\vec{\omega}_i}(\omega)$ presents the performance rate versus the engine speed under the speed setting $\vec{\omega}_i$. The overall performance is the integration of the performance rate on

the time. In practice, the overall performance can be computed by accumulating the performance of each data point in the driving cycles, which equals to the product of the corresponding performance rate and the driving cycle sampling period. Note that the performance rate for the engine speed shall vary over the different transition speed settings. In general, function PR can be obtained by the theoretical analysis, the experimental simulation or the actual test bench.

Optimization Objective: $\max \mathcal{P}(\vec{\omega})$ or $\min \mathcal{P}(\vec{\omega})$

Outputs: the transition speeds and fixed priority assignment for each sub driving cycle.

3.1 Motivated Example

The similar example in [8] is used to illustrate the optimization problem and highlight why we need to dynamically set the transition speeds for the engine profile. Consider a tasks system consisting of 6 periodic tasks with the common activation timer and 82.5% total utilization shown in Table 2 and one AVR task similar to the AVR task example in Table **??** where the transition speeds are not unassigned. And the exponential coefficients of the implementations are $\{2, 3, 4, 5, 7, 10\}$. Finally, Table 3[1] shows the results when adopting the optimization algorithms proposed in [8].

	Task1	Task2	Task3	Task4
$C\ (us)$	1000	6500	10000	10000
$T = D\ (us)$	5000	20000	50000	100000
$U = C/T$	0.2	0.325	0.2	0.1

Table 2: Periodic tasks used in the running example.

Algo	ω^1	ω^2	ω^3	ω^4	ω^5	ω^6
UB [8]	6500	4238	3582	2949	1801	1098
BB [8]	6500	4228	3530	2771	1789	1040
BS [8]	6500	4235	3399	2777	1789	1049

Table 3: The optimized results for the motivated example.

However, these results cannot be used in the actual driving trip. Figure 4 shows the performance results for this example when considering the constant engine speed. For example, when the engine has the constant engine speed of 2800 RPM, the optimized results derived by **BB** and **BS** has the performance loss of 20% compared to the case with the transition setting $\{6500, *, *, 2949, *, *\}$ (where there are only two transition modes 1 and 4).

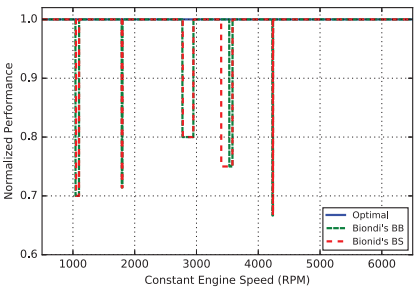

Figure 4: Normalized Performance with the constant speed.

4. GENERAL OPTIMIZATION PROCEDURE

[1]Here the acceleration/deceleration is not assumed to be constant.

4.1 Pre-Processing

4.2 Determining the Dynamic Engine Configurations

4.2.1 Each interval with single vehicle speed data

4.2.2 Each interval with a number of vehicle speed data

1: **procedure** OPTIMIZATIONALGORITHM($\Gamma = \{\tau_i\}, \vec{C}, DC$)
2: $\vec{\omega}_{ub} \leftarrow$ COMPUTEUBS(Γ,$\vec{C}$);
3: $M \leftarrow \vec{C}.size()$;
4: $k \leftarrow 0$;
5: $\mathcal{M}_{prev} \leftarrow \emptyset$;
6: **for** each $es \in DC$ **do**
7: $j \leftarrow \max_{m=j,\ldots,1}\{m|\omega^m_{ub} \geq es\}$;
8: **if** $\mathcal{M}_{prev} = \emptyset$ **then**
9: $\mathcal{M}_k \leftarrow \{(C^1, \omega^{\max}), (C^j, es)\}$;
10: $\mathcal{M}_{prev} \leftarrow \mathcal{M}_k$;
11: $k \leftarrow k + 1$;
12: **continue**;
13: **end if**
14: **for** $m = j, \ldots, M$ **do**
15: $\mathcal{M}_{curr} \leftarrow \{(C^1, \omega^{\max}), (C^m, es)\}$;
16: **if** SCHEDULABLE(Γ,$\mathcal{M}_{prev}$, $\mathcal{M}_{curr}$, $T(es)$) **then**
17: $\mathcal{M}_k \leftarrow \mathcal{M}_{curr}$;
18: $\mathcal{M}_{prev} \leftarrow \mathcal{M}_{curr}$;
19: **break**;
20: **end if**
21: **end for**
22: $k \leftarrow k + 1$;
23: **end for**
24: **end procedure**

Figure 5: Pseudo-code for our optimization procedure.

Figure **??** illustrates the pseudo code for the optimization procedure. At line **??** calculate the upper bound speeds in the similar way in [8] but with the **NO2** analysis. Since the **NO2** analysis is necessary and fast, the upper bound speeds shall be fast computed and over-estimated compared to the way using the exact analysis.

In the while loop (line -**??**), we iteratively use the necessary-only analyses in the order of **NO2** and **NO1** to perform the **ModifiedBSAlgo** algorithm. And if the task system configured with the current speeds is schedulable (checky by the **LUB** analysis), then they must be schedulable since the LUB analysis is sufficient (line **??**-**??** and **??**-**??**).

Note after the while loop it is possible that the task system is not schedulable under the current transition speeds $\vec{\omega}$. Therefore, at line **??** we use the exact analysis to do the **ModifiedBSAlgo2** algorithm, which is same to **ModifiedBSAlgo** but without the local search. In our experiments, the accuracy of the **NO1** analysis is much close to the **exact** analysis and as a result the verification times in line **??** are marginal (almost one time).

5. IMPROVEMENT

5.1 Existed Response-Time Analysis

5.2 Fast Response-Time Analyses

5.2.1 Fast Necessary-Only Response-Time Analyses

5.2.2 Fast Sufficient-Only Response-Time Analyses

5.3 Improved Optimization Procedure

6. EXPERIMENTAL EVALUATION

In this section, we study the accuracy and analysis runtime of the proposed methods in this work over the randomly generated task sets consisting of a few of periodic tasks and one single AVR task. These experiments are implemented in the C++ language running on a machine with an Intel Core i7 3.4GHz CPU.

6.1 Experimental Setting

In the experiments, we use the similar configuration as [7, 5, 8]. The engine speed is restricted between 500 and 6500 RPM (i.e., $\omega^{\min} = 500$ and $\omega^{\max} = 6500$) and the maximum acceleration/deceleration is $1.62\ 10^{-4}$ rev/msec^2 such that we only need 35 revolutions to accelerate/decelerate the engine between the minimum and maximum speeds.

The random task systems are generated as follows.

- **Periodic task set $\{\tau_i\}$:** Given the total utilization $U_P = \sum U_i$ and task number n ($n = 5$ in our experiments), we generate a set of periodic tasks where the period of each one is randomly selected from the set $\{5, 10, 20, 50, 80, 100\}ms$ and each utilization is calculated by the UUnifast algorithm [4].
- **One AVR task τ^*:** The mode number M is fixed as 6 in our experiments. The base wcets of the control implementations are randomly selected from a uniformly distributed set between $[100, 1000]$ with the minimum step 100 and are monotonically increasing with the mode index. The actual wcets are scaled by a factor s which results in different workload for the AVR task.

6.2 Random Tested Systems with Small Size

6.3 Real-World Benchmark

6.4 An Automative Case Study

7. 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.

8. REFERENCES

[1] Avl. http://www.avl.com/. Accessed: 2017-02-27.

[2] N. C. Audsley. On priority assignment in fixed priority scheduling. *Information Processing Letters*, 79(1):39–44, 2001.

[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. C. Buttazzo. Measuring the performance of schedulability tests. *Real-Time Systems*, 30(1-2):129–154, 2005.

[5] A. Biondi, G. Buttazzo, and S. Simoncelli. Feasibility analysis of engine control tasks under edf scheduling. In *Real-Time Systems (ECRTS), 2015 27th Euromicro Conference on*, pages 139–148. IEEE, 2015.

[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 *the 26th Euromicro Conference on Real-Time Systems (ECRTS)*, pages 165–174. IEEE, 2014.

[7] A. Biondi, M. D. Natale, and G. Buttazzo. Response-time analysis for real-time tasks in engine control applications. In *the 6th International Conference on Cyber-Physical Systems (ICCPS)*, pages 120–129. ACM, 2015.

[8] A. Biondi, M. D. Natale, and G. Buttazzo. Performance-driven design of engine control tasks. In *the 7th International Conference on Cyber-Physical Systems (ICCPS)*, pages 1–10. IEEE, 2016.

[9] 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.

[10] M. Mohaqeqi, J. Abdullah, P. Ekberg, and W. Yi. Refinement of Workload Models for Engine Controllers by State Space Partitioning. In M. Bertogna, editor, *29th Euromicro Conference on Real-Time Systems (ECRTS 2017)*, volume 76 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 11:1–11:22, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[11] V. Schwarzer and R. Ghorbani. Drive cycle generation for design optimization of electric vehicles. *IEEE Transactions on vehicular technology*, 62(1):89–97, 2013.