Hindawi Publishing Corporation
Mobile Information Systems
Volume 2016, Article ID 3414816, 11 pages
http://dx.doi.org/10.1155/2016/3414816

Research Article

Control-Scheduling Codesign Exploiting Trade-Off between Task Periods and Deadlines

Hyun-Jun Cha,[1] Woo-Hyuk Jeong,[2] and Jong-Chan Kim[1]

[1]*Graduate School of Automotive Engineering, Kookmin University, 77 Jeongneung-ro, Seongbuk-gu, Seoul 02707, Republic of Korea*
[2]*Department of Computer Science, Kookmin University, 77 Jeongneung-ro, Seongbuk-gu, Seoul 02707, Republic of Korea*

Correspondence should be addressed to Jong-Chan Kim; jongchank@kookmin.ac.kr

Received 1 January 2016; Revised 22 March 2016; Accepted 27 March 2016

Academic Editor: Qixin Wang

A control task's performance heavily depends on its sampling frequency and sensing-to-actuation delay. More frequent sampling, that is, shorter period, improves the control performance. Similarly, shorter delay also has a positive effect. Moreover, schedulability is also a function of periods and deadlines. By taking into account the control performance and schedulability at the same time, this paper defines a period and deadline selection problem for fixed-priority systems. Our problem is to find the optimal periods and deadlines for given tasks that maximize the overall system performance. As our solution, this paper presents a novel heuristic algorithm that finds a high-quality suboptimal solution with very low complexity, which makes the algorithm practically applicable to large size task sets.

1. Introduction

In a cyberphysical system (CPS), real-time control systems monitor and control physical systems (i.e., target plants) with precise control performance requirements and tight resource constraints. When designing such a system, two different approaches can be applied. First, the traditional approach separates the *control design phase* and *implementation phase* such that scheduling parameters such as sampling rates are determined in the control design phase without considering the scheduling issue. The second approach, known as *control-scheduling codesign*, takes into account both the control performance and task scheduling simultaneously for the purpose of enhancing control performance with limited resources [1–4]. This paper advocates the second approach when designing a CPS.

Performance of a real-time control system in a CPS depends on not only its functional correctness but also its scheduling parameters such as *sampling frequency*. With more frequent sensing and actuation, more accurate control results can be obtained [5]. Certainly, this performance enhancement is at the cost of increased computing demands. Another important but often ignored timing property is the delay between sensing and actuation, that is, *input-output delay*. Since a shorter delay means more recent sensing data has been used to produce the actuation value, it can provide higher control performance [5]. Then, one interesting observation is that a task's period, that is, inverse of sampling frequency, can be lengthened without hurting the control performance if we can somehow reduce the delay. To maintain a task's delay within a certain range, the desired maximum delay should be used as the relative deadline in the schedulability check. If the schedulability check passes, the task's input-output delay is guaranteed less than the relative deadline. One more timing attribute we have to discuss is *jitter*, which is the amount of uncertain variation of sampling time or input-output delay, called *sampling jitter* and *input-output jitter*, respectively. Generally, large jitter has negative effects on the control performance even though the effect is not that significant as the input-output delay [6]. Even regarding jitters, a shorter relative deadline also gives tighter upper bounds of the sampling jitter and input-output jitter such that a better control result can be produced [6]. As a result, the control performance can be enhanced by either way of shorter periods or shorter deadlines, and periods and deadlines have a *trade-off relation* in terms of control performance.

Besides control performance, schedulability is also heavily affected by periods and deadlines of the tasks. Generally speaking, longer periods and longer deadlines both make the system more schedulable, however, at the cost of a reduced control performance. In other words, a better control performance can be obtained with a lower chance of being schedulable. Moreover, similar to the control performance case, periods and deadlines are mutually tradable to maintain schedulability. Therefore, it is important to select proper periods and deadlines which satisfy both the control performance and the schedulability. For this control-scheduling codesign issue, an optimization problem can be formulated which finds the best feasible periods and deadlines for given tasks that maximize the overall system performance.

In the literature, with a similar motivation, *period selection problem* has been extensively studied [7–10] for both dynamic and fixed-priority scheduling algorithms. However, *period and deadline selection problem*, which this paper is dealing with, has gathered relatively little attention and only the dynamic-priority case has been studied [6, 11]. With this motivation, targeting fixed-priority systems, this paper proposes a novel task set synthesis algorithm that finds the proper periods and deadlines which maximize the overall system performance while guaranteeing the system schedulability. Since, even with a small number of tasks, finding the optimal solution is intractable due to the huge solution space to be searched, our algorithm is basically structured as a search-based heuristic algorithm. As will be shown in Section 6, our algorithm has a linear complexity and finds a high-quality suboptimal solution even with a large task set.

For the quantitative analysis of control performance with varying periods and deadlines, we also conduct a measurement study with an automotive control application. The measured control performance of the task is defined as a nonlinear and nonconvex function of period and deadline. This function is used as an input to our heuristic algorithm.

This paper's contribution can be summarized as follows:

(i) We identify and demonstrate the trade-off relation between period and deadline in terms of control performance through actual experimental studies with an automotive control application.

(ii) Exploiting the above trade-off relation, a novel task set synthesis algorithm is proposed, which heuristically finds near-optimal feasible (period and delay) combinations maximizing the overall control performance.

The rest of our paper is organized as follows. The next section briefly explains related work. Section 3 presents brief background knowledge and formally describes our problem. In Section 4, the trade-off relation of control performance is formally described. Section 5 presents our heuristic algorithm for the period and deadline selection problem. The experimental results are presented in Section 6. Finally, Section 7 concludes this paper.

2. Related Work

Control-scheduling codesign problem has been extensively studied in the literature. Seto et al. [7] first defined the period selection problem assuming that the control performance can be expressed as an exponential decay function of the sampling period and the tasks are scheduled using dynamic-priority methods. The problem is extended to fixed-priority systems by Seto et al. [8] by finding the finite set of feasible period ranges using a branch and bound-based integer programming method. In their work, the cost function is assumed to be a monotonically increasing function of task period. Later, Bini and Di Natale [9] proposed a faster algorithm that finds a suboptimal period assignment, which can be used for a task set of practical size that was intractable by previous methods due to its high computing demands. Recently, Du et al. [12] proposed an analytical solution using the method of Lagrange multipliers and an online algorithm for the overloaded situation.

The common assumption of the above researches regarding the period selection problem is that the control performance is only affected by the sampling rate, that is, task period, of the controller. However, the delay between sensing and actuation also has a significant effect on the control performance. With this motivation, Bini and Cervin [10] incorporated each task's sensing to actuation delay into the cost function. In their work, in order to find the optimal period assignment, cost functions are approximated as linear functions of period and delay, and the delay is also approximated assuming the fluid model scheduler. Through these approximations, they proposed an analytical solution.

Wu et al. [6] further enhanced the algorithm by finding task periods and deadlines altogether for EDF scheduled systems. As a result, the problem had become a period and deadline selection problem. They showed that, by regulating relative deadlines of tasks, we can upper-limit the amount of delays and jitter each task can experience. In their work, the cost function is assumed to be a nonlinear function which increases in both period and deadline of tasks. A two-step approach was presented which first fixes periods and tries to minimize deadlines using unused resources. Recently, Tan et al. [11] proposed a new algorithm which simultaneously adjusts periods and deadlines assuming EDF scheduling and LGQ controller tasks. They showed that the new algorithm is more robust with different workloads than the previous method.

Despite the above researches, however, compared to the period selection problem, the period and deadline selection problem has gained less attention even though it has more flexibility to enhance control performance with scarce resources. Moreover, only EDF scheduling is considered in the period and deadline selection problem due to its ease of schedulability analysis, though the fixed-priority scheduling is more commonly used in the practice. On the contrary, this paper is dealing with the period and deadline selection problem under the fixed-priority scheduling.

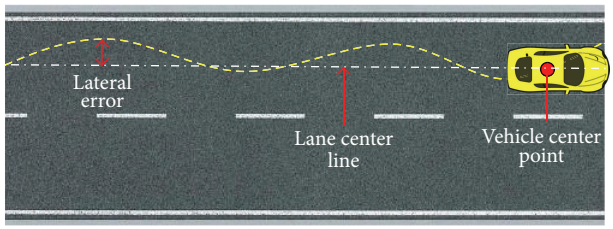

Figure 1: Lane keeping assist system. Its system error is defined as the lateral distance between the vehicle center point and the lane center line.

3. Background and Problem Description

3.1. System Model. This paper considers a system with n independent periodic real-time tasks $\{\tau_1, \tau_2, \ldots, \tau_n\}$, which control n different plants, respectively. The tasks are scheduled by the fixed-priority scheduler and the priorities are assigned according to the Deadline Monotonic (DM) order. Each τ_i is characterized by the following scheduling parameters:

(i) C_i: the worst-case execution time (WCET).

(ii) T_i: the sampling period.

(iii) D_i: the relative deadline.

From the above, C_i is a known parameter decided by the control code, whereas T_i and D_i are operational parameters which can be controlled by system designers. Regarding deadlines, this paper considers only *constrained deadlines* where D_i is always less than or equal to T_i. During system execution, each τ_i generates infinite sequence of periodic jobs $\tau_{i,1}, \tau_{i,2}, \ldots$ with its period T_i, which controls its corresponding target plant by (i) sensing the state of the plant, (ii) calculating the actuation values, and (iii) actuating the plant.

3.2. Control Performance as a Function of Period and Deadline. When defining the performance of a controller, various metrics can be used, such as transient response time and steady-state accuracy [6, 13]. In some cases, even the energy consumption can be a control performance metric [7]. Among the various control performance metrics, this paper chooses the system error as our optimization target. System error is defined as the difference between the desired state and the actual state of the plant [6]. This can be thought of as how well the plant is acting following the controller's intention.

More specifically, let us take the lane keeping assist system (LKAS) as an example. In a modern vehicle, LKAS controls the steering angle such that the vehicle is able to follow the center of its lane. Then, the system error can be defined as the lateral error between the center of the vehicle and the center of the lane, which is illustrated in Figure 1. Since this system error also varies along with time t, we further define the worst-case system error as the largest lateral error the vehicle can experience during its driving. More interested readers are referred to [14].

Following the notation in [6], each task's control performance is defined as a function of its period and deadline, which is denoted by

$$J_i\left(T_i, D_i\right). \tag{1}$$

Generally, $J_i(T_i, D_i)$ is defined as a nonlinear cost function which increases in both T_i and D_i; that is, if period or deadline increases, system error always increases. The intuition behind this assumption will be discussed in Section 4. We assume that $J_i(T_i, D_i)$ is not continuous but discrete in both T_i and D_i, which are also constrained within $[T_i^{\min}, T_i^{\max}]$ and $[D_i^{\min}, D_i^{\max}]$, respectively.

Besides each task's system error, the overall system error is denoted by

$$J\left(T, D\right), \tag{2}$$

where T is the vector of T_i's and D is the vector of D_i's. For the notational simplicity, $J(T, D)$ is shortened to J as in the following:

$$J = \sum_{1 \le i \le n} w_i J_i\left(T_i, D_i\right), \tag{3}$$

where J is defined as a weighted sum of $J_i(T_i, D_i)$'s and w_i is a user-defined weight constant for the purpose of normalizing each $J_i(T_i, D_i)$ to a desired range. Now, the system's overall system error can be obtained by giving every task's period and deadline.

3.3. Problem Description. Assuming the above concepts and notations, this subsection describes our period and deadline selection problem, which can be formally defined as follows.

Problem Description. For a given task set $\{\tau_1, \tau_2, \ldots, \tau_n\}$, each τ_i's C_i and $J_i(T_i, D_i)$ are given a priority. Then, our problem is to find the optimal $T = (T_1, T_2, \ldots, T_n)$ and $D = (D_1, D_2, \ldots, D_n)$ such that the overall system error $J(T, D)$ is minimized while guaranteeing every τ_i's schedulability.

Since it is assumed that $J_i(T_i, D_i)$ is not continuous, we do not try to make an analytical solution for our optimization problem. Instead, we formulate our problem as a combinatorial optimization problem. Figure 2 shows a graphical representation of an example $J_i(T_i, D_i)$ with 10 discrete (T_i, D_i) combinations. Note that since we only consider constrained deadlines, the left upper triangular matrix is not considered at all.

4. How Scheduling Affects Control Performance

This section deals with the rationale behind the definition of $J_i(T_i, D_i)$, which is introduced in Section 3.2. It is claimed in many literatures that the control performance of a task can be defined as a function of the task period and deadline [6, 10, 11]. If the control system is composed of only a single periodic task, it should be strictly periodic and the input-output delay is simply bounded by C_i. Thus, the control performance should be defined as a function of T_i and C_i. However, when

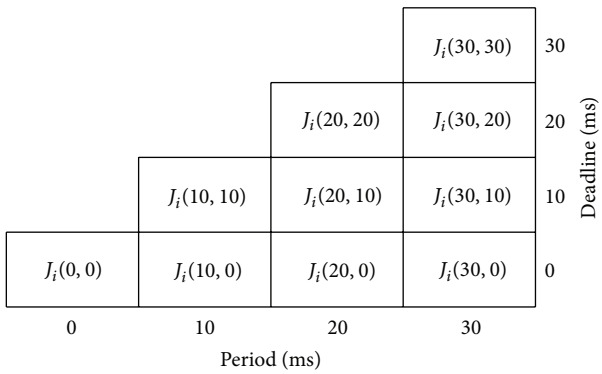

Figure 2: A graphical representation of an example $J_i(T_i, D_i)$ function with its period varying from 0 ms to 30 ms and its deadline also varying from 0 ms to 30 ms. The granularity of periods and deadlines is 10 ms.

multiple control tasks are implemented and scheduled on a shared processor, their execution is also subject to the following scheduling effects [6]:

(i) *Sampling Delay*. Although a job is released at a certain time point, the actual beginning of the job can be delayed due to the execution of higher-priority jobs. The time distance between the release time and the actual beginning of a job is defined as a sampling delay.

(ii) *Input-Output Delay*. The time distance between the beginning and the completion of a job is defined as an input-output delay, which is composed of the actual execution time of the job which is bounded by C_i and the blocking time by higher-priority jobs.

(iii) *Sampling Jitter*. As a task produces consecutive jobs, its sampling delay is not constant but varies according to the scheduling pattern each job experiences. The difference between the minimum and maximum sampling delay is defined as the sampling jitter of the task.

(iv) *Input-Output Jitter*. The difference between the minimum and maximum input-output delay of a task is defined as the input-output jitter of the task.

By the above scheduling effects, the actual execution of a control task is not strictly periodic and the input-output delay significantly varies due to different scheduling scenarios. Thus, the above scheduling effects as well as the control period collectively affect the control performance. Note that, for each task, the only tunable scheduling parameters are T_i and D_i in our system model. The control period T_i itself has a significant effect on the control performance since it determines the control frequency. Besides, for the above four scheduling effects, Wu et al. [6] analyzed the worst-case scenarios as in the following under the condition that every task is schedulable:

(i) Sampling delay is bounded by $D_i - C_i$, which happens when the beginning of a job is delayed as long as possible without hurting the schedulability, and the job executes without any interference by higher-priority jobs once it begins.

(ii) Input-output delay is bounded by D_i, which happens when a job begins right after it is released and finishes right before the job's deadline.

(iii) Sampling jitter is bounded by $D_i - C_i$. The minimum possible sampling delay is 0 and the maximum possible sampling delay is $D_i - C_i$. Thus, the difference between the maximum and minimum sampling delay is $D_i - C_i$.

(iv) Input-output jitter is bounded by $D_i - C_i^b$, where C_i^b is the best-case execution time which can be simply assumed to be zero when not available. The minimum possible input-output delay is C_i^b and the maximum possible input-output delay is D_i. Thus, the difference between the maximum and minimum input-output delay is $D_i - C_i^b$.

The analysis above shows that a shorter deadline makes shorter sampling delay and shorter input-output delay. Even the jitters can be controlled by a shorter deadline. The implication of the above analysis is that a better control performance can be obtained by shortening D_i as well as T_i [6]. From the above observation, we can conclude that the control performance generally increases as T_i or D_i decreases. In other words, $J_i(T_i, D_i)$, the system error, is a monotonically increasing function in both T_i and D_i. Note that $J_i(T_i, D_i)$ also reflects the effect of jitters as well as delays and sampling frequency. In order to account for the above scheduling effects when estimating the control performance, Jitterbug [15], which is a MATLAB toolbox, can be used to analyze the cost of delay and jitter in terms of control performance. Besides, TrueTime [16] provides a simulation environment which facilitates cosimulation of controller task execution and real-time scheduling.

Although analysis and simulation methods are useful when estimating control performance, this paper proposes a different approach when finding $J_i(T_i, D_i)$. As will be further explained in Section 6.1, a measurement environment is developed for an automotive control application. In practice, T_i and D_i cannot be an arbitrary number but should be chosen from a number of predefined candidate parameters the software platform provides. Thus, it is practically feasible to measure the system errors for each T_i and D_i combination for each control task by arbitrarily making the worst-case scenarios. Using these discrete functions $J_i(T_i, D_i)$'s, the optimization algorithm in Section 5 can be used to find optimal T and D that minimize $J(T, D)$.

5. Task Set Synthesis Algorithm

In order to find T and D with the minimum $J(T, D)$, the simplest solution is to explore the entire solution space for every (T_i, D_i) combination checking the schedulability and calculating the overall system error. This exhaustive search method, however, has a too high complexity such that it is not applicable to a task set even with a small number of

tasks like five or six tasks. The computational feasibility of the exhaustive search algorithm will be further discussed in Section 6.2. Instead, this section proposes an alternative heuristic algorithm that finds a suboptimal result, however, with a very low computational complexity. Even with this low complexity, our solution can find a very-high-quality solution for very large task sets, which will be shown later in Section 6.2.

For the ease of explanation, let us define the following function:

$$\text{Schedulability}\,(T, D, C)\,, \tag{4}$$

where T and D are vectors of chosen periods and deadlines. C is a vector of each task's C_i's. It is assumed that C is a constant vector. Inside this function, tasks are sorted according to the DM order where the task with the shortest D_i gets index 1 and the task with the longest D_i gets index n. Then, following Audsley et al. [17], the exact schedulability check is performed. For that, the following recursive equation computes the worst-case response time R_i of each τ_i:

$$R_i^{k+1} = C_i + \sum_{1 \le m < i} \left\lceil \frac{R_i^k}{T_m} \right\rceil \cdot C_m, \tag{5}$$

where $R_i^0 = C_i$. The recursive equation continues until $R_i^k = R_i^{k+1}$ and the converged value is taken as the final R_i. Then, since we know every R_i, we can simply check each τ_i's schedulability by comparing R_i with D_i. If every R_i is less than D_i for $1 \le i \le n$, the system is schedulable and every τ_i's input-output delay is guaranteed under D_i. From the result of the schedulability check, Schedulability(T, D, C) returns either of the following values:

(i) *True*: if the system is schedulable.

(ii) *False*: if the system is not schedulable.

This schedulability check function is used inside the outer loop of our heuristic search algorithm to check the feasibility of the chosen T and D.

In the beginning of our heuristic algorithm, the initial solution is set to

$$\{\tau_1\,(0,0)\,, \tau_2\,(0,0)\,, \ldots, \tau_n\,(0,0)\}\,; \tag{6}$$

that is, all the periods and deadlines are equal to zero. Certainly, this initial solution has the lowest possible $J(T, D)$ but is definitely not schedulable. Then, our heuristic algorithm iteratively selects (i) the task and (ii) the direction to move the chosen task until Schedulability(T, D, C) = True. Our rule of thumb for selecting the proper task is to choose the task with the lowest $J_i(T_i, D_i)$ for the purpose of preventing a certain task from moving too quickly to the higher $J_i(T_i, D_i)$. For choosing the moving direction for each iteration, the basic idea is to choose the direction with the lower slope in order to minimize the resulting $J_i(T_i, D_i)$ after the move.

When applying the above basic idea, however, it can suffer from the following worst-case scenario. Starting a new iteration, the algorithm chooses τ_i as the task to be moved. By looking at τ_i's current position in $J(T_i, D_i)$, the right direction has the lower slope compared to the upper direction. Naturally, our algorithm moves τ_i to the right direction. However, imagine that even though the upper direction requires higher slope, the task set can be schedulable immediately after moving τ_i to the upper direction. If this case happens repeatedly, τ_i will move to the right direction too many times, but still making the system unschedulable.

To prevent such scenarios, we slightly tune the algorithm by looking at the system schedulability as well as $J(T_i, D_i)$'s slope when determining the moving direction. Figure 3 shows the four cases our algorithm should consider when the current location of the task is $\tau_i(10, 0)$:

(i) Figure 3(a) shows a case where both directions make the system schedulable. In that case, it is desirable to choose the direction with the lower slope.

(ii) Figure 3(b) shows another case where both directions are not schedulable. Then, our choice is also to choose the direction with the lower slope.

(iii) Figure 3(c) shows a case where the lower slope direction is schedulable, but the higher slope direction is not schedulable. Then, the choice is to take the lower slope direction, which makes the system schedulable immediately.

(iv) Figure 3(d) shows a case where the lower slope is not schedulable, but the higher slope is schedulable. In this case, it is not possible to decide the correct direction from the current information available. If we take the upper direction, the system will be immediately schedulable with $J_i(T_i, D_i) = 0.07$. However, if we move to the right direction two times, it will also make the system schedulable with even lower $J_i(T_i, D_i) = 0.06$. One interesting observation is that the cells in the right-hand side of $\tau_i(10, 10)$ make $J_i(T_i, D_i)$ always larger than 0.07. Thus, if there is a better solution compared to $\tau_i(10, 10)$, it must be among the cells in the right-hand side of the current location, that is, $\tau(10, 0)$. Then, our quick fix is that, upon meeting this condition, every cell in the lower slope direction is quickly visited to compare the resulting $J_i(T_i, D_i)$ with $J_i(T_i, D_i)$ when taking the high slope direction to choose the better direction.

Procedure 1 shows the pseudocode of our heuristic iterative search algorithm. After positioning each τ_i at the initial solution, the while loop iteratively chooses the next moving τ_i and the direction to move considering the four cases in Figure 3 until the system becomes schedulable. Then, using break, the while loop is terminated and the output is finally decided.

6. Experiments

6.1. Control Performance Measurement Study. In this subsection, the experimental results for the control performance measurement study are presented. First, we explain how the measurement environment is designed and implemented. Then, the actual measurement data is presented for an

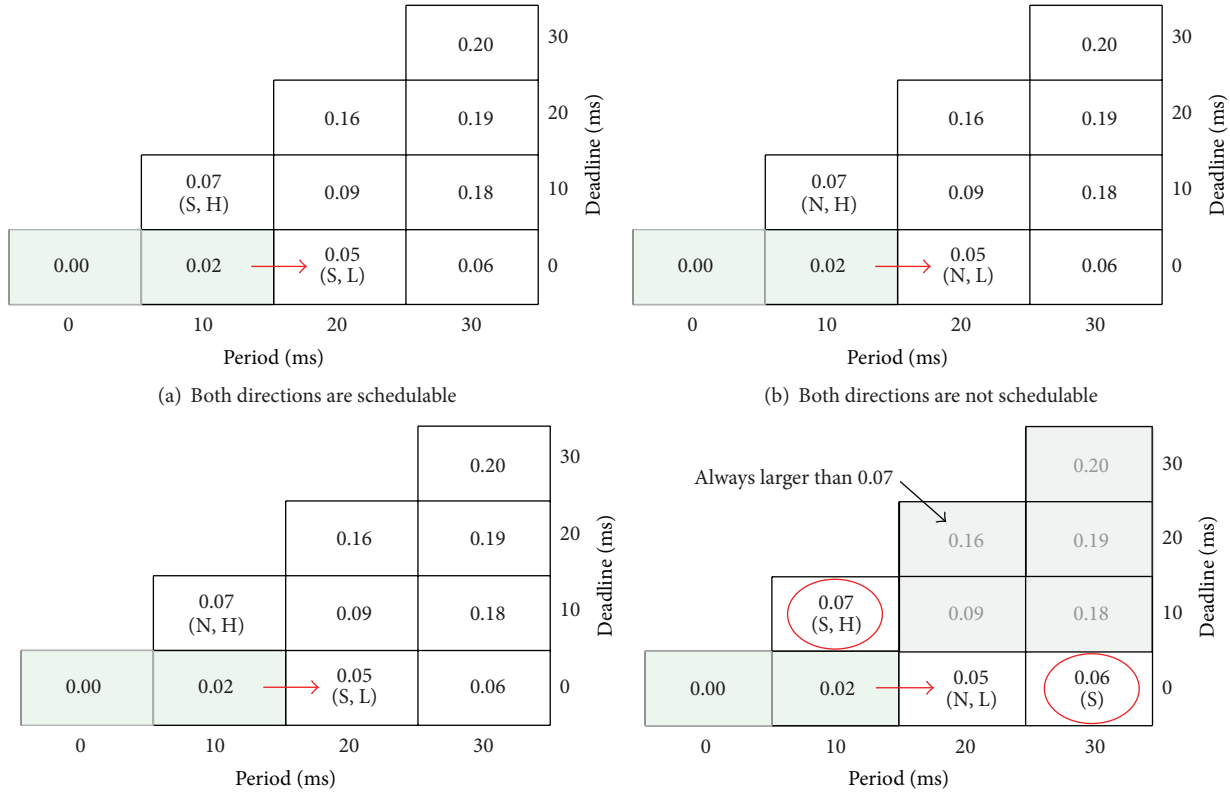

FIGURE 3: The four cases to be considered by our heuristic algorithm. S and N mean schedulable and not schedulable, respectively. H and L mean higher slope and lower slope, respectively.

example control application. For the measurement study, as the target control application, we use LKAS, in which the system error is defined as the maximum lateral error the vehicle experiences during the measurement.

Figure 4 shows the measurement environment, which consists of a vehicle dynamics simulator and two automotive electronic control units (ECUs) connected by a controller area network (CAN) bus. Inside the ECUs, control application codes are deployed upon our specially tuned real-time operating system (RTOS), which is a modified version of Erika Enterprise [18]. We specifically tuned the RTOS such that the worst-case job release and delay pattern always happens to simulate the worst-case scenario the vehicle can experience. Among the two ECUs, the first one contains the LKAS code and the second one has the cruise control (CC) code. The CC algorithm controls the throttle and brake to make the vehicle run at a predefined constant speed. In the following, each component is explained in more detail:

(i) *Vehicle Simulator*. For simulating the real-time dynamics of a vehicle, we use a modified version of the open source TORCS [19] simulator on a PC with Ubuntu-14.04. TORCS has a precise vehicle dynamics engine and 3D visualization features.

(ii) *ECU (LKAS)*. This ECU contains the LKAS code, which receives sensing data (e.g., vehicle speed, steering angle, yaw, and lateral distance) from the vehicle simulator and sends out the steering actuation values. Infineon TC1797 MCU [20] is used with 180 MHz CPU, 4 MB Flash, and 1 MB RAM.

(iii) *ECU (Cruise Control)*. This ECU actuates the throttle and brake of the vehicle simulator to keep the vehicle at a constant speed. Since we are only interested in the LKAS performance, we just set this ECU to maintain 100 km/h speed throughout the experiment with 1 ms period. Infineon TC1796 MCU [21] with 150 MHz CPU, 2 MB Flash, and 512 KB RAM is used.

(iv) *Operation and Measurement Console*. We made a control panel using LabVIEW [22], which can control the period and deadline of the ECUs by the operator person. Also, it can gather the resulting system error and visualize it using a real-time plotting screen.

(v) *CAN Bus Interfaces*. For the real-time communication between the vehicle simulator, ECUs, and the operation and measurement console, a 500 kbps CAN bus is used. For PCs, UBS-CAN interfaces [23] are used. For ECUs, its onboard controller is used.

(iv) *Human-Vehicle Interface*. Driving wheel, throttle, and brake are installed for manual driving. Logitech G25 model [24] is used for the interface.

Using the measurement environment, we actually measure the maximum system error as varying periods and

FindOptimalPhases:
Input: $\{\tau_1, \tau_2, \ldots, \tau_n\}$, $\tau_i = (C_i, J_i(T_i, D_i))$
Output: T and D
begin procedure
(1) Set each τ_i at $(0, 0)$
(2) **while** (1) **do**
(3) Choose τ_i with the lowest $J_i(T_i, D_i)$
(4) Check the schedulability for upper and right directions
(5) **if** Case in Figure 3(a) **then**
(6) Move τ_i to the lower slope direction
(7) **end if**
(8) **if** Case in Figure 3(b) **then**
(9) Move τ_i to the lower slope direction
(10) **end if**
(11) **if** Case in Figure 3(c) **then**
(12) Move τ_i to the lower slope direction
(13) **end if**
(14) **if** Case in Figure 3(d) **then**
(15) Visit every cell in the lower slope direction and compare J_i
(16) Move τ_i to the lower J_i direction
(17) **end if**
(18) **if** the system is schedulable **then**
(19) **break**
(20) **end if**
(21) **end while**
end procedure

Procedure 1: Procedure for finding optimal T and D.

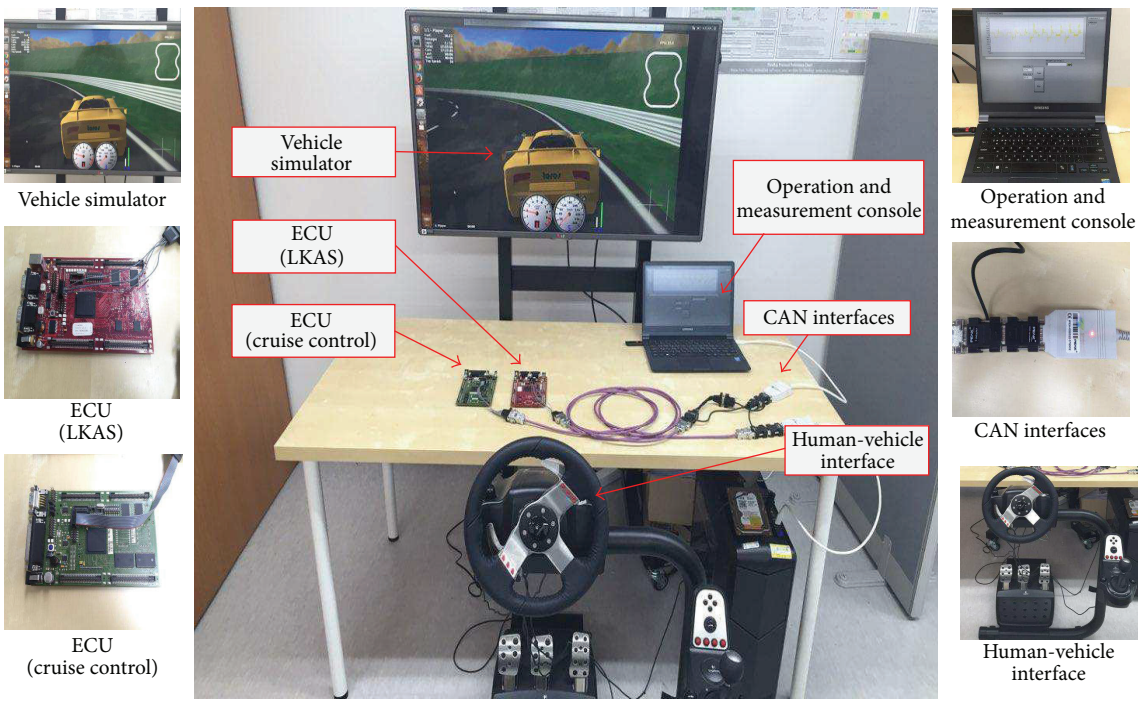

Figure 4: Measurement environment with vehicle dynamics simulator and automotive control ECUs.

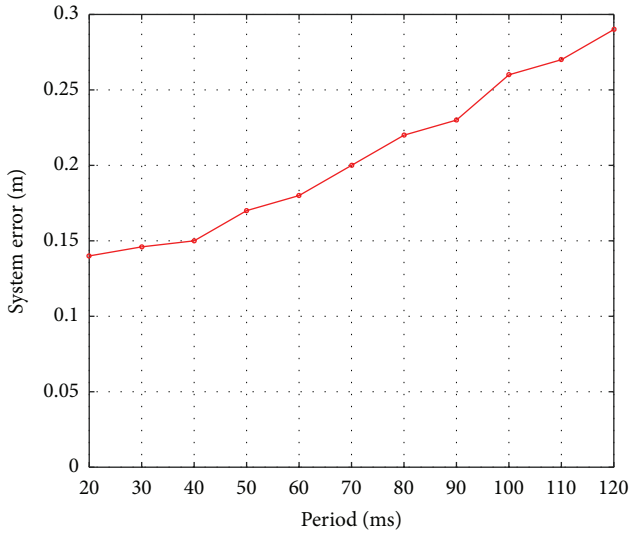

FIGURE 5: System error as varying periods with a fixed deadline.

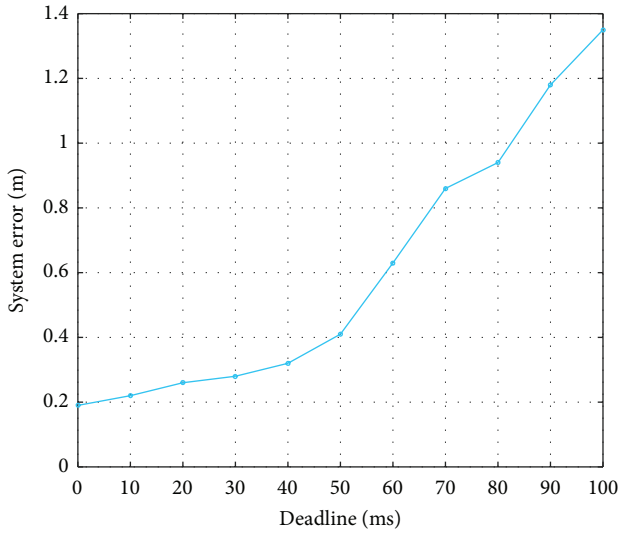

FIGURE 6: System error as varying deadlines with a fixed period.

delays. Period ranges within [0 ms, 120 ms] and deadline ranges from 0 ms to its period. The timing granularity is 10 ms for both period and delay. Figure 5 shows the system errors as varying periods with a fixed deadline at 20 ms. As shown in the figure, the system error monotonically increases as period increases, which means that larger periods have a negative impact on the control performance. Comparing the shortest period (20 ms) and the longest period (120 ms), the system error is almost doubled. Figure 6 shows a different configuration where the period is fixed at 100 ms and the delay is varying from 0 ms to its period, that is, 100 ms. By looking at the trends, we can conclude that larger deadlines have also a negative impact. Comparing Figures 5 and 6, the measured data shows that deadlines have a more significant effect on the performance than periods. Figure 7 shows the system errors as varying periods and deadlines in a 3D graph. The two axes on the floor are period and delay,

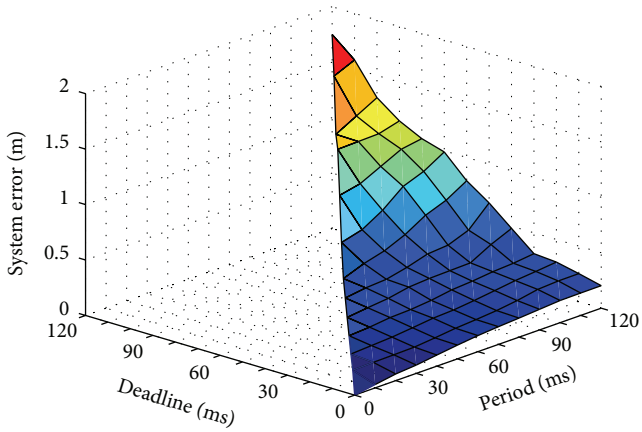

FIGURE 7: System error as varying periods and deadlines.

respectively. The vertical axis is the measured system error. From the figure, it is clearly shown that the system error is monotonically increasing in both period and delay and the delay has a more significant impact on the control performance compared to the period.

6.2. Evaluation of Our Proposed Algorithm. This subsection evaluates our heuristic algorithm in terms of optimality and computational feasibility with synthesized task sets. When generating task sets, for each task, the following were considered:

(i) C_i is an integer value which is randomly selected from the uniform distribution in the interval from 1 ms to 10 ms.

(ii) $J(T_i, D_i)$ is generated as a monotonically increasing function in T_i and D_i. The minimum and maximum of both T_i and D_i are 0 ms and 100 ms, respectively, with a granularity of 10 ms. Since we only assume the cases with $D_i \leq T_i$, a total of 66 values should be generated for each T_i and D_i pair, which are randomly chosen real values uniformly distributed in the interval from 0 to 1.

Figure 8 is an example task set $\{\tau_1, \tau_2, \tau_3\}$ with $n = 3$. For each of them, C_i is simply set to 10 ms. In the figure, note that the cells with $D_i > T_i$ are not generated since we only consider constrained deadlines. The figure also depicts how our heuristic algorithm iteratively finds the solution with an example. The initial solution is set to

$$\{\tau_1(0,0), \tau_2(0,0), \tau_3(0,0)\}, \quad (7)$$

where each tuple is (T_i, D_i) pair. Then, for each iteration, the task with the lowest $J_i(T_i, D_i)$ is chosen and the task is moved to the proper direction as explained in Section 5. Each circled number means the movement of the consecutive search iterations. The move continues until the system becomes schedulable. In this example, after 16 moves, the final solution is found, that is,

$$\{\tau_1(30,30), \tau_2(30,20), \tau_3(30,20)\}. \quad (8)$$

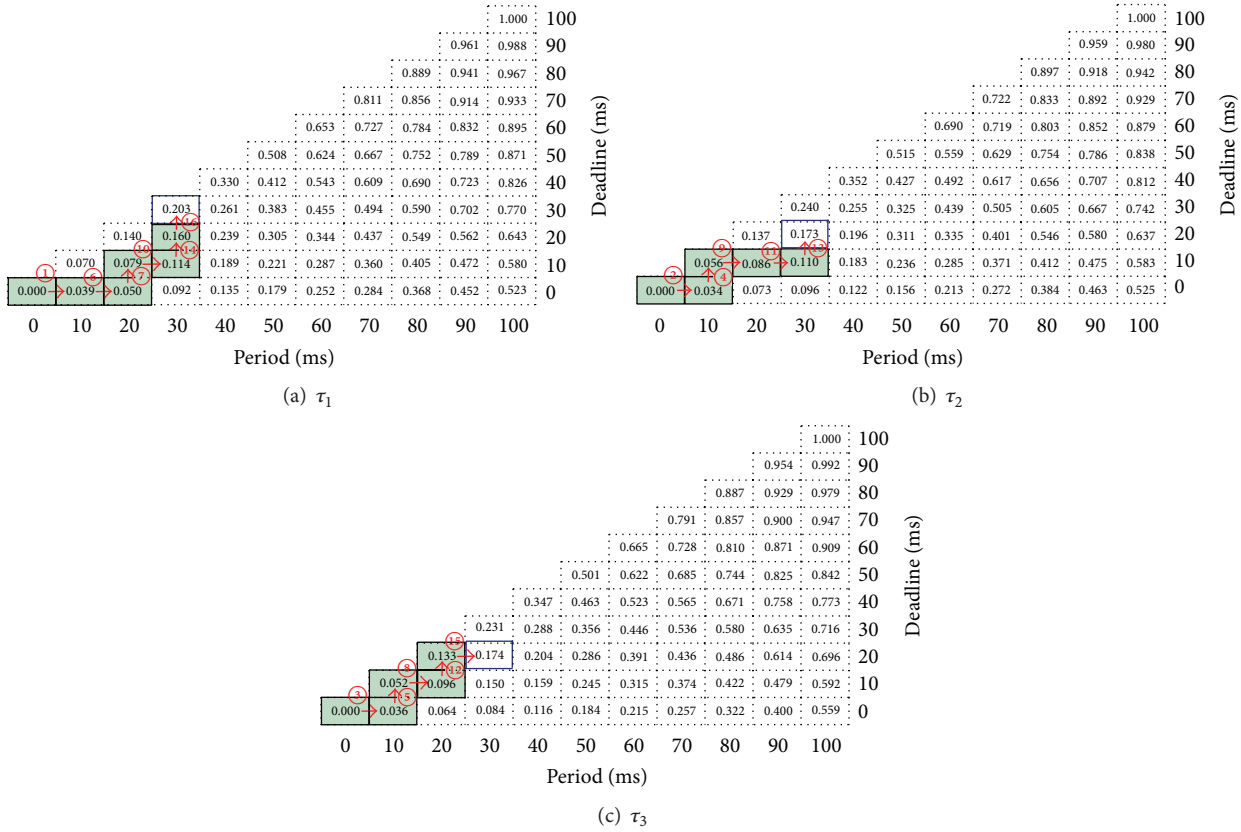

FIGURE 8: The process of heuristic algorithm.

At the final solution, $J(T, D) = 0.55 = 0.203 + 0.173 + 0.174$. Note that when deciding the overall system error, the weights w_i's are simply given 1 in the entire experiment.

With the above exemplified task set generation method and heuristic algorithm, we compare the following three approaches:

(i) Searching every possible combination of periods and deadlines checking the schedulability and overall system error. The result is optimal for the entire solution space. This approach is denoted by *Exhaustive*.

(ii) Searching only the combinations with $P_i = D_i$, which significantly reduces the solution space compared to *Exhaustive*. The result is only optimal for the solution space with implicit deadlines. This approach is denoted by *Implicit*.

(iii) Search guided by our heuristic algorithm as explained in Section 5. The result may not be optimal compared to *Exhaustive*. This approach is denoted by *Ours*.

Figure 9 shows the overall system error with the above three approaches. The number of tasks is varied from 1 to 6. For each experiment, 100 task sets are generated and the result is the average of 100 task sets. As shown in the figure, *Exhaustive* shows the best result even though the algorithm almost never ends when the number of tasks exceeds 4. The results for 5 and 6 are approximated using a curve fitting method with quadratic equations. Comparing *Implicit* and *Ours*, *Ours* wins for every six experiments. Though the difference is marginal when the number of tasks is small, note that the difference is increasing as the number of tasks increases.

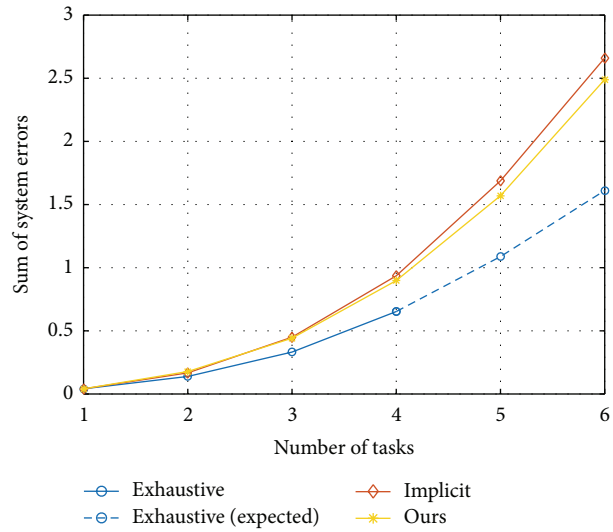

FIGURE 9: Comparison of our approach with optimal results.

Table 1 shows the required computing time for the three approaches when the number of tasks varies from 1 to 10.

Table 1: The required computing times for the three approaches with varying number of tasks.

Number of tasks	Exhaustive	Implicit	Ours
1	0.6835 ms	0.7885 ms	0.69 ms
2	13.684 ms	0.926 ms	0.88 ms
3	1305 ms	6 ms	0.75 ms
4	2 min	72 ms	0.74 ms
5	(3 hours)	986 ms	1.65 ms
6	(12 days)	1100 ms	2.44 ms
7	(1228 days)	(2 min)	16.19 ms
8	(3 years)	(26 min)	121 ms
9	(30577 years)	(5 hours)	609 ms
10	(290635 years)	(2 days)	1121 ms

The numbers in parenthesis are estimated values whereas the other numbers are actually measured. From the table, *Exhaustive* requires more than a year when the number of tasks is only 7. Even for *Implicit*, when the number of tasks is 10, which is relatively small in practice, the required computing time exceeds 2 days. Therefore, we can conclude that both *Exhaustive* and *Implicit* cannot be used as practical size task sets whereas *Ours* finds solutions even with larger task sets.

In order to prove that our heuristic algorithm produces a high-quality solution compared to other methods, we also compare *Ours* with the following two other heuristic algorithms:

(i) At each iteration, the approach chooses the moving direction with the higher slope of $J(T_i, D_i)$. This approach is denoted by *Higher*.

(ii) At each iteration, the approach chooses the moving direction with the lower slope of $J(T_i, D_i)$. This approach is denoted by *Lower*.

Note that *Ours* is an extension of *Lower*, which additionally considers the resulting schedulability as well as the slope when deciding the moving direction.

Figure 10 compares the performance of *Ours* with *Higher* and *Lower*. The result is the average of 100 synthesized task sets for each number of tasks from 1 to 6. As shown in the figure, *Ours* shows the best result compared to *Higher* and *Lower*. By comparing *Higher* and *Lower*, *Lower* shows a better result compared to *Higher*. The result first explains that taking the lower slope produces a better result than taking the higher slope. Meanwhile, *Ours* further enhances the performance by taking the schedulability into consideration as well as the slope at each iteration.

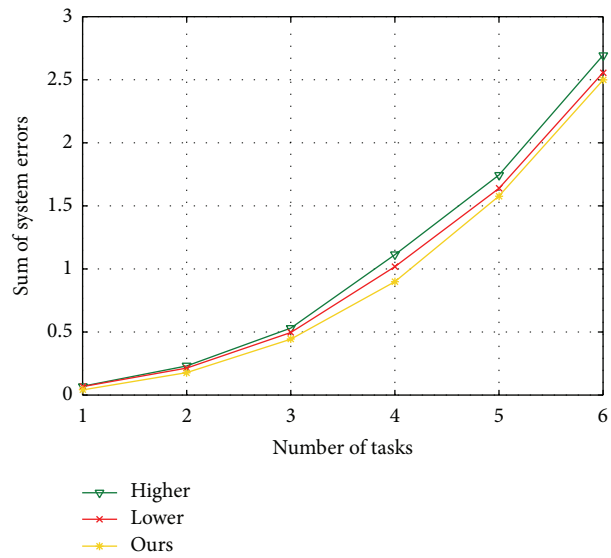

Figure 10: Comparison of our approach with other heuristic methods.

7. Conclusion

By exploiting the trade-off relation of task periods and deadlines, this paper proposes a novel task set synthesis algorithm for maximizing the overall system performance. For conducting a measurement study regarding the control performance, a simulation environment is developed, which can easily gather the performance variations of automotive control applications while applying various task periods and deadlines. Starting from the measured data, which confirms the trade-off relation, our problem is formally defined as a period and deadline selection problem. The input to our problem is each task's WCET and its measured control performance matrix with various periods and deadlines. For the scheduling, DM fixed-priority scheduling is assumed. Since it becomes quickly intractable to find the optimal solution with even relatively small number of tasks, this paper proposes a heuristic algorithm with a linear complexity that finds a high-quality suboptimal solution. Our heuristic algorithm is based on a gradient descent method with its initial solution at the smallest period and deadline for each task. Starting from the initial solution, our algorithm iteratively increases period or deadline one at a time until the task set is schedulable. For each iteration, the task and its moving direction are chosen comparing the control performance reductions.

In our future work, we consider a new system configuration where multiple implicit deadline periodic tasks collectively control a single plant. For such systems, a chain of tasks actually controls a plant from sensing to actuation. By controlling each task's sampling period, we have to indirectly control the sampling frequency and input-output delay the plant actually experiences.

Competing Interests

The authors declare that there are no competing interests regarding the publication of this paper.

References

[1] K.-E. Årzén, A. Cervin, J. Eker, and L. Sha, "An introduction to control and scheduling co-design," in *Proceedings of the 39th IEEE Confernce on Decision and Control*, pp. 4865–4870, IEEE, December 2000.

[2] F. Xia and Y. Sun, "Control-scheduling codesign: a perspective on integrated control and computing," *Dynamics of Continuous, Discrete and Impulsive Systems—Series B*, vol. 13, supplement 1, pp. 1352–1358, 2006.

[3] A. Cervin and J. Eker, "Control-scheduling codesign of real-time systems: the control server approach," *Journal of Embedded Computing*, vol. 1, no. 2, pp. 209–224, 2005.

[4] F. Xia and Y.-X. Sun, *Control and Scheduling Codesign: Flexible Resource Management in Real-Time Control Systems*, Springer Science & Business Media, 2008.

[5] A. Cervin, D. Henriksson, B. Lincoln, J. Eker, and K.-E. Årzén, "How does control timing affect performance? Analysis and simulation of timing using jitterbug and truetime," *IEEE Control Systems*, vol. 23, no. 3, pp. 16–30, 2003.

[6] Y. Wu, G. Buttazzo, E. Bini, and A. Cervin, "Parameter selection for real-time controllers in resource-constrained systems," *IEEE Transactions on Industrial Informatics*, vol. 6, no. 4, pp. 610–620, 2010.

[7] D. Seto, J. P. Lehoczky, L. Sha, and K. G. Shin, "On task schedulability in real-time control systems," in *Proceedings of the 1996 17th IEEE Real-Time Systems Symposium (RTSS '96)*, pp. 13–21, December 1996.

[8] D. Seto, J. P. Lehoczky, and L. Sha, "Task period selection and schedulability in real-time systems," in *Proceedings of the 19th IEEE Real-Time Systems Symposium (RTSS '98)*, pp. 188–198, December 1998.

[9] E. Bini and M. Di Natale, "Optimal task rate selection in fixed priority systems," in *Proceedings of the 26th IEEE Real-Time Systems Symposium (RTSS '05)*, 409, 399 pages, December 2005.

[10] E. Bini and A. Cervin, "Delay-aware period assignment in control systems," in *Proceedings of the Real-Time Systems Symposium (RTSS '08)*, pp. 291–300, December 2008.

[11] L. Tan, C. Du, and Y. Dong, "Control-performance-driven period and deadline selection for cyber-physical systems," in *Proceedings of the 10th Asian Control Conference (ASCC '15)*, pp. 1–6, Kota Kinabalu, Malaysia, May 2015.

[12] C. Du, L. Tan, and Y. Dong, "Period selection for integrated controller tasks in cyber-physical systems," *Chinese Journal of Aeronautics*, vol. 28, no. 3, pp. 894–902, 2015.

[13] G. Buttazzo, M. Velasco, and P. Marti, "Quality-of-control management in overloaded real-time systems," *IEEE Transactions on Computers*, vol. 56, no. 2, pp. 253–266, 2007.

[14] H.-J. Cha, S.-W. Park, W.-H. Jeong, and J.-C. Kim, "Performance tradeoff between control period and delay: lane keeping assist system case study," *Journal of the Korea Society of Computer and Information*, vol. 20, no. 11, pp. 39–46, 2015.

[15] B. Lincoln and A. Cervin, "Jitterbug: a tool for analysis of real-time control performance," in *Proceedings of the 41st IEEE Conference on Decision and Control*, vol. 2, pp. 1319–1324, IEEE, Las Vegas, Nev, USA, December 2002.

[16] D. Henriksson, A. Cervin, M. Andersson, and K.-E. Årzén, "Truetime: simulation of networked computer control systems," in *Proceedings of the 2nd IFAC Conference on Analysis and Design of Hybrid Systems*, Alghero, Italy, June 2006.

[17] N. Audsley, A. Burns, M. Richardson, K. Tindell, and A. Wellings, "Applying new scheduling theory to static priority preemptive scheduling," *Software Engineering Journal*, vol. 8, no. 5, pp. 284–292, 1993.

[18] EVIDENCE, "Erika enterprise manual," http://erika.tuxfamily.org/drupal/.

[19] B. Wymann, "Torcs manual installation and robot tutorial," http://www.berniw.org/aboutme/publications/torcs.pdf.

[20] Infineon, "Tc1797 user's manual," http://www.infineon.com/cms/en/product/.

[21] Tc1796 user's manual, http://www.infineon.com/cms/en/product/.

[22] National Instruments, Labview user manual, http://www.ni.com/labview/ko/.

[23] PEAK-System, "Pcan-basic parameters description," http://www.peak-system.com/PCAN-USB.199.0.html?&L=1.

[24] Logitech, Logitech g25 user manual, http://support.logitech.com/enau/product/g25-racing-wheel.