Optimal Dynamic Voltage Scaling in Energy-Limited Nonpreemptive Systems with Real-Time Constraints

Jianfeng Mao, *Student Member*, *IEEE*, Christos G. Cassandras, *Fellow*, *IEEE*, and Qianchuan Zhao, *Member*, *IEEE*

Abstract—Dynamic voltage scaling is used in energy-limited systems as a means of conserving energy and prolonging their life. We consider a setting in which the tasks performed by such a system are nonpreemptive and aperiodic. Our objective is to control the processing rate over different tasks so as to minimize energy subject to hard real-time processing constraints. Under any given task scheduling policy, we prove that the optimal solution to the offline version of the problem can be efficiently obtained by exploiting the structure of optimal sample paths, leading to a new dynamic voltage scaling algorithm termed the *Critical Task Decomposition Algorithm* (CTDA). The efficiency of the algorithm rests on the existence of a set of critical tasks that decompose the optimal sample path into decoupled segments within which optimal processing times are easily determined. The algorithm is readily extended to an online version of the problem as well. Its worst-case complexity of both offline and online problems is $O(N^2)$.

Index Terms—Hard real-time system, voltage scaling, optimal control, sensor networks, nonpreemptive.

1 Introduction

Minimizing energy consumption in low-power systems has become a critical design consideration, especially in view of proliferating portable and mobile real-time embedded systems whose lifetime is strongly dependent on battery management. In sensor networks, for example, nodes incorporate small, inexpensive devices with limited battery capabilities. Prolonging battery life is closely tied to the network's overall performance; for instance, the failure of a few nodes can cause significant topological changes which require substantial additional energy to reorganize the network [1]. In low-power systems, the processor reportedly accounts for 18 to 30 percent of the overall power consumption and often exceeds 50 percent [2]. Controlling the voltage and clock frequency provides the means to regulate processor power consumption leading to Dynamic Voltage Scaling (DVS) techniques [2], [3], [4]. For CMOS processors, the total energy consumption E_{total} mainly includes two parts: the dynamic energy consumption $E_{dyn} \propto V_{dd}^2$ caused by the supply voltage V_{dd} and the leakage energy consumption E_{leak} caused by the leakage currents. Thus,

$$E_{total} = E_{dyn} + E_{leak}, \quad E_{dyn} = C_1 V_{dd}^2, \tag{1}$$

• J. Mao and C.G. Cassandras are with the Department of Manufacturing Engineering, 15 St. Mary's St., Boston University, Brookline, MA 02446. E-mail: {jfmao, cgc}@bu.edu.
• Q. Zhao is with the Center for Intelligent and Networked Systems, Department of Automation, Tsinghua University, Beijing 100084, China. E-mail: zhaoqc@tsinghua.edu.cn.

Manuscript received 1 June 2006; revised 2 Nov. 2006; accepted 17 Dec. 2006; published online 7 Feb. 2007.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-0149-0606.
Digital Object Identifier no. 10.1109/TMC.2007.1024.

and the processing frequency (clock speed) is given by

$$f = \frac{(V_{dd} - V_t)^{\alpha}}{C_2 V_{dd}}, \tag{2}$$

where C_1, C_2, and $\alpha \in [1, 2]$ are constants dependent on the physical characteristics of a device and V_t is the threshold voltage, so that $V_{dd} \geq V_t$. These relationships may be approximate, but the functional interdependence of V_{dd}, E_{total}, and f clearly indicates that reducing the voltage provides an opportunity to reduce energy at the expense of longer delays, which adversely affects performance with possibly catastrophic consequences in hard real-time systems [5]. Thus, managing this trade-off becomes an essential design and dynamic control problem.

In this paper, we consider system processing tasks which are nonpreemptive and aperiodic. We also assume that the order in which the tasks are to be executed is given a priori according to some scheduling policy. Note that determining the optimal order in which to execute tasks is a separate (NP-hard) problem. Our goal is to assign processing times (equivalently, processor speeds through voltage control) to tasks so as to minimize a total energy consumption function while guaranteeing that no task completion exceeds a given deadline. Our approach is based on a similar optimization framework as in [6] and [7], but the structure of the problem in our setting leads to some attractive properties that we exploit to develop a new DVS algorithm. This algorithm is shown to improve upon the DVS algorithm developed by Yao et al. [8] by a factor of N and is guaranteed to obtain the optimal solution of the offline problem (where task arrivals and deadlines are known) under any given task scheduling policy. The algorithm is also readily extended to an online problem where task arrivals and deadlines are not known in advance, but tasks are assumed to arrive within a given interval. Its worst-case complexity, for both offline and online cases, is $O(N^2)$. The efficiency of the algorithm rests on the existence of a set of *critical tasks* that decompose the optimal sample path into decoupled segments within which optimal processing times are easily determined. The key to the low complexity lies in the fact that a simple procedure can be developed to detect these critical tasks.

The paper is organized as follows: Section 2 summarizes related work. Section 3 develops an offline DVS algorithm. In Section 3.1, the offline DVS problem with hard real-time constraints is formulated and the two main challenges it poses are described. We subsequently identify the structural properties of the optimal sample path for this problem, first based on a *busy period* decomposition (Section 3.2) and then based on further decomposing a busy period through *critical tasks* (Section 3.3). The final solution to the offline problem is obtained in Section 3.3, where our main result, the *Critical Task Decomposition Algorithm* (CTDA), is presented. Section 4 extends the offline DVS algorithm to online problems. In Section 5, some simulation-based experimental results are presented, illustrating the low computational complexity of the CTDA. Section 6 concludes the paper.

2 Related Work

A number of DVS algorithms have been proposed over the last decade. Most of them are designed for preemptive scheduling of real-time systems, as in [8], [9], and [10]. In practice, when considering systems with very limited resources (e.g., restricted battery life, storage capacity) *nonpreemptive* scheduling is often a better choice because uncontrolled preemption can give rise to a large number of context switches requiring larger stack sizes and increased energy consumption [11], [12]. Moreover, nonpreemptive scheduling is necessary in some applications, such as executing wireless packet transmission tasks which also happen to be particularly energy-intensive and fall within the framework presented in this paper [13], [14]. DVS algorithms developed for the nonpreemptive case have been reviewed in [15]. Many of them were developed for systems with periodic tasks, as in [16]. In this paper, we consider a system with *aperiodic* tasks which arise in a setting consisting of asynchronously operating components (e.g., a sensor network where sensing units asynchronously supply data to a processing node). In the offline version of our optimization problem, task arrival times and associated deadlines are known in advance. In the online case, task arrival times at the processor are generally random. As discussed in Section 4, we will model them as being constrained to occur in a given time interval.

Our approach is motivated by the work in [6], where the nonpreemptive and aperiodic case is considered with a known arrival time schedule and an optimal control problem is solved with an objective function incorporating the trade-off between processing performance and task timeliness. The problem is solved through the so-called Forward Decomposition Algorithm (FA) with applications motivated from manufacturing systems; it was later applied to the DVS problem in [17]. The FA can avoid the combinatorial complexity that often comes with such scheduling problems, but it still requires the solution of N (the number of tasks) nonlinear programming (NLP) problems, which is generally demanding for online applications with limited on-board computational capacity. For the case of tasks with real-time constraints, Yao et al. [8] developed an algorithm (referred to as YDSA in the rest of the paper) which does not involve solving NLP problems. Although the YDSA was originally designed for preemptive systems, it can also be applied to nonpreemptive cases after some simple modifications. Its complexity in solving offline problems is $O(N^3)$ (originally claimed to be $O(N \log^2 N)$) and becomes $O(N^4)$ when solving online problems; this is typically not sufficiently efficient for demanding real-time applications. In the following sections, we will develop a new algorithm provably optimizing energy costs subject to real-time constraints whose complexity is $O(N^2)$ by exploiting the structure of optimal sample paths of the associated nonlinear optimization problem.

3 Offline DVS Algorithm

3.1 Problem Formulation

The hard real-time system we consider is modeled as a single-stage queuing system with the objective of minimizing energy consumption while guaranteeing to meet hard real-time constraints (deadlines). In the offline problem, the arrival times of tasks and their associated deadlines are known.

To achieve this goal, there are two issues we need to handle: 1) how fast to process tasks and 2) how to order tasks. Since the latter issue is NP-hard for nonpreemptive systems with aperiodic tasks [15], it is necessary for a system with limited computational resources to adopt a specific scheduling policy of sufficient simplicity to be used in the real-time environment we have described and to facilitate the development of DVS algorithms with polynomial complexity. For example, the Earliest Deadline First (EDF) policy and the First Come First Served (FCFS) policy are appropriate candidates. Our approach is independent of the scheduling policy, since, once the policy to decide task order is selected, we can fix the processing order of tasks and model their queuing dynamics through the standard Lindley equation [18] as follows:

$$x_i = \max(x_{i-1}, a_i) + \mu_i \tau_i, \tag{3}$$

where a_i and x_i are the arrival time and departure time, respectively, of task i, τ_i is the processing time per operation for task i (the controllable variable in our problem), and μ_i is the number of operations (or instructions) needed for task i. For example, if we apply the FCFS policy, we then order the task indices so that $a_1 \le a_2 \le \cdots \le a_N$; if we apply the EDF policy, then we order the task indices so that $d_1 \le d_2 \le \cdots \le d_N$. Clearly, EDF and FCFS may be poor policies for some nonpreemptive cases, but our approach is not limited to these policies; it is applicable to any given policy which may be chosen depending on the application of interest.

We can now concentrate on the first issue above, i.e., controlling the processing rate of tasks so as to minimize an energy consumption function while ensuring that all task deadlines are satisfied. Thus, we formulate a deterministic finite-horizon nonlinear optimization problem which we will refer to in the sequel as Problem J:

$$\min_{\tau_1,\ldots,\tau_N}\left\{J=\sum\nolimits_{i=1}^{N}\mu_i\theta(\tau_i)\right\}$$
$$\text{s.t.}\quad \tau_i\geq 0,\quad i=1,\ldots,N;\quad x_0=-\infty;$$
$$x_i=\max(x_{i-1},a_i)+\mu_i\tau_i,\quad i=1,\ldots,N;$$
$$x_i\leq d_i,\quad i=1,\ldots,N,$$

where N is the total number of tasks to be processed, d_i is the deadline of task i, and $\theta(\cdot)$ is the energy consumption per operation. Note that the hard real-time constraints are captured through $x_i \leq d_i$, $i=1,\ldots,N$. In this problem, the control variable is the processing time per operation τ_i (equivalently, $f_i = 1/\tau_i$ is the processor speed when processing task i) which is directly related to voltage as explained below. Usually, DVS techniques operate through the software/hardware interface [17]. The software is not aware of the operating voltage used by the hardware, i.e., the voltage is adjusted depending on the processing time per operation for the desired task.

Regarding the energy consumption function $\theta(\tau_i)$ in Problem J, we emphasize that the analysis that follows is independent of its precise form as long as it is a strictly convex and monotonically decreasing function of τ_i. However, we can obtain more information on $\theta(\tau_i)$ by using the relationships (1)-(2) from which we can write

$$\theta(\tau_i)=E_{total}=C_1(h^{-1}(\tau_i))^2+E_{leak},\tag{4}$$

where

$$\tau_i=h(V_{dd})=\frac{C_2V_{dd}}{(V_{dd}-V_t)^{\alpha}}$$

and $h^{-1}(\tau_i)$ is its inverse function. For $\alpha\in[1,2]$, it can be easily seen that $h(V_{dd})$ is convex and monotonically decreasing. Based on this fact, $h^{-1}(\tau_i)$ can be shown to be also convex and monotonically decreasing. In particular, since $h(V_{dd})$ is convex, then for any τ_1, τ_2, and any $\beta\in[0,1]$,

$$\begin{aligned}&\beta\tau_1+(1-\beta)\tau_2\\&=\beta h(h^{-1}(\tau_1))+(1-\beta)h(h^{-1}(\tau_2))\\&\geq h(\beta h^{-1}(\tau_1)+(1-\beta)h^{-1}(\tau_2)).\end{aligned}$$

Since $h(V_{dd})$ is monotonically decreasing, $h^{-1}(\tau_i)$ is also monotonically decreasing, which implies, for any $\beta\in[0,1]$:

$$\begin{aligned}&h^{-1}(\beta\tau_1+(1-\beta)\tau_2)\\&\leq h^{-1}(h(\beta h^{-1}(\tau_1)+(1-\beta)h^{-1}(\tau_2)))\\&=\beta h^{-1}(\tau_1)+(1-\beta)h^{-1}(\tau_2),\end{aligned}$$

that is, $h^{-1}(\tau_i)$ is convex and also monotonically decreasing.

Due to the presence of E_{leak} in (4), $\theta(\tau_i)$ is generally not convex or monotonically decreasing [19] since E_{leak} increases with V_{dd} decreasing. However, there exists a lower operating bound, $V_{\min}$, on V_{dd} such that E_{leak} can be regarded as a small constant in the operating range. Since $h^{-1}(\tau_i)$ is strictly convex and monotonically decreasing in τ_i, $\theta(\tau_i)$ is a convex and monotonically decreasing function in τ_i with the corresponding upper bound expressed as $\tau_{\max}=\frac{C_2V_{\min}}{(V_{\min}-V_t)^{\alpha}}$. Moreover, the supply voltage V_{dd} is physically limited by an upper bound $V_{\max}$, that is, there is a corresponding lower bound for τ_i expressed as $\tau_{\min}=\frac{C_2V_{\max}}{(V_{\max}-V_t)^{\alpha}}$. To summarize, we have

$$\begin{aligned}&\tau_{\min}\leq\tau_i\leq\tau_{\max},\quad \tau_{\min}=\frac{C_2V_{\max}}{(V_{\max}-V_t)^{\alpha}},\\&\tau_{\max}=\frac{C_2V_{\min}}{(V_{\min}-V_t)^{\alpha}}.\end{aligned}\tag{5}$$

Note that the boundary constraints $\tau_{\min}\leq\tau_i\leq\tau_{\max}$ are omitted in Problem J. However, we shall show in Section 3.4 (Proposition 3) that the solution of the modified Problem J with the constraints (5) included is easily recovered from the solution of Problem J. We can, therefore, first study this problem. Before proceeding, we stress once again that the precise form of $\theta(\tau_i)$ or the values of the constants C_1, C_2 are not essential; as we shall see, what matters is only the assumption that $\theta(\tau_i)$ is a strictly convex and monotonically decreasing function of τ_i.

Looking at Problem J, we can see that there are two main difficulties: 1) The potentially high dimensionality of the control vector $(\tau_1,\ldots,\tau_N)$, given that the number of tasks N may be very large, which can lead to a combinatorial complexity similar to that encountered in [6], and 2) the nondifferentiability of the constraints introduced by the presence of the "max" operators. With regard to the second issue, it is certainly possible to eliminate the max operators by replacing the nondifferentiable constraints of the form (3) by inequality constraints $s_i\geq a_i$, $s_i\geq x_{i-1}$, where s_i is a dummy variable, together with linear constraints $x_i=s_i+\mu_i\tau_i$, $i=1,\ldots,N$. However, this approach would double the dimensionality of the problem and also add $2N$ inequality constraints, which makes it even less likely for a resource-limited device to be able to handle the computational complexity of such an NLP problem. In the next two subsections, we will show how to overcome both of these difficulties by decomposing the original problem into smaller, simpler problems whose solution (and, hence, the solution of Problem J) can be obtained without any need for an NLP problem solver.

3.2 Busy Period Decomposition

We begin with the observation that a sample path of the queuing system whose dynamics are captured through (3) can be decomposed into *busy periods*, i.e., intervals during which the processor is busy processing tasks, separated by idle periods. This decomposition allows us to eliminate the nondifferentiable constraints in Problem J without introducing any additional decision variables or inequality constraints and replace the original problem with several simpler ones of lower dimensionality. Let x_i^* denote the optimal departure time of task i.

Definition 1. *A **Busy Period (BP)** is a contiguous set of tasks $\{k,\ldots,n\}$ such that $x_{k-1}^*<a_k$, $x_n^*<a_{n+1}$, and $x_i^*\geq a_{i+1}$ for $i=k,\ldots,n-1$.*

If we know that a particular set of tasks $\{k,\ldots,n\}$ defines a BP, then we can associate a problem with this BP which is simpler than Problem J and will be referred to as Problem $Q(k,n)$:

$$\min_{\tau_k,\ldots,\tau_n} \left\{ Q(k,n) = \sum\nolimits_{i=k}^{n} \mu_i \theta(\tau_i) \right\}$$
$$\text{s.t.} \quad x_i = a_k + \sum\nolimits_{j=k}^{i} \mu_j \tau_j, \quad \tau_i \geq 0, \quad i = k, \ldots, n;$$
$$x_n \leq d_n; \quad a_{i+1} \leq x_i \leq d_i, \quad i = k, \ldots, n-1.$$

By the definition of a BP, it is clear that $x_k = a_k + \mu_k \tau_k$ and $x_i = x_{i-1} + \mu_i \tau_i$ for all $i = k+1, \ldots, n$. Therefore, the nondifferentiable constraints (3) are replaced by $x_i = a_k + \sum_{j=k}^{i} \mu_j \tau_j$, $i = k, \ldots, n$, and the dimensionality of $Q(k,n)$ is no larger than N. Thus, if we can identify all BPs in the task set $\{1, \ldots, N\}$, then we can decompose Problem J into a number of smaller and simpler problems of the form $Q(k,n)$. Let us, therefore, focus on identifying this BP structure. Proposition 1 provides the means for achieving this through simple comparisons of the known data a_{i+1} and d_i for every $i = 1, \ldots, N$.

Proposition 1. *Tasks $\{k, \ldots, n\}$ constitute a single BP and its last task n ends at d_n, i.e., $x_n^* = d_n$, if and only if $a_k > d_{k-1}$, $a_{n+1} > d_n$, and $a_{i+1} \leq d_i$ for each $i = k, \ldots, n-1$.*

Proof. See the Appendix. □

This property implies that a BP can be identified by simple comparisons of the available arrival and deadline information, a procedure whose complexity is $O(N)$. Moreover, since $x_n^* = d_n$, the end of every BP is given by the (known) deadline of task n. This implies that we can further simplify $Q(k,n)$ by replacing the inequality constraint $x_n \leq d_n$ by the equality constraint $x_n = d_n$.

The BP decomposition above allows us to concentrate on solving Problem $Q(k,n)$ with the added simplification that the inequality constraint "$x_n \leq d_n$" is replaced by "$x_n = d_n$." However, it should be noted that the dimensionality of Problem $Q(k,n)$ may still be high (N in the worst case, where tasks $\{1, \ldots, N\}$ constitute a single BP). In addition, $Q(k,n)$ includes inequality constraints on x_i that make its solution harder to obtain. In the next subsection, we will introduce a new structural property allowing us to further decompose each one of the $Q(k,n)$ problems and ultimately solve it through a computationally simpler algorithm.

3.3 Critical Task Decomposition

In this section, we identify an additional structural property that leads to decomposition of a BP $\{k, \ldots, n\}$ into subsets termed "blocks" associated with certain "critical tasks." We begin with the following lemma:

Lemma 1. *1) If $\tau_i^* > \tau_{i+1}^*$, then $x_i^* = a_{i+1}$, and 2) if $\tau_i^* < \tau_{i+1}^*$, then $x_i^* = d_i$.*

Proof. See the Appendix. □

Lemma 1 identifies two cases when an optimal task departure time x_i^* is readily computed: If we can detect which of the two cases applies, then x_i^* is directly determined as either a_{i+1} or d_i. We associate these two cases with "critical tasks" through the following definitions:

Definition 2. *If $\tau_i^* \neq \tau_{i+1}^*$, task i is* ***critical****. If $\tau_i^* > \tau_{i+1}^*$, then task i is* ***left-critical****. If $\tau_i^* < \tau_{i+1}^*$, then task i is* ***right-critical****.*

Definition 3. *A* ***block*** *in a BP $\{k, \ldots, n\}$ is a contiguous set of tasks $\{p, \ldots, q\}$ $(k \leq p \leq q \leq n)$ such that $\tau_{p-1}^* \neq \tau_p^*$, $\tau_q^* \neq \tau_{q+1}^*$, and $\tau_i^* = \tau_j^*$ for all $i, j \in \{p, \ldots, q\}$.*

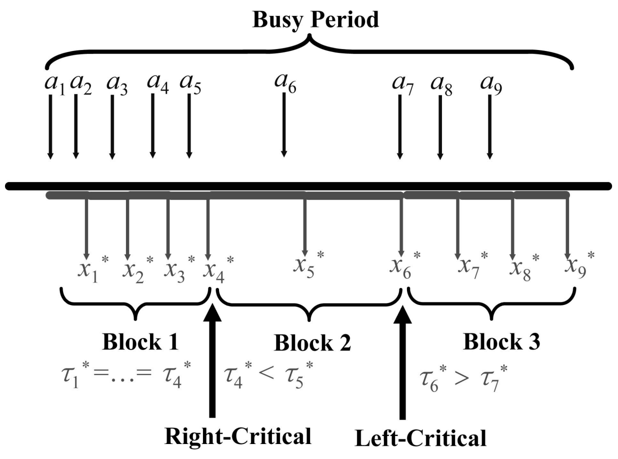

Busy Period
a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8 a_9
x_1^* x_2^* x_3^* x_4^* x_5^* x_6^* x_7^* x_8^* x_9^*
Block 1
$\tau_1^* = \ldots = \tau_4^*$
Block 2
$\tau_4^* < \tau_5^*$
Block 3
$\tau_6^* > \tau_7^*$
Right-Critical
Left-Critical

Fig. 1. An example with optimal controls.

Using the definitions above, a BP consists of a set of blocks separated by critical tasks. If all left-critical and right-critical tasks in a BP can be detected, then the optimal controls for tasks in each block can be directly and easily determined. An example is shown in Fig. 1, where, for simplicity, we let $\mu_1 = \cdots = \mu_9 = 1$. Assume we can detect that $\tau_4^* < \tau_5^*$ so that task 4 is right-critical, which implies that $x_4^* = d_4$ from Lemma 1. Then, the optimal controls of tasks {1, 2, 3, 4} in BLOCK 1 are all equal and can be determined as $\tau_i^* = (d_4 - a_1)/4$, $i = 1, \ldots, 4$. Similarly, if we can also detect that $\tau_6^* > \tau_7^*$, then task 6 is left-critical, which implies that $x_6^* = a_7$. Thus, the optimal controls of tasks {5, 6} in BLOCK 2 can be determined as $\tau_i^* = (a_7 - d_4)/2$, $i = 5, 6$, and the optimal controls of tasks {7, 8, 9} in BLOCK 3 are given by $\tau_i^* = (d_9 - a_7)/3$, $i = 7, 8, 9$. Hence, all that remains is to identify all critical tasks. This is accomplished next and our final result is formalized in Proposition 2.

When examining a BP, we apply a forward way to identify all critical tasks; that is, in every iteration, we always search for the first critical task in the BP and identify the corresponding block. In the next iteration, we examine the remainder of the BP as if it were a new one and detect the first critical task in it. Repeating the process, all critical tasks can be found. Based on this idea, we only need to focus on finding the first critical task in a BP. This is accomplished by formulating Problem $C(p)$ below, where $k \leq p \leq n$ and s_p is the starting time of the current BP, i.e.,

$$s_p = a_k \mathbf{1}[p = k] + x_{p-1}^* \mathbf{1}[p > k],$$

where $\mathbf{1}[\cdot]$ is the usual indicator function. Problem $C(p)$ is defined as follows:

$$\min_{\tau_p,\ldots,\tau_n} \left\{ C(p) = \sum\nolimits_{i=p}^{n} \mu_i \theta(\tau_i) \right\}$$
$$\text{s.t.} \quad x_i = s_p + \sum\nolimits_{j=p}^{i} \mu_j \tau_j, \quad \tau_i \geq 0, \quad i = p, \ldots, n;$$
$$a_{i+1} \leq x_i \leq d_i, \quad i = p, \ldots, n-1; \quad x_n = d_n.$$

This can be regarded as a subproblem of $Q(k,n)$. In particular, if $p = k$, then $C(k)$ coincides with Problem $Q(k,n)$. If $p > k$, then task $p-1$ is the critical task identified in the last iteration and the remaining tasks $\{p, \ldots, n\}$ are viewed as a new BP starting at $s_p = x_{p-1}^*$ and ending at d_n.

Proposition 2 provides the means to determine the first critical task in the BP $\{p, \dots, n\}$. We first establish some convenient notations. Define, for $i = p, \dots, n-1$,

$$G_i(p) = \frac{a_{i+1} - s_p}{\sum_{j=p}^{i} \mu_j}, \quad H_i(p) = \frac{d_i - s_p}{\sum_{j=p}^{i} \mu_j}, \tag{6}$$

and

$$G_n(p) = H_n(p) = \frac{d_n - s_p}{\sum_{j=p}^{n} \mu_j}.$$

Note that $G_i(p)$ is the processing time per operation assigned to all tasks in a block $[s_p, a_{i+1}]$ which ends with a right-critical task i (by Lemma 1). The same is true for $H_i(p)$ when the block $[s_p, d_i]$ ends with a right-critical task. Both $G_i(p)$ and $H_i(p)$ are readily computed for any given p and $i = p, \dots, n$.

In addition, we define, for $i = p+1, \dots, n$,

$$\begin{aligned} G_i^*(p) &= \max_{j \in \{p,\dots,i-1\}} G_j(p), \\ H_i^*(p) &= \min_{j \in \{p,\dots,i-1\}} H_j(p), \end{aligned} \tag{7}$$

and

$$\begin{aligned} L_i(p) &= \arg \max_{j \in \{p,\dots,i-1\}} G_j(p), \\ R_i(p) &= \arg \min_{j \in \{p,\dots,i-1\}} H_j(p), \end{aligned} \tag{8}$$

where the operators "$\arg\max$" and "$\arg\min$" stand for the argument of the maximum and the minimum, respectively. By convention, if there are multiple indices in $\{p, \dots, i-1\}$ that satisfy the definitions of $L_i(p)$ and $R_i(p)$ in (8), then we choose the largest index.

Proposition 2. *Consider the problem $C(p)$ and some $i \in \{p+1, \dots, n\}$ such that $G_j(p) \le H_j^*(p)$ and $H_j(p) \ge G_j^*(p)$ for all $j \in \{p+1, \dots, i-1\}$. Then,*

1. *If $G_i(p) > H_i^*(p)$, then task $R_i(p)$ is a right-critical task and also the first critical task in the BP $\{p, \dots, n\}$. Moreover, $\tau_j^* = H_i^*(p)$ for all $j \in \{p, \dots, R_i(p)\}$.*
2. *If $H_i(p) < G_i^*(p)$, then task $L_i(p)$ is a left-critical task and also the first critical task in the BP $\{p, \dots, n\}$. Moreover, $\tau_j^* = G_i^*(p)$ for all $j \in \{p, \dots, L_i(p)\}$.*

Proof. See the Appendix. □

Based on Lemma 1 and Proposition 2, an algorithm that executes the critical task decomposition and determines the optimal controls is given in Table 1. We shall refer to it as the *Critical Task Decomposition Algorithm* (CTDA).

3.4 Additional Boundary Constraints

As already mentioned, in Problem J, we omitted the constraints (5) imposed on the controllable variables τ_i, $i = 1, \dots, N$. Having obtained through Proposition 2 the solution of Problem J, we will now show that the solution of the modified Problem $\tilde{J}$ below (incorporating the boundary constraints $\tau_{\min} \le \tau_i \le \tau_{\max}$) can be recovered from that of Problem J:

TABLE 1
Pseudocode of the Critical Task Decomposition Algorithm (CTDA)

```
Step 1: Initialization: $p = k$, $s_p = a_k$;
Step 2: Identification of the first critical task in $C(p)$;
        $i = p + 1$;
        while($i \le n$) {
          if ($G_i(p) > H_i^*(p)$) {
            $R_i(p)$ is the first critical task in $C(p)$, and
                right-critical;
            $\tau_j^* = H_i^*(p)$, $j = p, ..., R_i(p)$;
            $p = R_i(p) + 1$; $s_p = x_{R_i(p)}^* = d_{R_i(p)}$;
            goto Step 2; }
          if ($H_i(p) < G_i^*(p)$){
            $L_i(p)$ is the first critical task in $C(p)$, and
                left-critical;
            $\tau_j^* = G_i^*(p)$, $j = p, ..., L_i(p)$;
            $p = L_i(p) + 1$; $s_p = x_{L_i(p)}^* = a_{L_i(p)+1}$;
            goto Step 2; }
          $i = i + 1$; }
Step 3: Termination: $\tau_j^* = G_n(p)$, $j = p, .., n$;
```

$$\min_{\tau_1,\dots,\tau_N} \left\{ \tilde{J} = \sum_{i=1}^{N} \mu_i \theta(\tau_i) \right\}$$

$$\begin{aligned} \text{s.t.} \quad & \tau_{\min} \le \tau_i \le \tau_{\max}, \quad i = 1, \dots, N; \\ & x_i = \max(x_{i-1}, a_i) + \mu_i \tau_i, \quad i = 1, \dots, N; \\ & x_0 = -\infty; \quad x_i \le d_i, \quad i = 1, \dots, N. \end{aligned}$$

The connection between Problems J and $\tilde{J}$ is established through Proposition 3.

Proposition 3. *Let $\tilde{\tau}_i^*$ denote the optimal solution of Problem $\tilde{J}$. If $\tau_i^* \ge \tau_{\min}$ for $i = 1, \dots, N$, then $\tilde{\tau}_i^* = \min(\tau_i^*, \tau_{\max})$; otherwise, Problem $\tilde{J}$ is infeasible.*

Proof. See the Appendix. □

It follows from Proposition 3 that, having obtained the solution of Problem J, we can check the feasibility of Problem $\tilde{J}$ and derive its optimal solution $\tilde{\tau}_i^*$, $i = 1, \dots, N$, as a function of $\tau_{\min}$, $\tau_{\max}$, and τ_i^*, $i = 1, \dots, N$.

3.5 Complexity Analysis

In this section, we analyze the complexity of the CTDA and compare it to the YDSA [8]. As mentioned in Section 2, the YDSA is designed for preemptive models, but it can also be used (after some simple modifications) to solve the offline DVS problem with nonpreemptive scheduling which we have considered. Thus, it is worth comparing the two approaches in terms of their relative computational complexities. As shown next, through the use of *critical tasks* in the CTDA, we have developed results in lower complexity than the YDSA, which relies on the identification of *critical intervals*. We point out that the notion of "criticality" in the two algorithms is entirely different.

In the CTDA, we define a critical task as one whose processing time per operation differs from that of the next task, i.e., $\tau_i^* \ne \tau_{i+1}^*$. In [8], a critical interval (when the system is nonpreemptive and operates under FCFS) is defined as one with the largest intensity, i.e.,

$$I^* = \arg \max_{I \in \{[a_i, d_j] | i,j=1,\ldots,N\}} \sum_{k=i}^{j} \mu_k/(d_j - a_i),$$

which can be regarded as containing those tasks with the shortest processing time per operation. Let n_b denote the number of tasks in a BP. In each iteration, the CTDA identifies a critical task with complexity $O(n_b)$, while the YDSA identifies a critical interval with complexity $O(n_b^2)$. Thus, the CTDA solves Problem $Q(k,n)$ with complexity $O(n_b^2)$, while the YDSA solves it in $O(n_b^3)$.

Looking back at Problem J, let n_m denote the size of the longest BP in the task set $\{1,\ldots,N\}$. Then, the CTDA solves Problem J with complexity $O(Nn_m)$, while the YDSA solves it in $O(Nn_m^2)$ since there are approximately N/n_m BPs. In the worst case, $n_m = N$, i.e., when the whole process consists of a single BP, the complexity of the CTDA is $O(N^2)$ while the complexity of YDSA is $O(N^3)$.

4 Online DVS Algorithm

4.1 Problem Formulation and Solution

In the offline problem solved so far, the task arrival times a_i, $i = 1,\ldots,N$, are known in advance. In practice, these arrival times may be unknown a priori. In this section, we define an online problem based on the notion of "release time jitter" [20] to capture this uncertainty. Specifically, a_i is a random variable defined over a known interval $[a_i^-, a_i^+]$, which includes situations where expected tasks not received within a particular time interval are considered useless and are never processed (e.g., expected data that arrive too late to a processing node in a sensor network).

The uncertainty introduced by "release time jitter" makes it infeasible to obtain optimal controls based on the actual arrival times. To guarantee that deadlines are met *in all cases*, we inevitably have to make decisions based on the worst case, that is, assuming future tasks will arrive at the latest possible time a_i^+. However, the optimal solution based on the worst case analysis must be conservative. To obtain better performance, we utilize an online algorithm which allows us to reoptimize controls based on new observations made at a number of appropriately defined decision points. In particular, new information is acquired when the events characterizing the system take place, i.e., at task arrival times and at task departure times. In what follows, we choose task departure times to be these decision points, corresponding to what is known as "intertask" DVS [10] (other approaches for online control are considered in [21]). Our approach does not depend on the choice of decision points chosen. However, from a practical standpoint, updating controls upon each arrival time can be problematic when arrivals are bursty, in which case, it is even possible that the calculation of new controls takes longer than an interarrival time and this can lead to unstable behavior.

At decision points, we can observe the *actual* arrival times of all tasks in the queue and this information can be utilized to improve the optimal control. Fig. 2 is an example to illustrate how to improve optimal controls by updating arrival information. At the decision point x_{K-2}, task K has not arrived, thus, a_K, its actual arrival time, is unknown. The optimal control at x_{K-2} has to be computed based on the worst-case arrival time a_K^+. On the other hand, at the next decision point x_{K-1}, task K has arrived and is in the queue so that we can make use of a_K to evaluate the optimal controls obtained when a_K^+ is replaced by a_K.

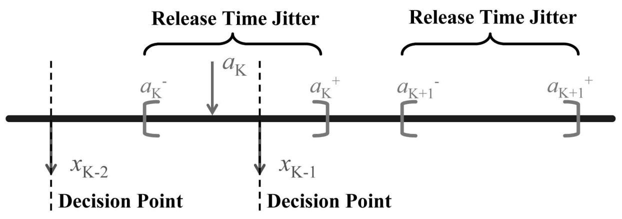


Fig. 2. Online framework example: a_K does not need to be estimated at x_{K-1}.

The observed arrival time of task i at the Kth decision point (i.e., x_{K-1}) is defined as

$$\bar{a}_i = a_i \mathbf{1}[x_{K-1} \geq a_i] + a_i^+ \mathbf{1}[x_{K-1} < a_i].$$

Then, based on these observations, we solve the following online problem at the Kth decision point:

$$\begin{aligned}
&\min_{\tau_K,\ldots,\tau_N} \left\{ J(K) = \sum\nolimits_{i=K}^{N} \mu_i \theta(\tau_i) \right\} \\
\text{s.t.} \quad & \tau_{\min} \leq \tau_i \leq \tau_{\max}, \quad i = K,\ldots,N; \\
& x_i = \max(x_{i-1}, \bar{a}_i) + \mu_i \tau_i, \quad i = K,\ldots,N; \\
& x_0 = -\infty; \quad x_i \leq d_i \quad i = K,\ldots,N.
\end{aligned}$$

It is important to note, however, that we apply *only* the optimal control for task K, since future task controls may be different at the next decision point, based on newly acquired arrival time information. Since Problem $J(K)$ above has the same form as the offline problem $\tilde{J}$ we have already solved, we can still use the busy period and critical task decomposition method to obtain its optimal solution. In fact, it is not necessary to solve Problem $J(K)$ completely: We only need to identify the first BP and its first critical task, since only the optimal control of the first task is required at the time that $J(K)$ is solved.

4.2 Complexity Analysis

Let n_b denote the size of the first BP when solving Problem $J(K)$. As mentioned above, at each decision point, the CTDA only needs to identify the first critical task with complexity $O(n_b)$ so as to obtain the optimal control *of the first task only*. In contrast, the YDSA needs to solve the whole problem related to the first BP in the worst case with complexity $O(n_b^3)$ so as to obtain the optimal control of the first task.

Let n_m denote the size of the longest BP. If there are N tasks to be processed, the overall complexity of the CTDA is $O(Nn_m)$, the same as in the offline case, while the complexity of the YDSA becomes $O(Nn_m^3)$, worse than the offline case. In the worst case $n_m = N$, the complexity of the CTDA is still $O(N^2)$ while the complexity of the YDSA becomes $O(N^4)$. Thus, the CTDA is more suitable as an online algorithm by virtue of the forward decomposition way in which it operates.

5 Simulation Results

5.1 CTDA versus YDSA

We provide some simulation results in order to illustrate the computational complexity of the CTDA and compare it

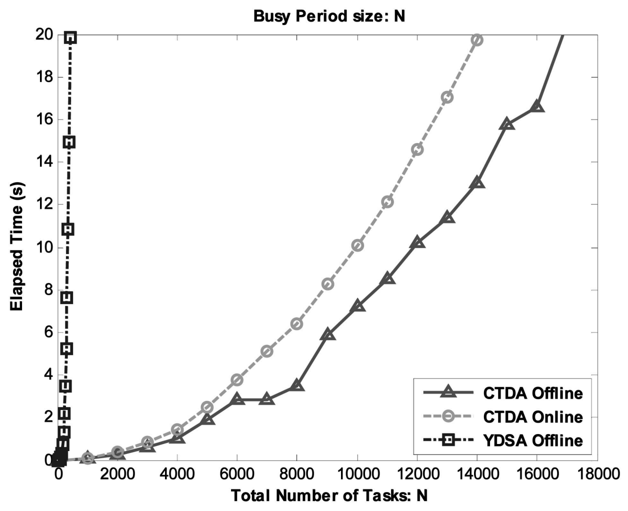


Fig. 3. $n_b = N$.

to that of the YDSA in [8]. We omit simulation results for the performance obtained under various DVS algorithms since we have formally shown that the CTDA gives optimal performance.

Define the probability that a task $i+1$ arrives after the deadline of the last task as $P(d_i < a_{i+1}) = 1/n_b$. We can use this probability to derive a process with the average BP size set to n_b. In the following, all algorithms are programmed in C++ and are executed on a computer with CPU Intel Pentium M 2.8 GHz. We compare the complexity of CTDA and YDSA based on the elapsed time (seconds) in two cases: $n_b = N$ and $n_b = 100$ (the online YDSA [8] applies the average rate heuristic, which is not applicable to non-preemptive systems but only preemptive systems; therefore, we only compare with the offline YDSA). The results are shown in Figs. 3 and 4, respectively.

In the case of $n_b = N$, we can see that the complexity of the CTDA increases quadratically with N and the complexity of the YDSA increases cubically in N, as detailed in the previous sections. We can see that the complexity of the CTDA increases linearly in N in the case of $n_b = 100$. In addition to the CTDA having a lower level of complexity compared to the YDSA in [8], note that both the offline and online versions have the same order of complexity. Thus, the CTDA can fully take advantage of an online implementation.

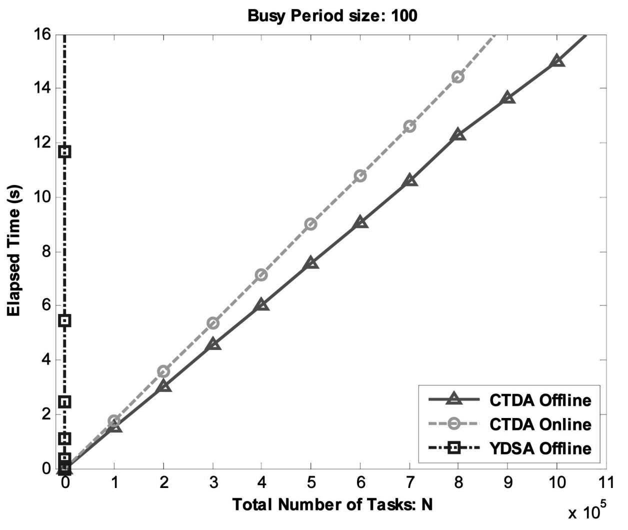


Fig. 4. $n_b = 100$.

TABLE 2
The Effect of the Size of $[a_i^-, a_i^+]$

$[a_i^-, a_i^+]$	$J_{on}(\times 10^4)$	$J_{off}(\times 10^4)$	λ
0.35	1.5074	1.4303	1.0539
0.70	1.4827	1.3737	1.0793
1.05	1.4574	1.3389	1.0885
1.40	1.4389	1.3153	1.0940

5.2 Online versus Offline

Consider the problem

$$\min_{u_1,\dots,u_N} \left\{ J = \sum_{i=1}^{N} \theta_i(u_i) = \sum_{i=1}^{N} \mu_i C_1 \left(\frac{V_t u_i}{u_i - \mu_i C_2} \right)^2 \right\}$$

$$\text{s.t.} \quad u_i \geq \frac{\mu_i C_2 V_{\max}}{V_{\max} - V_t}, \quad i = 1,\dots,N; \quad x_0 = 0;$$

$$x_i = \max(x_{i-1}, a_i) + u_k \leq d_i, \quad i = 1,\dots,N;$$

$$a_i \in [a_i^-, a_i^+], \quad a_i^+ < a_{i+1}^-, \quad i = 1,\dots,N,$$

where $V_{\max} = 5$, $V_t = 1$, $C_1 = 1$, and $C_2 = 0.1$. The parameter values selected are motivated by CMOS microprocessor power consumption data. We will assume that there is a total of 50 tasks to be processed, i.e., $N = 50$. In the simulated system operation, task arrivals are randomly generated within intervals $[a_i^-, a_i^+]$, $i = 1,\dots,50$, whose size is varied (see Table 2).

In this subsection, we apply a competitive analysis for the specific example above. We define λ as the ratio $\lambda = \frac{J_{on}}{J_{off}}$, where J_{on} is the optimal cost derived by the online method and J_{off} is the optimal cost derived by the offline method; the latter is the best that could have been achieved because the optimal controls obtained by the offline method are computed based on knowing all actual arrival times.

The effect of arrival uncertainty, i.e., the size of $[a_i^-, a_i^+]$, on λ is shown in Table 2. As seen in the table, λ is close to 1. Moreover, λ increases as the size of $[a_i^-, a_i^+]$ (the arrival time uncertainty) increases.

6 Conclusions

The problem formulated and solved in this paper is motivated by low power systems with hard real-time nonpreemptive and aperiodic tasks. We have developed a DVS algorithm for both offline and online problems with worst-case computational complexity of $O(N^2)$ by exploiting the structure of optimal sample paths in terms of identifying busy periods and "critical tasks" allowing us to efficiently decompose the original optimization problem into a set of smaller easy to solve problems.

The development of such an efficient algorithm paves the way for a variety of natural extensions. For example, we are currently investigating the effect of making decisions at task arrival times, as opposed to task departure times, which should intuitively provide additional opportunities for cost reduction in the online setting; in the offline setting, we have recently shown [21] that the static controller developed in this paper is in fact optimal even if one were to include feasible policies in which the control (voltage) may be varied while a task is executed; it turns out that such dynamic control provides no extra benefit to power management. Moreover, it is possible to relax the constraint $[a^-, a^+]$ when applying online control, allowing arrivals to occur at any time, as recently proposed in [22] using a receding horizon approach which builds on the algorithm developed in this paper.

Future work is aiming at incorporating additional uncertainty factors such as uncertain deadlines and task processing time, as well as systems that process tasks over multiple stages.

APPENDIX

Proof of Proposition 1. First, we prove that

$$\text{Iff} \quad a_{i+1} \le d_i, \quad \text{then} \quad a_{i+1} \le x_i^*. \tag{9}$$

Necessary condition: Assume on the contrary that $a_{i+1} > x_i^*$ under the optimal control τ_i^*. Then, there must exist some $\tau_i' > \tau_i^*$ such that $x_i' = a_{i+1}$. It follows from $x_{i+1} = \max(x_i, a_{i+1}) + \mu_i\tau_{i+1}$ that this increase from τ_i^* to τ_i' does not affect any other control. Obviously, $\theta(\tau_i') < \theta(\tau_i^*)$ since $\theta(\tau_i)$ is monotonically decreasing. This contradicts the optimality of τ_i^*. Thus, $a_{i+1} \le x_i^*$ must hold.

Sufficient condition: If $a_{i+1} \le x_i^*$ and $x_i^* \le d_i$ (from the feasibility constraint), it immediately follows that $a_{i+1} \le d_i$.

To complete the proof of (9), we use contrapositivity, i.e., iff $a_{i+1} > d_i$, then $a_{i+1} > x_i^*$. This implies that 1) $x_{k-1}^* < a_k$ iff $a_k > d_{k-1}$ and 2) $x_n^* < a_{n+1}$ iff $a_{n+1} > d_n$. From the definition of a BP, the result immediately follows.

Finally, $x_n^* = d_n$ easily follows from the fact that $\theta(\tau_i)$ is monotonically decreasing: Suppose τ_i^* is such that $x_n^* < d_n$. Then, there exists some $\tau_i' > \tau_i^*$ such that $x_n^* = d_n$ and $\theta(\tau_i') < \theta(\tau_i^*)$, contradicting the optimality of τ_i^*. □

Proof of Lemma 1. Let

$$\begin{aligned} L(\tau_i, \alpha_i, \beta_i, \lambda) = &\textstyle\sum_{i=k}^{n} \mu_i\theta(\tau_i) \\ &+ \textstyle\sum_{i=k}^{n-1} \alpha_i\Big(a_{i+1} - a_k - \sum_{j=k}^{i} \mu_j\tau_j\Big) \\ &+ \textstyle\sum_{i=k}^{n-1} \beta_i\Big(a_k + \sum_{j=k}^{i} \mu_j\tau_j - d_i\Big) \\ &+ \lambda\Big(a_k + \textstyle\sum_{j=k}^{n} \mu_j\tau_j - d_n\Big), \end{aligned}$$

where we have introduced multipliers $\alpha_i \ge 0$, $\beta_i \ge 0$, and λ to adjoin the constraints in Problem $Q(k,n)$ to the cost function. We get

$$\begin{aligned} \nabla_i L(\tau_i, \alpha_i, \beta_i, \lambda) = \mu_i\Big(&\theta'(\tau_i^*) - \alpha_i + \beta_i \\ &- \textstyle\sum_{j=i+1}^{n-1} \alpha_j + \sum_{j=i+1}^{n-1} \beta_j + \lambda\Big) = 0 \\ \nabla_{i+1} L(\tau_i, \alpha_i, \beta_i, \lambda) = \mu_{i+1}\Big(&\theta'(\tau_{i+1}^*) - \textstyle\sum_{j=i+1}^{n-1} \alpha_j \\ &+ \textstyle\sum_{j=i+1}^{n-1} \beta_j + \lambda\Big) = 0, \end{aligned}$$

which implies that $\theta'(\tau_i^*) - \theta'(\tau_{i+1}^*) = \alpha_i - \beta_i$. If $\tau_i^* > \tau_{i+1}^*$, then $\theta'(\tau_i^*) > \theta'(\tau_{i+1}^*)$ since $\theta(\cdot)$ is differentiable and strictly convex. It follows that $\alpha_i - \beta_i > 0$, i.e., $\alpha_i > 0$ for $\beta_i \ge 0$. Thus, the constraint $a_{i+1} \le a_k + \sum_{j=k}^{i} \mu_j\tau_j$ is active, that is, $x_i^* = a_k + \sum_{j=k}^{i} \mu_j\tau_j = a_{i+1}$. Similarly, if $\tau_i^* > \tau_{i+1}^*$, we obtain $x_i^* = d_i$. □

Proof of Proposition 2. We will prove the first assertion only, since the second one is similarly shown. Since the proof is somewhat lengthy, we divide it into four parts.

Part 1: We first establish the following two inequalities:

$$G_j(p) \le H_i^*(p) \le H_j(p), \quad j = p, \ldots, R_i(p) - 1, \tag{10}$$

$$G_j(p) \le H_i^*(p) < H_j(p), \quad j = R_i(p) + 1, \ldots, i - 1. \tag{11}$$

By the definition of $R_i(p)$,

$$H_i^*(p) \le H_j(p), \quad j = p, \ldots, R_i(p) - 1, \tag{12}$$

$$H_i^*(p) < H_j(p), \quad j = R_i(p) + 1, \ldots, i - 1, \tag{13}$$

$$H_j^*(p) = H_i^*(p), \quad j = R_i(p) + 1, \ldots, i - 1. \tag{14}$$

By assumption, $G_j(p) \le H_j^*(p)$ for all

$$j = R_i(p) + 1, \ldots, i - 1.$$

Thus, using (14), we get

$$G_j(p) \le H_i^*(p), \quad j = R_i(p) + 1, \ldots, i - 1. \tag{15}$$

Combining (13) and (15) yields inequality (11). Next, by the definition of $G_{R_i(p)}^*(p)$,

$$G_{R_i(p)}^*(p) \ge G_j(p), \quad j = p, \ldots, R_i(p) - 1. \tag{16}$$

For $R_i(p) \in \{p, \ldots, i-1\}$, $H_{R_i(p)}(p) \ge G_{R_i(p)}^*(p)$ by assumption; hence, (16) gives

$$G_j(p) \le H_{R_i(p)}(p) = H_i^*(p), \quad j = p, \ldots, R_i(p) - 1. \tag{17}$$

Combining (12) and (17) yields inequality (10).

Part 2: We shall now prove the following two inequalities:

$$\tau_{R_i(p)}^* \le H_i^*(p), \tag{18}$$

$$\tau_{R_i(p)+1}^* > H_i^*(p). \tag{19}$$

Using the equality constraint in problem $C(p)$, we have $\sum_{j=p}^{R_i(p)} \mu_j\tau_j^* = x_{R_i(p)} - s_p \le d_{R_i(p)} - s_p$. Combining this with the definitions in (6)-(8), we have

$$\begin{aligned} \textstyle\sum_{j=p}^{R_i(p)} \mu_j\tau_j^* &\le H_{R_i(p)}(p) \textstyle\sum_{j=p}^{R_i(p)} \mu_j \\ &= H_i^*(p) \textstyle\sum_{j=p}^{R_i(p)} \mu_j. \end{aligned} \tag{20}$$

We can now proceed with a contradiction argument to obtain (18). In particular, assume $\tau^*_{R_i(p)} > H^*_i(p)$. Then, (20) implies that there must exist some $t < R_i(p)$ such that

$$\tau^*_j > H^*_i(p) \quad \text{for } j = t+1, \ldots, R_i(p) \text{ and } \tau^*_t \leq H^*_i(p). \quad (21)$$

Based on the definition of $H_t(p)$ in (6), we have $d_t - s_p = H_t(p)\sum_{j=p}^{t}\mu_j$. From (21), we have $\tau^*_t < \tau^*_{t+1}$, which implies that $x^*_t = d_t$ by Lemma 1. Therefore, recalling the first constraint in problem $C(p)$, we have

$$\sum\nolimits_{j=p}^{t}\mu_j\tau^*_j = x^*_t - s_p = H_t(p)\sum\nolimits_{j=p}^{t}\mu_j.$$

Then, from (10), $\sum_{j=p}^{t}\mu_j\tau^*_j \geq H^*_i(p)\sum_{j=p}^{t}\mu_j$. However, from (21), we have $\sum_{j=t+1}^{R_i(p)}\mu_j\tau^*_j > H^*_i(p)\sum_{j=t+1}^{R_i(p)}\mu_j$ and, from (20), it follows that $\sum_{j=p}^{t}\mu_j\tau^*_j < H^*_i(p)\sum_{j=p}^{t}\mu_j$, which contradicts the inequality above.

Proceeding along the same lines, we can establish (19). From the definition of problem $C(p)$,

$$\sum_{j=p}^{R_i(p)}\mu_j\tau^*_j = x_{R_i(p)} - s_p \geq a_{R_i(p)} - s_p.$$

Combining this with the definitions in (6)-(8), we get

$$\begin{aligned}\sum\nolimits_{j=p}^{i}\mu_j\tau^*_j &\geq G_i(p)\sum\nolimits_{j=p}^{i}\mu_j,\\ \sum\nolimits_{j=p}^{R_i(p)}\mu_j\tau^*_j &\leq H_{R_i(p)}(p)\sum\nolimits_{j=p}^{R_i(p)}\mu_j\\ &= H^*_i(p)\sum\nolimits_{j=p}^{R_i(p)}\mu_j,\end{aligned}$$

which implies

$$\sum\nolimits_{j=R_i(p)+1}^{i}\mu_j\tau^*_j \geq G_i(p)\sum\nolimits_{j=p}^{i}\mu_j - H^*_i(p)\sum\nolimits_{j=p}^{R_i(p)}\mu_j. \quad (22)$$

By the assumption $G_i(p) > H^*_i(p)$, we get

$$\begin{aligned}G_i(p)\sum\nolimits_{j=p}^{i}\mu_j &> H^*_i(p)\sum\nolimits_{j=p}^{i}\mu_j\\ &= H^*_i(p)\sum\nolimits_{j=p}^{R_i(p)}\mu_j + H^*_i(p)\sum\nolimits_{j=R_i(p)+1}^{i}\mu_j,\end{aligned}$$

which implies

$$G_i(p)\sum\nolimits_{j=p}^{i}\mu_j - H^*_i(p)\sum\nolimits_{j=p}^{R_i(p)}\mu_j > H^*_i(p)\sum\nolimits_{j=R_i(p)+1}^{i}\mu_j. \quad (23)$$

Combining (22) and (23), we have

$$\sum\nolimits_{j=R_i(p)+1}^{i}\mu_j\tau^*_j > H^*_i(p)\sum\nolimits_{j=R_i(p)+1}^{i}\mu_j. \quad (24)$$

As in proving (18) above, from (11) and (24), we can similarly proceed with a contradiction argument to obtain (19).

Part 3: We finally prove that task $R_i(p)$ is a right-critical task, and also the first critical task in $C(p)$, and show that $\tau^*_j = H^*_i(p), j = p, \ldots, R_i(p)$. From (18) and (19), we have $\tau^*_{R_i(p)+1} > \tau^*_{R_i(p)}$; that is, task $R_i(p)$ is indeed right-critical. To obtain $\tau^*_p, \ldots, \tau^*_{R_i(p)}$, we need to minimize the additive cost function $\sum_{j=p}^{R_i(p)}\mu_j\theta(\tau_j)$ subject to the two constraints in problem $C(p)$. By Lemma 1, we have $x^*_{R_i(p)} = d_{R_i(p)}$. Thus, the first constraint in problem $C(p)$ becomes $\sum_{j=p}^{R_i(p)}\mu_j\tau^*_j = d_{R_i(p)} - s_p$ and, using the definition of $H_{R_i(p)}(p)$ in (6), we can rewrite it as

$$\sum\nolimits_{j=p}^{R_i(p)}\mu_j\tau^*_j = H_{R_i(p)}(p)\sum\nolimits_{j=p}^{R_i(p)}\mu_j = H^*_i(p)\sum\nolimits_{j=p}^{R_i(p)}\mu_j. \quad (25)$$

The remaining inequality constraints can similarly be written as

$$G_j(p)\sum\nolimits_{m=p}^{j}\mu_m \leq \sum\nolimits_{m=p}^{j}\mu_m\tau_m \leq H_j(p)\sum\nolimits_{m=p}^{j}\mu_m \quad (26)$$

for $j = p, \ldots, R_i(p) - 1$. Ignoring the latter constraints for the moment, set $B = H^*_i(p)\sum_{j=p}^{R_i(p)}\mu_j$ and consider the problem of minimizing $\sum_{j=p}^{R_i(p)}\mu_j\theta(\tau_j)$ subject to $\sum_{j=p}^{R_i(p)}\mu_j\tau_j = B$. Adjoining $\sum_{j=p}^{R_i(p)}\mu_j\tau_j - B$ to the cost function using a Lagrange multiplier λ, the necessary condition for optimality is

$$\mu_j\theta'(\tau_j) + \mu_j\lambda = 0, \quad j = p, \ldots, R_i(p).$$

Therefore, $\theta'(\tau_j) = -\lambda$ must hold for all $j = p, \ldots, R_i(p)$. Since $\theta(\tau_j)$ is strictly convex and differentiable, this implies that $\tau^*_p = \ldots = \tau^*_{R_i(p)}$ and it follows that $\tau^*_j = B/\sum_{j=p}^{R_i(p)}\mu_j = H^*_i(p), \ j = p, \ldots, R_i(p)$. Moreover, observe that $\tau_j = H^*_i(p), \ j = p, \ldots, R_i(p)$, satisfies the inequality constraints (26) by using (10), which gives, for $j = p, \ldots, R_i(p) - 1$,

$$G_j(p)\sum\nolimits_{m=p}^{j}\mu_m \leq H^*_i(p)\sum\nolimits_{m=p}^{j}\mu_m \leq H_j(p)\sum\nolimits_{m=p}^{j}\mu_m. \quad (27)$$

Thus, $\tau^*_j = H^*_i(p), j = p, \ldots, R_i(p)$ are indeed the optimal controls. Finally, from above, we have $\tau^*_j = H^*_i(p)$, $j = p, \ldots, R_i(p)$. Therefore, task $R_i(p)$ is also the first critical task in $C(p)$. □

Proof of Proposition 3. First, we prove that, if there exist some tasks i such that the optimal solution of Problem J, $\tau^*_i < \tau_{\min}$, then Problem $\tilde{J}$ is infeasible. Without loss of generality, assume that there exist tasks $p, \ldots, q$ $(k < p \leq q < n)$ in a BP $\{k, \ldots, n\}$ of an optimal sample path of Problem J such that

$$\tau^*_{p-1} \geq \tau_{\min}, \quad \tau^*_{q+1} \geq \tau_{\min}, \quad \tau^*_i < \tau_{\min}, \quad i = p, \ldots, q. \quad (28)$$

Using (28) and Lemma 1, we know that the optimal departure times of Problem J satisfy

$$x^*_{p-1} = a_p, \quad x^*_q = d_q. \quad (29)$$

Since $x^*_q = x^*_{p-1} + \sum_{i=p}^{q}\mu_i\tau^*_i$, combining this equation with (28) and (29), we have

$$d_q - a_p = x^*_q - x^*_{p-1} = \sum\nolimits_{i=p}^{q}\mu_i\tau^*_i < \tau_{\min}\sum\nolimits_{i=p}^{q}\mu_i. \quad (30)$$

For all possible solutions $\tau_p, \ldots, \tau_q$ of Problem $\tilde{J}$, it must hold that $\sum_{i=p}^{q} \mu_i \tau_i \le d_q - a_p$. Combining this inequality with (30) gives

$$\sum\nolimits_{i=p}^{q} \mu_i \tau_i < \tau_{\min} \sum\nolimits_{i=p}^{q} \mu_i, \tag{31}$$

which implies that there always exists at least one task $i \in [p, q]$ such that $\tau_i < \tau_{\min}$, that is, Problem $\tilde{J}$ is infeasible.

Second, we prove that, if $\tau_i^* \ge \tau_{\min}$ for $i = 1, \ldots, N$, then $\tilde{\tau}_i^* = \min(\tau_i^*, \tau_{\max})$. Since $\tau_i^* \ge \tau_{\min}$ for $i = 1, \ldots, N$, Problem $\tilde{J}$ is feasible and the removal of inequality constraints $\tau_i \ge \tau_{\min}$ will not affect the optimal solution of Problem $\tilde{J}$. Since the BPs in Problem J are decoupled from each other, we only need to show that $\tilde{\tau}_i^* = \min(\tau_i^*, \tau_{\max})$ holds in a single BP of an optimal sample path of Problem J. Without loss of generality, a single BP can be divided into several segments according to whether optimal controls exceed the threshold $\tau_{\max}$ or not. We define U as the set of segments with $\tau_i^* > \tau_{\max}$ and define L as the set of segments with $\tau_i^* \le \tau_{\max}$.

For any task i in the segments belonging to U, in order to satisfy the constraint $\tau_i \le \tau_{\max}$, the monotonicity of $\theta(\tau_i)$ implies that we must reduce τ_i^* to $\tau_{\max}$, i.e., the optimal control is $\tilde{\tau}_i^* = \tau_{\max}$. For tasks in the segments belonging to L, we assume the contiguous tasks $\{p, \ldots, q\}$ form a segment belonging to L without loss of generality. Since the segments in L must be separated by segments in U, that is, $\tau_{p-1}^* > \tau_{\max} \ge \tau_p^*$ and $\tau_q^* \le \tau_{\max} < \tau_{q+1}^*$, it must hold that task p starts at a_p and task q ends at d_q based on Lemma 1; that is, $\sum_{i=p}^{q} \tau_i^*$ has reached its maximum value $d_q - a_p$. So, τ_i^* for all $i \in \{p, \ldots, q\}$ can never increase no matter how τ_i^* changes for $i \notin \{p, \ldots, q\}$. Thus, we have $\tilde{\tau}_i^* = \tau_i^*$ for all tasks i in the segments belonging to L. We conclude from the above that $\tilde{\tau}_i^* = \min(\tau_i^*, \tau_{\max})$ for $i = 1, \ldots, N$. □

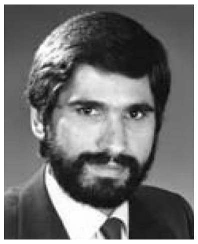
ACKNOWLEDGMENTS

This work was supported in part by the US National Science Foundation under grant DMI-0330171, by the US Air Force Office of Scientific Research under grants FA9550-04-1-0133 and FA9550-04-1-0208, by the US Department of Energy under grant DE-FG52-06NA27490, by the US Army Research Office under grant DAAD19-01-0610, by NFSC under grants 60274011 and 60574067, and by the NCET (NCET-04-0094) program of China.

REFERENCES

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A Survey on Sensor Networks," *IEEE Comm. Magazine,* vol. 40, no. 8, pp. 102-114, 2002.
[2] J. Pouwelse, K. Langendoen, and H. Sips, "Dynamic Voltage Scaling on a Low-Power Microprocessor," *Proc. Seventh Ann. Int'l Conf. Mobile Computing and Networking,* pp. 251-259, 2001.
[3] T. Pering, T. Burd, and R. Brodersen, "Dynamic Voltage Scaling and the Design of a Low-Power Microprocessor System," *Proc. Power Driven Microarchitecture Workshop,* 1998.
[4] E. Shih, S. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, and A. Chandrakasan, "Physical Layer Driven Protocol and Algorithm Design for Energy-Efficient Wireless Sensor Networks," *Proc. ACM MobiCom,* pp. 272-287, 2001.
[5] G.C. Buttazzo, *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications.* Kluwer Academic, 1997.
[6] Y.C. Cho, C.G. Cassandras, and D.L. Pepyne, "Forward Decomposition Algorithms for Optimal Control of a Class of Hybrid Systems," *Int'l J. Robust and Nonlinear Control,* vol. 11, no. 5, pp. 497-513, 2001.
[7] C.G. Cassandras, D.L. Pepyne, and Y. Wardi, "Optimal Control of a Class of Hybrid System," *IEEE Trans. Automatic Control,* vol. 46, no. 3, pp. 398-415, 2001.
[8] F. Yao, A. Demers, and S. Shenker, "A Scheduling Model for Reduced CPU Energy," *Proc. 36th Ann. Symp. Foundations of Computer Science (FOCS '95),* pp. 374-382, 1995.
[9] H. Aydin, R. Melhem, D. Mossé, and P. Mejia-Alvarez, "Power-Aware Scheduling for Periodic Real-Time Tasks," *IEEE Trans. Computers,* vol. 53, no. 5, pp. 584-600, May 2004.
[10] W. Kim, D. Shin, H.S. Yun, J. Kim, and S.L. Min, "Performance Comparison of Dynamic Voltage Scaling Algorithms for Hard Real-Time Systems," *Proc. Real-Time Technology and Applications Symp.,* pp. 219-228, 2002.
[11] K. Jeffay, D.F. Stanat, and C.U. Martel, "On Non-Preemptive Scheduling of Periodic and Sporadic Tasks," *Proc. IEEE Real-Time Systems Symp.,* pp. 129-139, 1991.
[12] J. Jonsson, H. Lonn, and K.G. Shin, "Non-Preemptive Scheduling of Real-Time Threads on Multi-Level-Context Architectures," *Proc. IEEE Workshop Parallel and Distributed Real-Time Systems,* vol. 1586, pp. 363-374, 1999.
[13] A.E. Gamal, C. Nair, B. Prabhakar, E. Uysal-Biyikoglu, and S. Zahedi, "Energy-Efficient Scheduling of Packet Transmissions over Wireless Networks," *Proc. INFOCOM,* vol. 3, nos. 23-27, pp. 1773-1782, 2002.
[14] L. Miao and C.G. Cassandras, "Optimal Transmission Scheduling for Energy-Efficient Wireless Networks," *Proc. INFOCOM,* 2006.
[15] I. Hong, D. Kirovski, G. Qu, M. Potkonjak, and M.B. Srivastava, "Power Optimization of Variable-Voltage Core-Based Systems," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems,* vol. 18, no. 12, pp. 1702-1714, 1999.
[16] V. Swaminathan and K. Chakrabarty, "Real-Time Task Scheduling for Energy-Aware Embedded System," *Proc. IEEE Real-Time Systems Symp. (Work-in-Progress Session),* 2000.
[17] J. Lee, "Optimization of Power Consumption in Low Power System Using Hybrid System," *Proc. 14th Common Intrusion Specification Language Winter Workshop,* 2001.
[18] C.G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems.* Kluwer Academic Publishers, 1999.
[19] B. Zhai, D. Blaauw, D. Sylvester, and K. Flautner, "Theoretical and Practical Limits of Dynamic Voltage Scaling," *Proc. 41st Ann. Conf. Design Automation,* pp. 868-873, 2004.
[20] J.W.S. Liu, *Real-Time Systems.* Prentice Hall, 2000.
[21] L. Miao and C.G. Cassandras, "Optimality of Static Control Policies in Some Discrete Event Systems," *IEEE Trans. Automatic Control,* vol. 50, no. 9, pp. 1427-1431, Sept. 2005.
[22] L. Miao and C.G. Cassandras, "Receding Horizon Control for a Class of Discrete Event System with Real-Time Constraints," *Proc. 44th IEEE Conf. Decision and Control,* pp. 7714-7719, Dec. 2005.

Jianfeng Mao received the BE degree in automatic control and the ME degree in control theory and applications from Tsinghua University, Beijing, China, in 2001 and 2004, respectively. Currently, he is a PhD candidate in manufacturing engineering at Boston University, Massachusetts. He specializes in the areas of modeling and optimization of complex system with application to sensor networks and manufacturing systems. He is a student member of the IEEE.



Christos G. Cassandras received the BS degree from Yale University, New Haven, Connecticut, the MSEE degree from Stanford University, California, and the SM and PhD degrees from Harvard University, Cambridge, Massachusetts, in 1977, 1978, 1979, and 1982, respectively. From 1982 to 1984, he was with ITP Boston, Inc., where he worked on the design of automated manufacturing systems. From 1984 to 1996, he was a faculty member in the Department of Electrical and Computer Engineering, University of Massachusetts, Amherst. Currently, he is professor of manufacturing engineering and a professor of electrical and computer engineering at Boston University, Massachusetts and a founding member of the Center for Information and Systems Engineering (CISE). He specializes in the areas of discrete event and hybrid systems, stochastic optimization, and computer simulation with applications to computer networks, sensor networks, manufacturing systems, transportation systems, and command-control systems. He has published more than 200 papers in these areas and two textbooks, one of which was awarded the 1999 Harold Chestnut Prize by the IFAC. Dr. Cassandras is currently the editor-in-chief of the *IEEE Transactions on Automatic Control* and has served on several editorial boards and as a guest editor for various journals. He is a member of the IEEE Control Systems Society Board of Governors and a recipient of several awards, most recently the IEEE Control System Society's 2006 Distinguished Member Award. He is a member of Phi Beta Kappa and Tau Beta Pi and a fellow of the IEEE.

Qianchuan Zhao received the BE degree in automatic control in July 1992, the BS degree in applied mathematics in July 1992, and PhD degree in control theory and its applicaitons in July 1996, all from Tsinghua University, Beijing, China. He is currently a professor and the associate director of the Center for Intelligent and Networked Systems (CFINS) in the Department of Automation at Tsinghua University, Beijing, China. He was a visiting scholar at Carnegie Mellon University and Harvard University in 2000 and 2002, repsectively. He was a visiting professor at Cornell University in 2006. His research interests include discrete event dynamic systems (DEDS) theory and applications, optimalization of complex systems, and wireless sensor networks. He is an associate editor for the *Journal of Optimization Theory and Applications* (JOTA) and a member of the IEEE.