Period and Deadline Selection Problem for Real-Time Systems

Thidapat Chantem and Xiaobo Sharon Hu
Department of Computer Science & Engineering
University of Notre Dame
Notre Dame, IN 46556
{tchantem, shu}@cse.nd.edu

M.D. Lemmon
Department of Electrical Engineering
University of Notre Dame
Notre Dame, IN 46556
lemmon@nd.edu

Abstract

Task period adaptations are often used to alleviate temporal overload conditions in real-time systems, as they permit performance guarantee. Existing frameworks assume that only task periods are adjustable parameters and that task deadlines remain unchanged at all times. This paper formally introduces a more general real-time task model where task deadlines, which are less than or equal to task periods, are functions of task periods. This tight coupling between task deadlines and task periods has been discussed in a recent work in control systems and presents a novel real-time scheduling challenge. To solve this period and deadline selection problem, this paper presents some analyses in helping to identify feasible period-deadline combinations and proposes a heuristic for finding a schedulable task set.

1 Introduction

Task scheduling has long been an important research topic in real-time systems. Missing a deadline in a hard real-time system may lead to catastrophic consequences, such as failure to stop an automatically controlled train on time [8]. Although control systems have traditionally been treated as hard real-time systems, they in fact have more flexible timing requirements. That is, in general, depending on the system state, the sampling rate of a control system can vary within some interval without causing significant performance degradation. This observation is very useful when temporal overload situations occur. A real-time system is said to experience an overload when it cannot finish executing one or more tasks on time due to resource constraints. Although robust, if too many deadlines are missed or such misses occur in an unpredictable manner, a control system may no longer reach its equilibrium point, even if all system resources are dedicated to it.

Two main approaches to dealing with overloads are dropping some instances of tasks and increasing task periods. In this paper we focus on the second approach. Some existing works use optimization theory to solve the period selection problem for different scheduling algorithms [9], [10], [2]. In [4], an online period adjustment mechanism is proposed, while varying task computation times are handled in [6]. An optimal period selection algorithm was proposed in [3] based on the elastic task model.

Most previous works assume that only task periods can change. In [11], the deadline of a task varies with time, but tasks do not have periods (i.e., tasks are non-periodic). Our first contribution is the introduction of a more general task model where both task deadlines and task periods can vary within some intervals. The deadline in the real-time system sense really denotes the maximum allowable delay that a given task (a control task, for instance) can tolerate. As shown by the authors in [12], different sampling rates for a control system lead to different acceptable maximum delays (deadlines). Specifically, a smaller sampling rate means that the corresponding control task executes more often, which, in turns, allows the system to be more tolerant to a larger delay. Conversely, a larger sampling period would be more susceptible to delays and thus requires a smaller deadline.

The relationship between task periods and task deadlines poses an interesting scheduling problem, as one can no longer assume that increasing task periods will always improve schedulability. Although it is possible to set task deadlines to the smallest deadlines and only vary task periods, doing so may significantly worsen schedulability. As our second contribution, we propose a heuristic to identify a feasible period-deadline combination given that task deadlines are less than or equal to task periods.

2 Preliminaries

2.1 System Model and Assumptions

We consider a set of N periodic tasks with the following attributes: $(C_i, T_i, T_{i0}, T_{i_{max}}, D_i)$, for $i = 1, \ldots, N$, where C_i is the worst case execution time of task τ_i, and T_i is τ_i's

actual period, which must lie somewhere between T_{i0} and $T_{i_{max}}$. The parameter T_{i0} denotes the most desirable period of τ_i, as specified by the application, whereas $T_{i_{max}}$ represents the maximum period beyond which the system performance is no longer acceptable. The parameter D_i is the deadline of τ_i, and is a function of T_i. Since we focus on the case where task deadlines are less than or equal to task periods, we will assume that $D_i = f(T_i)$, where, for $[T_{i0}, T_{i_{max}}]$, $f(T_i) \leq T_i$, and $f(T_i)$ is some continuous function. All tasks start at time 0.

2.2 Schedulability Test

We assume that the Earliest Deadline First (EDF) scheduling algorithm [7] is used. A necessary condition for schedulability of any given task set is

Lemma 1 [5] *Let D_i be the deadline of task τ_i in a given task set Γ, $i = 1, \ldots, N$, and let all tasks start at time 0. Let the tasks in Γ be ordered in a non-decreasing order of deadlines. Regardless of the choices of periods, any task set that is schedulable must satisfy the following property*

$$\sum_{i=1}^{j} C_i \leq D_j, j = 1, \ldots, N \tag{1}$$

Since task deadlines can be less than or equal to periods, there exists an exact, albeit complex, schedulability test for EDF as specified by Baruah et al [1].

Theorem 1 [1] *Given a periodic task set with $D_i \leq T_i$, the task set is schedulable if and only if the following constraint is satisfied $\forall L \in \{kT_i + D_i \leq \min(B_p, H)\}$ and $k \in \mathcal{N}$ (the set of natural numbers including 0) , where B_p and H denote the busy period and hyperperiod, respectively,*

$$L \geq \sum_{i=1}^{N} \left(\left\lfloor \frac{L - D_i}{T_i} \right\rfloor + 1 \right) C_i \tag{2}$$

Verifying that (2) is satisfied for all L is the main source of complexity in the above schedulability test. To reduce the complexity of the test in Theorem 1, the authors in [5] proposed the following sufficient condition for schedulability.

Theorem 2 [5] *Given a set Γ of N tasks that satisfy Lemma 1. Let the tasks in Γ be sorted in a non-decreasing order of deadlines. A given task set is schedulable if*

$$L^* \geq \sum_{i=1}^{N} \left(\frac{L^* - D_i}{T_i} + 1 \right) C_i \tag{3}$$

where

$$L^* = \begin{cases} D_2 & : \quad D_1 + T_1 \leq D_2 \\ \min_{i=1}^{N} (T_i + D_i) & : \quad \text{otherwise} \end{cases}$$

Table 1. Task set for motivating example

Task	C_i	T_{i0}	$T_{i_{max}}$	D_i
τ_1	2	7	13	$9 - (T_1 - 10)^2, T_1 \in [7, 13]$
τ_2	2	7	13	$9 - (T_2 - 10)^2, T_2 \in [7, 13]$

For completeness, we include another existing sufficient condition for EDF schedulability.

Theorem 3 [8] *A set Γ of N tasks with $D_i \leq T_i$, $i = 1, \ldots, N$, is schedulable by the EDF policy if*

$$\sum_{i=1}^{N} \frac{C_i}{D_i} \leq 1 \tag{4}$$

2.3 Problem Definition

Given an initially infeasible set Γ of N real-time tasks where each task τ_i has an acceptable period range $[T_{i_{min}}, T_{i_{max}}]$ and the deadline D_i of τ_i is some function of its period $f(T_i)$, determine a period-deadline combination such that the task set Γ becomes schedulable. In other words, we wish to find T_i such that

$$\sum_{i=1}^{N} \left(\frac{L - f(T_i)}{T_i} \right) \cdot C_i \leq L - \sum_{i=1}^{N} C_i \tag{5}$$

$$L = \begin{cases} D_2 & : \quad D_1 + T_1 \leq D_2 \\ \min (T_i + D_i) & : \quad \text{otherwise} \end{cases} \tag{6}$$

$$T_i \geq T_{i0} \quad \text{for } i = 1, 2, \cdots, N \tag{7}$$

$$T_i \leq T_{i_{max}} \quad \text{for } i = 1, 2, \cdots, N \tag{8}$$

for $D_i = f(T_i)$, $i = 1, \ldots, N$, assuming that tasks are ordered in a non-decreasing order of deadlines, and the period bounds are provided by the applications

3 Motivations

In control systems, an advantage in using the traditional periodic task model is that the systems can then be treated as discrete-time systems for which there exists a variety of mature controller synthesis methods. However, the resultant task periods and deadlines are often very conservative. This leads to wasted resources and worsens schedulability. To address these issues, there has been a movement in the control system community to focus on an alternative approach to the periodic task model.

The state-based self triggering control system in [12] is such an example. Each task determines its own release time based on the current system state. Self-triggering can be viewed as a closed-loop form of releasing tasks for execution, whereas the traditional periodic task model is considered open-loop. Since each control task is aware of the

system state, it can dynamically adjust its own period and deadline. That is, when the period is small, the task is executed relatively often and the system is thus more tolerant to delays, permitting the deadline to be larger. On the other hand, when the period is large, the system is more susceptible to disturbances, requiring that the deadline be smaller to reduce jitters.

To understand how the deadline as a function of the period affects schedulability, let's consider a task set, which consists of two identical tasks whose attributes are shown in Table 1. Figure 1 plots the task deadlines as a function of task periods. Initially, the task set is not schedulable with $T_1 = T_2 = 7$ time units, since the initial deadlines $D_{1_0} = D_{2_0} = 0$ time units and the aggregate execution time required is 4 time units. If we simply set $T_1 = T_2 = 13$ time units, which is the maximum allowable periods, then the corresponding deadlines will be $D_1 = D_2 = 0$ time units, as before. The task set is, again, not schedulable and one can wrongly conclude that the task set cannot be made feasible. However, there exists many feasible period-deadline combinations; when $T_1 = T_2 = 10$ time units and $D_1 = D_2 = 9$ time units, for example.

In the period selection problem, since task deadlines are considered fixed, system designers must use the smallest possible deadlines to ensure that given a specific range of periods, the system will still meet some performance requirements. In this example, the smallest deadline for both task is 0 time units, which means that the task set can never be made schedulable. It is not difficult to see in this example that the task deadlines can be set to 4 time units for the task set to be feasible, regardless of the resultant periods. In general, however, both task periods and task deadlines must be considered simultaneously.

4 Period-Deadline Selection Problem

As shown in the previous section, since a task's deadline is a function of its period, adjusting the period affects both the corresponding deadline and the schedulability of the entire task set. Due to the condition in (5), the problem defined in Section 2.3 is nonlinear and non-convex. Solving such a problem directly using a nonlinear solver is inefficient and it cannot be guaranteed that a solution will be found, even if one exists. For these reasons, we propose using a heuristic, shown in Algorithm 1, to identify a feasible period-deadline combination.

The heuristic first identifies the minimum and maximum deadlines, $D_{i_{min}}$ and $D_{i_{max}}$, for each task τ_i, respectively. The maximum deadline of τ_i can directly be solved by finding the maximum of $f(T_i)$. While it may seem that the minimum deadline of τ_i can be derived in a similar manner, we use Lemma 1 to help eliminate some infeasible period-deadline combinations (shown by the right-slanted

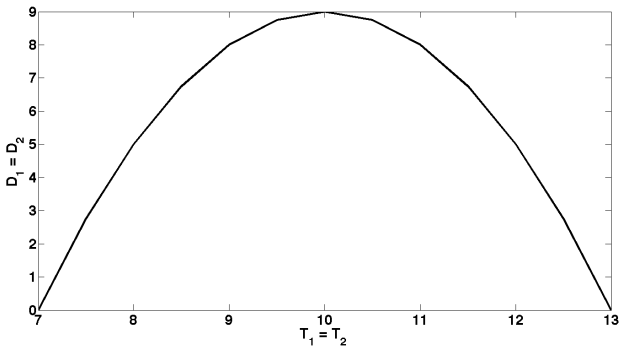

Figure 1. Deadline as a function of period

pattern in Figure 2). That is, we set $D_{i_{min}} = \sum_{j=1}^{i} C_j$, for $i = 1, \ldots, N$. If there exist many corresponding period values when the deadline of τ_i is $D_{i_{min}}$, we select the largest period that is no greater than $T_{i_{max}}$. Such a period is referred to as T_i'', $i = 1, \ldots, N$.

We now apply a series of efficient schedulability tests in an attempt to find a feasible period-deadline combination. The heuristic starts with the sufficient condition from Theorem 3. The following lemma helps to explain why only $D_{i_{max}}$, $i = 1, \ldots, N$, need to be considered.

Lemma 2 *Given a set Γ of N tasks, if the schedulability condition from Theorem 3 is not satisfied for $D_{i_{max}}$, $i = 1, \ldots, N$, then it is not satisfied for any $D_i' < D_{i_{max}}$, for $i = 1, \ldots, N$.*

If a feasible task set has not been identified after having applied the condition from Theorem 3, we move on to use the schedulability test from Theorem 2 with $D_{i_{max}}$ and T_i', for $i = 1, \ldots, N$, where T_i' is the period at which the task deadline is maximum. The following theorem is used to support this decision.

Theorem 4 *Given a set Γ of N tasks and let T_i' be the period obtained when $D_i = D_{i_{max}}$, for $i = 1, \ldots, N$. If the condition in Theorem 2 is not satisfied for $D_{i_{max}}$ and T_i', $i = 1, \ldots, N$, then it is not satisfied for any $D_i < D_{i_{max}}$ and $T_i \leq T_i'$, $i = 1, \ldots, N$.*

The left-slanted region in Figure 2 is a result of Theorem 4, while the area with no pattern indicates the remaining search region. The next theorem is essential in identifying a solution when all previous, more efficient tests have failed.

Theorem 5 *Given a set Γ of N tasks with the deadline of τ_i as a function of its period, and given that the period of τ_i can vary within $[T_i', \min(T_i'', T_{i_{max}})]$, for $i = 1, \ldots, N$. A feasible solution to the problem defined in (5)–(8) must satisfy the following conditions.*

$$0 = \mu_0 \left(\sum_{j=1}^{N} (L - f(U_j)) \cdot U_j - L + \sum_{j=1}^{N} C_j \right) \quad (9)$$

Algorithm 1 FindFeasiblePeriodsDeadlines(Γ)

for each $\tau_i \in \Gamma$ **do**
  $D_{i_{max}} \leftarrow \max_{T_i \in [T_{i_{min}}, T_{i0}]} f(T_i)$
  $T_i' \leftarrow$ period when deadline is $D_{i_{max}}$
  $D_{i_{min}} \leftarrow \sum_{j=1}^{i} C_j$
  $T_i'' \leftarrow$ period when deadline is $D_{i_{min}}$
  if $D_{i_{min}} > D_{i_{max}}$ **then**
    return $\emptyset$
  end if
end for
$result \leftarrow \sum_{i=1}^{N} \frac{C_i}{D_{i_{max}}}$
if $result \leq 1$ **then**
  return $[D_{i_{max}}, T_i']$, for $i = 1, \ldots, N$
end if
order tasks in a non-decreasing order of $D_{i_{max}}$ $i = 1, \ldots, N$
compute L as in (6) using $D_{i_{max}}$ and T_i', $1 \leq i \leq N$
$result \leftarrow \sum_{i=1}^{N} \left(\frac{L - D_{i_{max}}}{T_i'} + 1 \right) \cdot C_i$
if $result \leq L$ **then**
  return $[D_{i_{max}}, T_i']$, for $i = 1, \ldots, N$
end if
order tasks in a non-decreasing order of $D_{i_{min}}$ $i = 1, \ldots, N$
compute L as in (6) using $D_{i_{min}}$ and T_i'', $1 \leq i \leq N$
$result \leftarrow \sum_{i=1}^{N} \left(\frac{L - D_{i_{min}}}{T_i''} + 1 \right) \cdot C_i$
if $result \leq L$ **then**
  return $[D_{i_{min}}, T_i'']$, for $i = 1, \ldots, N$
end if
search for a solution using Theorem 5

$$0 = \mu_i(U_i'' - U_i) \quad (10)$$

$$0 = \lambda_i(U_i - U_i') \quad (11)$$

For notational clarity, $U_i' = C_i/T_i'$ and $U_i'' = C_i/T_i''$ are used instead of T_i' and T_i'', respectively. The above theorem is a direct consequence of applying the Karush-Kuhn-Tucker conditions to the problem defined in Section 2.3. The above theorem can be used to identify a feasible task set during the search. The best search algorithm for the period and deadline selection problem is currently under investigation.

5 Conclusions

We introduced a more general real-time task model where each task deadline is a function of the corresponding period. This requirement is directly derived from control systems where the deadlines reflect the maximum allowable delays as tolerated by a given system and vary according to the sampling periods. Since existing schedulability conditions cannot adequately be used to determine feasibility for this novel task model, we also proposed a heuristic to identify a schedulable period-deadline combination.

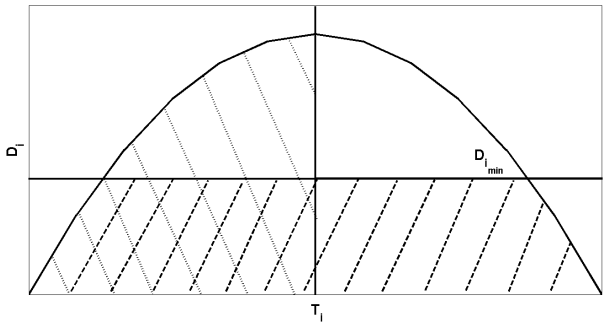

Figure 2. Infeasible schedulability regions

References

[1] S. Baruah, L. Rosier, and R. Howell. Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor. *Real-Time Systems*, 2(4):301–324, Nov. 1990.

[2] E. Bini and M. D. Natale. Optimal task rate selection in fixed priority systems. In *Proc. Real-Time Systems Symposium*, pages 399–409, 2005.

[3] G. Buttazzo, G. Lipari, and L. Abeni. Elastic task model for adaptive rate control. In *Proc. Real-Time Systems Symposium*, pages 286–295, 1998.

[4] A. Cervin, J. Eker, B. Bernhardsson, and K.-E.Årzén. Feedback-feedforward scheduling of control tasks. *Real-Time Systems*, 23(1):25–53, July 2002.

[5] T. Chantem, X. Hu, and M. Lemmon. Generalized elastic scheduling. In *Proc. Real-Time Systems Symposium*, pages 236–245, 2006.

[6] X. Koutsoukos, R. Tekumalla, B. Natarajan, and C. Lu. Hybrid supervisory utilization control of real-time systems. In *Proc. Real-Time & Embedded Technology and Applications Symposium*, pages 12–21, 2005.

[7] C. Liu and J. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the ACM*, 20(1):46–61, Jan. 1973.

[8] J. W. S. Liu. *Real-Time Systems*. Prentice-Hall, NJ, 2000.

[9] D. Seto, J. Lehoczky, and L. Sha. Task period selection and schedulability in real-time systems. In *Proc. Real-Time Systems Symposium*, pages 188–199, 1998.

[10] D. Seto, J. Lehoczky, L. Sha, and K. Shin. On task schedulability in real-time control systems. In *Proc. Real-Time Systems Symposium*, pages 13–21, 1996.

[11] C.-S. Shih and J. W. Liu. State-dependent deadline scheduling. In *Proc. Real-Time Systems Symposium*, pages 3–14, 2002.

[12] X. Wang and M. Lemmon. Self-triggered feedback control systems with finite-gain l2 stability. *Submitted to Transactions on Automatic Control*, 2007.