Optimal Control of Discrete Event Systems with Weakly Hard Real-Time Constraints

Shixin Zhuang and **Christos G. Cassandras** [1] [2]

sxzhuang@bu.edu, cgc@bu.edu

September 23, 2008

Abstract

We consider Discrete Event Systems (DES) that can dynamically allocate resources in order to process tasks with real-time constraints. In the case of "weakly hard" constraints, a fraction of tasks is allowed to violate them, as long as m out of any k consecutive tasks meet their respective constraints. This is a generalization of a system with purely hard real-time constraints where $m = k = 1$. For non-preemptive and aperiodic tasks, we formulate an optimization problem where task processing times are controlled so as to minimize a cost function while guaranteeing that a "weakly hard" criterion is satisfied. We establish a number of structural properties of the solution to this problem which lead to an efficient algorithm that does not require any explicit nonlinear programming problem solver. The low complexity of this algorithm makes it suitable for on-line applications. Simulation examples illustrate the performance improvements in such optimally controlled systems compared to ad hoc schemes.

Notes

[1]**Affiliation of Authors:** Dept. of Manufacturing Engineering and Center for Information and Systems Engineering, Boston University, Brookline, MA 02446

[2]The authors' work is supported in part by the National Science Foundation under Grants DMI-033017 and EFRI-0735974, by AFOSR under grants FA9550-07-1-0213 and FA9550-07-1-0361, and by DOE under Grant DE-FG52-06NA27490.

1 Introduction

A large class of Discrete Event Systems (DES) involves resources which must be dynamically allocated to tasks according to certain operating specifications. The basic modeling block for such DES is a single-server queueing system operating on a first-come-first-served basis, whose dynamics are given by the well-known max-plus equation

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

where a_i is the arrival time of task $i = 1, 2, \ldots$, x_i is the time when task i completes service, and u_i is a controllable service time. Examples arise in manufacturing systems, where the operating speed of a machine can be controlled to trade off between energy costs and requirements on timely job completion (Pepyne and Cassandras, 2000); in computer systems, where the CPU speed can be controlled to ensure that certain tasks meet specified execution deadlines (Liu, 2000); and in wireless networks where severe battery limitations call for new techniques aimed at maximizing the lifetime of such a network (Miao and Cassandras, 2006),(Gamal et al., 2002). A particularly interesting set of problems arises when such systems are subject to *real-time constraints*; specifically, the requirement $x_i \leq d_i$ for a given "deadline" d_i associated with task i. In order to meet such constraints, the processing rate of tasks may have to be increased imposing in turn additional costs on the controls u_i. This gives rise to optimization problems of the form

$$\min_{u_1,\cdots,u_K} \sum_{i=1}^{K} \theta_i(u_i) \tag{2}$$

$$\text{s.t.} \quad x_i = \max(a_i, x_{i-1}) + u_i \leq d_i, \qquad i = 1, ..., K$$

Here, $\theta_i(u_i)$ is a given cost function, $u_i \geq 0$, and all a_i and d_i are known over a total of K tasks. In general, a control u_i may be time-varying while task i is in process over the interval $[\max(a_i, x_{i-1}), x_i)$. However, as shown in (Miao and Cassandras, 2005), when $\theta_i(\cdot)$ is monotonically decreasing and convex, then the optimal control of each task is constant over this interval. We will consider such cases with u_i varying only over the task index i. Systems which process tasks with real-time constraints have been extensively studied, mostly in the computer science literature where interest is largely focused on periodic and preemptive tasks, e.g., (Yao et al., 1995),(Aydin et al., 2004), although nonpreemptive tasks are also considered (Jeffay et al., 1991),(Jonsson et al., 1999). Nonpreemptive aperiodic tasks are of particular interest in wireless communications where nonpreemptive scheduling is necessary to execute aperiodic packet transmission tasks which also happen to be highly energy-intensive (Gamal et al., 2002),(Miao and Cassandras, 2006),(Mao et al., 2007).

In general, (2) is a hard nonlinear optimization problem, complicated by the inequality constraints $x_i \leq d_i$ and the nondifferentiable max operator involved. Nonetheless, it was shown in (Mao et al., 2007) that when $\theta_i(u_i)$ is convex and differentiable the solution to such problems is characterized by attractive structural properties leading to a highly efficient *Critical Task Decomposition Algorithm* (CTDA). The CTDA completely eliminates the need for a numerical solver of (2) and reduces the solution to a simple scalable procedure for identifying a set of "critical" tasks in $\{1, \ldots, K\}$. The efficiency and scalability of the CTDA are crucial for applications where small, inexpensive, often wireless devices are required to perform on-line computations with minimal on-board resources. An extension to a two-stage problem with hard real-time end-to-end constraints was presented in (Mao and Cassandras, 2007b) and, more recently, it was shown that it is also possible to derive an efficient algorithm for solving similar M-stage ($M \geq 2$) problems with hard end-to-end constraints (Mao and Cassandras, 2006).

An alternative to the *hard* real-time constraints $x_i \leq d_i$ for all $i = 1, ..., K$, in (2) is a DES with *soft* constraints where these inequalities are removed and the cost function above consists of two parts (Cassandras and Zhuang, 2005): a cost on the resource which is increasing in the processing time, and a cost imposed on tasks that violate the constraint $x_i \leq d_i$. This is appropriate for applications which can tolerate occasional deadline misses of real-time tasks. However, this approach cannot rule out the possibility that a large number of deadline misses occur in a short period of time; this may cause the loss of important information and possibly render the system inoperable.

In this paper, we consider a third alternative referred to as a *weakly hard* real-time system (Bernat et al., 2001), in which only a fraction of tasks is required to meet assigned deadlines. In such systems, it is acceptable to occasionally miss some deadlines. However, the way in which tasks missing deadlines is distributed over time is important: if tasks i and j miss their deadlines while all $k = i+1, \ldots, j-1$ meet theirs, then the effect of such misses may be tolerated as long as $(j-i)$ is sufficiently large. Thus, rather than specifying a "maximum allowable loss fraction" (or "minimum completion ratio"), an alternative model (Ramanathan and Hamdaoui, 1995) is known as the (m, k)-*firm deadline* scheme. In this scheme, the requirement is that m out of any k consecutive tasks meet their respective deadlines, where m and k are two positive integers with $m \leq k$. The scheme encompasses a wide range of tolerances to deadline misses by properly selecting the values of m and k. In particular, the original hard deadline requirement corresponds to the $(1,1)$-firm deadline scheme, while a soft deadline requirement with a bound b on the fraction of deadline misses can be approximated by picking a large value of k and choosing m such that $(k-m)/k \simeq b$. Clearly, the (m, k)-firm deadline scheme is less stringent than the hard real-time case, but more restrictive than imposing soft deadlines.

Most work considering such weakly hard real-time constraints focuses on periodic tasks and constant speed processors, e.g., (Ramanathan, 1999),(Bernat and Burns, 1997),(Hua and Qu, 2004),(Quan et al., 2004),(Bernat et al., 2001). Recent work in (Quan et al., 2004) proposed a scheduler to reduce the energy cost of periodic tasks with weakly hard constraints on a processor with variable speed. In our work, motivated by applications such as intrusion detection and data collected through sensor networks, we consider arbitrary *aperiodic* tasks and seek controls that provably minimize energy costs. Our analysis incorporates the case of periodic tasks as well. We assume that all task arrival and deadline information is available and we represent the (m, k)-firm deadline scheme by partitioning the task set as in (Ramanathan and Hamdaoui, 1995) into a *mandatory* subset, containing all tasks required to be processed and meet their deadlines, and an *optional* subset containing the remaining tasks that may be discarded (Ramanathan, 1997) and have no explicit timing requirements when processed. In our model, tasks in the optional subset are allowed to be processed on a best-effort basis (and are not discarded), but we retain the terminology used in (Ramanathan and Hamdaoui, 1995) for convenience. We formulate an optimization problem by modifying (2) appropriately and solve it through an efficient procedure that does not require an explicit solution of the problem using nonlinear programming; instead, we identify and exploit several decomposition properties of the optimal solution and we make use of the CTDA (Mao et al., 2007) mentioned earlier. The CTDA is based on two decomposition properties of an optimal state trajectory. Specifically, the state trajectory consists of a number of "busy periods" (formally defined in Section 3) whose optimal controls are decoupled from each other. Further, it can be shown that each busy priod consists of "blocks" whose start and end points are defined by "critical tasks". Within each block, it is optimal to use a fixed, easily computed, service rate. Thus, the entire optimal solution is reduced to identifying these critical tasks and the CTDA provides an explicit, computationally efficient algorithm to accomplish this without invoking any nonlinear optimization problem solver. Along similar lines, for the problem in this paper we show that an optimal state trajectory can be decomposed into segments whose starting and ending points can be uniquely determined through the known arrival time and deadline requirements of the mandatory set tasks. This decomposes the original problem into a number of smaller and simpler problems, one for each such segment.

The paper is organized as follows. In Section 2, we formulate the optimization problem with mandatory set constraints. Section 3 establishes the properties of an optimal solution, leading to a number of smaller and simpler optimization problems whose explicit solution is discussed in Section 4. In Section 5, we provide simulation examples illustrating the improvements in performance obtained by our approach compared to other schemes in terms of both cost saving and the Quality of Service (QoS) that can be achieved. We conclude with Section 6.

2 Problem Formulation

We consider a single-stage DES, where a sequence of K tasks is characterized by known arrival times $0 \leq a_1 \leq \cdots \leq a_K$ and deadlines $d_1, \cdots, d_K$. Moreover, tasks are differentiated by their "size" which is modeled through the number of operations N_i associated with task i (e.g., the number of bits in a packet). Tasks are processed on a first-come-first-served basis by a non-preemptive server (alternative orderings can be considered by simply re-indexing tasks). We concentrate on controlling directly the service times of all i (equivalently, we may think of controlling the service rates $\rho_i = N_i/u_i$). Let x_i be the service completion time of task i, which satisfies the max-plus equation (1), where, by assumption, $x_0 = 0$. The cost associated with task i is $\theta_i(u_i) = N_i\theta(u_i/N_i)$ for $i = 1, \ldots, K$, where $\theta(\cdot)$ is a cost dependent only on the characteristics of the server. Thus, θ_i captures the cost of completing all N_i operations of task i, while θ represents the per-operation cost of the server when operating with rate N_i/u_i.

We represent the (m, k)-firm deadline scheme described earlier by partitioning the task set $\{1, \ldots, K\}$ into a *mandatory* subset, containing all tasks required to be processed and meet their deadlines, and an *optional* subset containing the remaining tasks; the latter have no explicit timing requirements when processed or they may be discarded; in our model we require them to be processed while retaining the "optional" terminology. This partition is specified and the deadlines of tasks tagged as optional are simply ignored and set to ∞ for the purpose of our analysis. Specifically, we define a "mandatory task set"

$$\mathcal{M} = \{m : x_m \leq d_m < \infty,\ m = 1, \ldots, K\}$$

and an "optional task set", $\bar{\mathcal{M}}$, containing all remaining task indices.

We assume that the server has a processing rate with a lower bound $1/\eta > 0$ and upper bound $1/\gamma > 1/\eta$. This implies that the controllable service time u_i is constrained by $\gamma N_i \leq u_i \leq \eta N_i$ with N_i as defined above. We will also assume that for all $m \in \mathcal{M}$ the deadline constraints are active in the sense that the "best case" completion time (i.e., when $m \in \mathcal{M}$ is processed upon arrival with maximum (least costly) processing time $a_m + \eta N_m$) exceeds its deadline d_m. Thus, we have:

Assumption 1: For all $m \in \mathcal{M}$,

$$d_m \leq a_m + \eta N_m, \quad m \in \mathcal{M} \tag{3}$$

Note that if Assumption 1 were to be violated, given the objective of minimizing a cost monotonically decreasing in u_i, tasks $m \in \mathcal{M}$ would behave like those in the optional set and be processed at the minimal rate available when there is no other mandatory task in the system.

Finally, we make the following assumption regarding the cost functions $\theta_i(u_i) = N_i \theta(u_i/N_i)$:

Assumption 2: $\theta(u)$ is positive, continuously differentiable, strictly convex, and monotonically decreasing in u.

This assumption is consistent with most applications of interest. For instance, in manufacturing systems the cost of operating a machine is monotonically decreasing and convex in the processing time of a part (Pepyne and Cassandras, 2000); in embedded devices, the processing time of a task is a convex monotonically decreasing function of the voltage applied to its CPU and the energy expended is monotonically decreasing and convex in the processing time of a task (Mao et al., 2007), (Cassandras and Zhuang, 2005). For a typical CMOS microprocessor, an extensive discussion found in (Mao et al., 2007) points to the fact that the interval over which the processor speed is controlled possesses these properties. The numerical example we will discuss in Section 5 is precisely motivated by such microprocessors. Finally, note that differentiability of $\theta(\cdot)$ is not required; it was included in (Mao et al., 2007) in order to simplify some of the technical proofs and we shall include it as well since some of our results depend on these proofs.

We can now formulate the following problem (denoted by P), with controls $\{u_1, \cdots, u_K\}$ and departure times (state variables) $\{x_1, \cdots, x_K\}$:

$$P : \min_{u_1,\cdots,u_K} \sum_{i=1}^{K} \theta_i(u_i) \tag{4a}$$

$$\text{s.t. } x_i = \max\{a_i, x_{i-1}\} + u_i \tag{4b}$$

$$\gamma N_i \le u_i \le \eta N_i, \quad i = 1, \cdots, K \tag{4c}$$

$$x_m \le d_m, \ m \in \mathcal{M} \tag{4d}$$

The optimal controls will be denoted by $\{u_1^*, \cdots, u_K^*\}$ and the corresponding service completion times (defining the optimal state trajectory) by $\{x_1^*, \cdots, x_K^*\}$. Note that this problem is an extension of (2) where all tasks have real-time constraints and thus inherits its difficulties, such as the high dimensionality of the control vector when K is large and the nondifferentiability of the constraints caused by the max operator. In the next section, we identify decomposition properties in the same spirit as those obtained for (2) in (Mao et al., 2007), which will lead to an efficient solution approach that overcomes these two difficulties.

We should also point out that problem P may not have a feasible solution. In this paper, we assume that at least one feasible solution exists. If that is not the case, then a separate admission control problem has to precede our analysis so as to eliminate certain tasks whose deadlines cannot be met and lead to a feasible problem as illustrated in Figure 1; some recent results on this associated problem are presented in (Mao and Cassandras, 2007a).

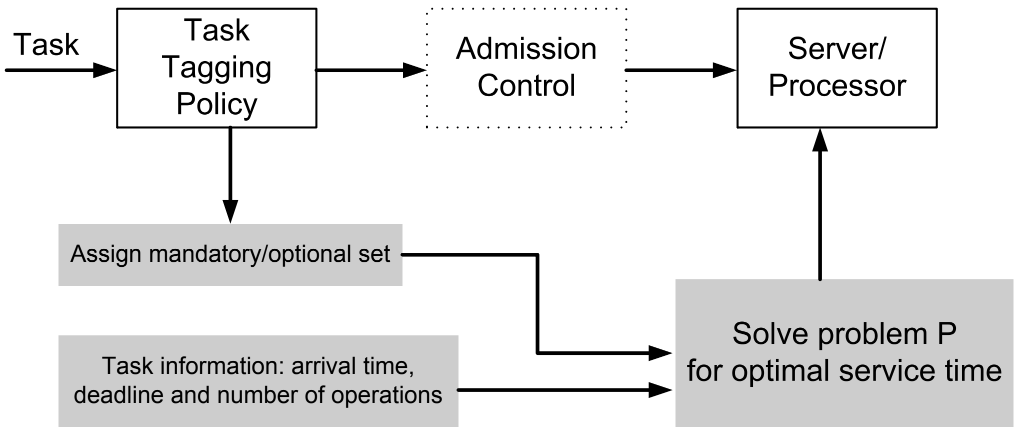
Task
Task Tagging Policy
Admission Control
Server/ Processor
Assign mandatory/optional set
Task information: arrival time, deadline and number of operations
Solve problem P for optimal service time

Figure 1: Optimal control procedure for weakly hard real-time problems

3 Optimal State Trajectory Properties

As in earlier work (Mao et al., 2007) dealing with problem (2), we begin with the observation that any state trajectory of the DES under study can be decomposed into *busy periods* separated by idle periods. In particular:

Definition: A *Busy Period* (BP) is an interval $[a_k, x_n]$ defined by a sequence of tasks $\{k, \cdots, n\}$, $1 \leq k \leq n \leq K$, such that the following three conditions are satisfied: (i) $x_{k-1} < a_k$, (ii) $x_n < a_{n+1}$, and (iii) $x_i \geq a_{i+1}$, for all $i = k, \cdots, n-1$.

Decomposing a state trajectory into BPs allows us to decouple the optimal controls of tasks in one BP from those in any other BP. This will reduce the solution of problem P to the solution of a set of simpler problems of lower dimension, provided that we can identify each BP in terms of a beginning task k and ending task n, respectively. Before identifying the BP structure in an optimal state trajectory of problem P, we first focus on the decoupling property mentioned above. Let us consider the optimization problem $P(k,n)$ defined as follows:

$$P(k,n): \min_{u_k,\cdots,u_n} \sum_{i=k}^{n} \theta_i(u_i)$$

$$\text{s.t. } x_i = a_k + \sum_{j=k}^{i} u_j \geq a_{i+1}, \quad i = k,\cdots,n-1 \tag{5a}$$

$$\gamma N_i \leq u_i \leq \eta N_i, \quad i = k,\cdots,n \tag{5b}$$

$$x_m \leq d_m, \ m \in \mathcal{M} \cap \{k,\cdots,n\} \tag{5c}$$

Let the solution of this problem be denoted by $u_i^*(k,n)$, $i = k,\cdots,n$. We can then show that the solution of problem P is identical to the collection of solutions of problems $P(k,n)$, one for each BP on the optimal state trajectory.

Theorem 1 *If tasks $\{k,\cdots,n\}$ constitute a single BP in the optimal sample path, then $u_i^* = u_i^*(k,n)$, $i = k,\cdots,n$.*

Proof. Proceeding by contradiction, assume that $\{u_i^*(k,n)\}$, $i = k,\cdots,n$, is the solution of problem $P(k,n)$ but $u_i^* \neq u_i^*(k,n)$ for at least some $i \in \{k,\cdots,n\}$. We can define a vector $\hat{u} \in R^K$ such that $\hat{u} = \{u_1^*,\cdots,u_{k-1}^*,u_k^*(k,n),\cdots,u_n^*(k,n),u_{n+1}^*,\cdots,u_K^*\}$ with corresponding service completion times $\{\hat{x}_1,\cdots,\hat{x}_K\}$. We first show $\hat{u}$ is a feasible solution for P. Since u_i^*, $i = 1,\cdots,K$ is a solution of problem P and tasks $\{k,\cdots,n\}$ constitute a single BP in the optimal state trajectory, we have $x_{k-1}^* < a_k$ and $x_n^* < a_{n+1}$ and using (4b):

$$\hat{x}_i = x_i^*, \quad i \in \{1,\cdots,k-1\} \cup \{n+1,\cdots,K\} \tag{6}$$

$$\hat{x}_i = a_k + \sum_{j=k}^{i} u_j^*(k,n), \quad i \in \{k,\cdots,n\} \tag{7}$$

We can see that all $i \in \{1,\cdots,k-1\} \cup \{n+1,\cdots,K\}$ satisfy the constraints (4a)-(4d), while the constraint (5a) of problem $P(k,n)$ guarantees that (7) holds. In addition, the constraints (5b)-(5c) of $P(k,n)$ and (4c)-(4d) of P are identical.

Since the cost function is differentiable and strictly convex, we know that $\{u_1^*, \cdots, u_K^*\}$ is a unique solution of problem P. Hence, we have $J(\hat{u}) > J(u^*)$, that is

$$\begin{aligned} J(\hat{u}) - J(u^*) &= \left(\sum_{i=1}^{k-1} \theta_i(u_i^*) + \sum_{i=k}^{n} \theta_i(u_i^*(k,n)) + \sum_{i=k+1}^{K} \theta_i(u_i^*) \right) - \left(\sum_{i=1}^{K} \theta_i(u_i^*) \right) \\ &= \sum_{i=k}^{n} \theta_i(u_i^*(k,n)) - \sum_{i=k}^{n} \theta_i(u_i^*) \\ &> 0 \end{aligned}$$

which implies that

$$\sum_{i=k}^{n} \theta_i(u_i^*(k,n)) > \sum_{i=k}^{n} \theta_i(u_i^*) \tag{8}$$

It is obvious that all $\{u_i^*\}$, $i = k, \cdots, n$, are feasible for problem $P(k,n)$. Since the inequality (8) contradicts the assumption that $\{u_i^*(k,n)\}$, $i = k, \cdots, n$ is the optimal solution of problem $P(k,n)$, it follows that $u_i^* = u_i^*(k,n)$ for all $i = k, \cdots, n$. $\blacksquare$

The implication of Theorem 1 is the following: if we can identify the BP structure, then the optimal solution can be obtained by solving a set of simpler (no max operation involved) lower-dimensional problems, one for each BP, as defined through $P(k,n)$. Thus, we concentrate next on this objective.

In the DES operating under (m,k)-firm deadline constraints, there are four types of BPs:

(i) A BP $\{k, \cdots, n\}$ such that $i \in \mathcal{M}$ for all $i \in \{k, \cdots, n\}$,

(ii) A BP $\{k, \cdots, n\}$ such that $i \in \bar{\mathcal{M}}$ for all $i \in \{k, \cdots, n\}$,

(iii) A BP $\{k, \cdots, n\}$ ending with $n \in \mathcal{M}$, but $i \in \bar{\mathcal{M}}$ for at least some $i \in \{k, \cdots, n-1\}$, and

(iv) A BP $\{k, \cdots, n\}$ ending with $n \in \bar{\mathcal{M}}$, but $i \in \mathcal{M}$ for at least some $i \in \{k, \cdots, n-1\}$.

In what follows, we derive a number of properties associated with these different types of

BPs, based on which we will be able to determine the BP structure without explicitly solving the problem P.

The first lemma below asserts that a task $i \in \mathcal{M}$ is the last task of a BP if its deadline occurs before the next task arrival time, regardless of the existence of optional set tasks in this BP. Otherwise, it does not end a BP.

Lemma 1 *Let $i \in \mathcal{M}$.*

1. *Task i ends a BP and $x_i^* = d_i$, if and only if $d_i < a_{i+1}$.*

2. *Task i does not end a BP, i.e., $x_i^* \geq a_{i+1}$, if and only if $a_{i+1} \leq d_i$.*

Proof.

1. Since task $i \in \mathcal{M}$, it satisfies $x_i^* \leq d_i$. If $d_i < a_{i+1}$, then $x_i^* < a_{i+1}$. Thus, task i ends a BP by definition. Therefore, there is no arrival during the processing of i. Moreover, since by Assumption 2 the cost function $\theta_i(u_i)$ is strictly decreasing in u_i, it follows that u_i^* is limited only by the constraints (4c) and (4d). If (4c) is active, then $u_i^* = \eta N_i$ and

$$x_i^* = \max\{a_i, x_{i-1}\} + \eta N_i \geq a_i + \eta N_i$$

   By Assumption 1, this implies that $x_i^* \geq d_i$ and since (4d) must hold, we conclude that $x_i^* = d_i$. On the other hand, if (4d) is active instead, then we immediately get $x_i^* = d_i$. Conversely, if i ends a BP, then, by the BP definition, we have $x_i^* < a_{i+1}$. Since, in addition, $x_i^* = d_i$, we get $d_i < a_{i+1}$.

2. Suppose $a_{i+1} \leq d_i$. We proceed by contradiction and assume that $x_i^* < a_{i+1}$. Then, by definition, task i ends a BP in the optimal state trajectory and $x_i^* = d_i$ by the same argument as in part 1. It follows that $a_{i+1} > d_i$, which contradicts $a_{i+1} \leq d_i$. Thus, i cannot end a BP. Conversely, if $a_{i+1} \leq x_i^*$ note that $x_i^* \leq d_i$ since $i \in \mathcal{M}$, implying that $a_{i+1} \leq x_i^* \leq d_i$.

■

Using this lemma directly leads to the proof of Theorem 2 allowing us to identify a BP of type (i), i.e., one that contains only mandatory set tasks.

Theorem 2 *Let $\{k, \cdots, n\} \subseteq \mathcal{M}$. Tasks $\{k, \cdots, n\}$ constitute a BP in the optimal sample path if and only if $x^*_{k-1} < a_k$, $a_{i+1} \le d_i$, for $i = k, \cdots, n-1$, and $d_n < a_{n+1}$.*

Proof. If, for $\{k, \cdots, n\} \subseteq \mathcal{M}$, $a_{i+1} \le d_i$ holds for $i = k, \cdots, n-1$, we know that $\{k, \cdots, n\}$ belong to one BP from part 2 of Lemma 1. Since $d_n < a_{n+1}$, by part 1 of Lemma 1 the BP ends with n and $x^*_n = d_n$. Moreover, task k begins this BP by definition since $x^*_{k-1} < a_k$. The converse follows from Lemma 1 and the definition of a BP. ■

Note that applying this theorem to identify a BP with $\{k, \cdots, n\} \subseteq \mathcal{M}$ requires knowledge of the optimal service completion time x^*_{k-1} of task $k-1$, which, except for $x^*_0 = 0$, is obviously not known in advance. However, we can always check the condition $x^*_{k-1} < a_k$ without requiring the solution of P. In the case where $(k-1) \in \mathcal{M}$, it follows from Lemma 1 that $x^*_{k-1} = d_{k-1} < a_k$. The case where $(k-1) \in \bar{\mathcal{M}}$ will be discussed later in this section.

Let us now consider a BP formed by tasks $\{k, \cdots, n\}$ containing at least some $i \in \bar{\mathcal{M}}$, $k \le i \le n$. Let $k \le p \le q \le n$ such that $\{p, \cdots, q\} \subseteq \bar{\mathcal{M}}$, $(p-1) \in \mathcal{M}$ and $(q+1) \in \mathcal{M}$. When $k = p$ and $q = n$, then $\{k, \cdots, n\}$ defines a type (ii) BP with all optional set tasks. When $k < p \le q = n$, an optional set task ends this BP and $(p-1) \in \mathcal{M}$, which implies the BP is of type (iv). If $k < p \le q < n$, we have $\{p-1, q+1\} \subseteq \mathcal{M}$. Looking at the last case in which a sequence of optional set tasks is preceded and followed by mandatory set tasks, the following lemmas present some properties of the optimal controls in such BPs.

Lemma 2 *Let tasks $\{k, \cdots, n\}$ constitute a single BP in the optimal state trajectory of problem P and let contiguous tasks $\{p, \cdots, q\}$ be such that $\{p, \cdots, q\} \subseteq \bar{\mathcal{M}} \cap \{k, \cdots, n\}$. Then,*

$$N_i/u^*_i \le N_{i+1}/u^*_{i+1}, \quad i = p, \cdots, q-1.$$

Proof. Proceeding by contradiction, assume that

$$N_i/u_i^* > N_{i+1}/u_{i+1}^*$$

for some i, $p \leq i < q$. A feasible solution of problem P, $\{u_1^*, \cdots, u_i', u_{i+1}', \cdots, u_K^*\}$, can be found such that all other values remain the same but u_i^* and u_{i+1}^* are replaced by u_i' and u_{i+1}' with

$$u_i^* + u_{i+1}^* = u_i' + u_{i+1}'$$

and

$$N_i/u_i^* > N_i/u_i' = N_{i+1}/u_{i+1}' > N_{i+1}/u_{i+1}^* \tag{9}$$

First, let us show that $\{u_1^*, \cdots, u_i', u_{i+1}', \cdots, u_K^*\}$ is indeed feasible for problem P. Since $1/\gamma \geq N_i/u_i^* > N_{i+1}/u_{i+1}^* \geq 1/\eta$ holds, it follows that $1/\gamma \geq N_i/u_i' = N_{i+1}/u_{i+1}' \geq 1/\eta$. Since $u_i^* + u_{i+1}^* = u_i' + u_{i+1}'$ and all other u_j^*, $j \neq i, i+1$, remain the same, we have $x_m' = x_m^* \leq d_m$, for all $m \in \mathcal{M}$.

Next, we show that

$$\theta_i(u_i') + \theta_{i+1}(u_{i+1}') < \theta_i(u_i^*) + \theta_{i+1}(u_{i+1}^*) \tag{10}$$

To establish this inequality, recall that $\theta_i(u_i) = N_i\theta(u_i/N_i)$ and consider the optimization problem

$$\begin{aligned} &\min_{v_i, v_{i+1}} N_i\theta(v_i) + N_{i+1}\theta(v_{i+1}) \\ &\text{s.t.} \quad N_i v_i + N_{i+1} v_{i+1} = C \end{aligned}$$

where $v_i = u_i/N_i$. Adjoining the constraint to the cost function using a Lagrange multiplier

λ leads to the necessary condition for optimality

$$N_j \theta'(v_j) + N_j \lambda = 0, \quad j = i, i+1$$

where $\theta'(\cdot)$ is the derivative of $\theta(\cdot)$ with respect to its argument. Therefore, $\theta'(v_i) = \theta'(v_{i+1}) = -\lambda$. Moreover, by Assumption 2, $\theta(\cdot)$ is convex and differentiable, so that the solution to this problem satisfies $v_i = v_{i+1} = \frac{C}{N_i + N_{i+1}}$, i.e., $u_i = \frac{CN_i}{N_i + N_{i+1}}$ and $u_{i+1} = \frac{CN_{i+1}}{N_i + N_{i+1}}$. In other words, $\theta_i(u_i) + \theta_{i+1}(u_{i+1})$ with $u_i + u_{i+1} = C$ is minimized by u_i, u_{i+1} such that

$$N_i/u_i = N_{i+1}/u_{i+1} = (N_i + N_{i+1})/C$$

which is satisfied by u'_i, u'_{i+1} in (9), hence establishing (10).

Since all other u^*_i, $i = 1, \cdots, i-1, i+2, \cdots, K$, remain unchanged, we obtain

$$\begin{aligned} &\sum_{j=1}^{i-1} \theta_j(u^*_j) + \sum_{j=i+2}^{K} \theta_j(u^*_j) + \theta_i(u'_i) + \theta_{i+1}(u'_{i+1}) \\ &< \sum_{j=1}^{i-1} \theta_j(u^*_j) + \sum_{j=i+2}^{K} \theta_j(u^*_j) + \theta_i(u^*_i) + \theta_{i+1}(u^*_{i+1}) \end{aligned}$$

which contradicts the assumption that $\{u^*_1, \cdots, u^*_K\}$ is the optimal solution. Hence, we prove that $N_i/u^*_i \le N_{i+1}/u^*_{i+1}$ must hold for all $i = p, \cdots, q-1$. ■

This result characterizes the processing rate relationship between adjacent optional set tasks and it helps us establish the next result pertaining to BPs on the optimal state trajectory that end with a sequence of optional set tasks.

Lemma 3 *Let tasks $\{k, \cdots, n\}$ constitute a single BP in the optimal sample path and contiguous tasks $\{p, \cdots, n\}$ be such that $\{p, \cdots, n\} \subseteq \bar{\mathcal{M}} \cap \{k, \cdots, n\}$. Then,*

$$u^*_i = \eta N_i, \quad i = p, \cdots, n.$$

Proof. Since $n \in \bar{\mathcal{M}}$ and, by Assumption 2, $\theta_n(u_n)$ is strictly decreasing, it follows that u_n^* is limited only by the constraint (4c). Since there is no task arriving during the processing of task n (because n ends a BP), the constraint is active for task n, i.e., $u_n^* = \eta N_n$.

Since tasks $n-1$ and n satisfy $\{n-1, n\} \subseteq \bar{\mathcal{M}} \cap \{k, \cdots, n\}$, we have $N_{n-1}/u_{n-1}^* \leq N_n/u_n^* = 1/\eta$ from Lemma 2, hence $u_{n-1}^* \geq \eta N_{n-1}$. The constraints in (4c) require that $u_{n-1}^* \leq \eta N_{n-1}$, therefore the last two inequalities imply $u_{n-1}^* = \eta N_{n-1}$. Applying this procedure recursively backward over tasks $n-2, \cdots, p$, we obtain $u_i^* = \eta N_i$ for all $i = p, \cdots, n$, when $\{p, \cdots, n\} \subseteq \bar{\mathcal{M}} \cap \{k, \cdots, n\}$. ■

This result asserts that when a BP ends with a sequence of optional set tasks, then it is optimal to process them all using the minimum feasible processing rate. Using Lemmas 2 and 3 we can now derive the condition for identifying a BP on the optimal state trajectory containing only optional set tasks.

Theorem 3 *Let $\{k, \cdots, n\} \subseteq \bar{\mathcal{M}}$. Tasks $\{k, \cdots, n\}$ constitute a BP in the optimal state trajectory if and only if*

$$x_{k-1}^* < a_k \tag{11}$$

$$a_k + \sum_{j=k}^{i} \eta N_i \geq a_{i+1}, \quad i = k+1, \cdots, n-1 \tag{12}$$

$$a_k + \sum_{j=k}^{n} \eta N_i < a_{n+1} \tag{13}$$

Proof. (If part) Task k starts a BP by definition based on (11). Next, we show that there is no idle period within tasks $\{k, \cdots, n\}$. Recall that, from (4c), the optimal solution must satisfy $u_i^* \leq \eta N_i$ for $i = k, \cdots, n$. First, suppose $u_i^* = \eta N_i$ for $i = k, \cdots, n$. If an idle period first occurs after the service completion of task s, $k \leq s < n$, then $a_k + \sum_{j=k}^{s} \eta N_j < a_{s+1}$, which contradicts (12). On the other hand, suppose $u_i^* < \eta N_i$ for some i, $k \leq i < n$ and an idle period occurs after the service completion of task s, $i \leq s < n$, Since $i \subseteq \bar{\mathcal{M}}$ and the cost function $\theta_i(u_i)$ is strictly decreasing in u_i, we can increase u_i^* and decrease $\theta_i(u_i)$, which contradicts the optimality of u_i^*. Therefore, u_i^* is large enough to either eliminate the idle period and still satisfy $u_i^* < \eta N_i$ or its value is $u_i^* = \eta N_i$. In the latter case, we have already seen that the presence of an idle period contradicts (12). Therefore, no idle period is possible if (12) holds.

Finally, we show that task n ends a BP. Since we have argued that no idle period occurs prior to task n, we have $x_n^* = a_k + \sum_{j=k}^{n} u_j^*$. In addition, the optimal solution must satisfy $u_i^* \leq \eta N_i$ for $i = k, \cdots, n$ and (13) implies that

$$x_n^* = a_k + \sum_{j=k}^{n} u_j^* \leq a_k + \sum_{j=k}^{n} \eta N_j < a_{n+1}$$

that is, task n ends a BP.

(Only if part) Since tasks $\{k, \cdots, n\}$ constitute a single BP in the optimal state trajectory, by Lemma 3 we have $u_i^* = \eta N_i$ for $i = k, \cdots, n$, that is,

$$x_i^* = a_k + \sum_{j=k}^{i} \eta N_j \text{ for } i = k, \cdots, n$$

Then, (11), (12) and (13) follow from the BP definition. ■

As in Theorem 2, note that condition (11) depends on x_{k-1}^*, but, as we will see, the validity of $x_{k-1}^* < a_k$ can be determined without requiring the explicit solution of P.

The next lemma identifies a condition under which the optimal departure time of a mandatory set task contained within a BP can be easily determined.

Lemma 4 *Let tasks m and $m+1$ belong to a BP in the optimal state trajectory and $m \in \mathcal{M}$. If $N_m/u_m^* > N_{m+1}/u_{m+1}^*$, then $x_m^* = d_m$.*

Proof. Proceeding by contradiction, assume that $x_m^* < d_m$ since $x_m^* > d_m$ is not feasible for $m \in \mathcal{M}$. Since it is assumed that $N_m/u_m^* > N_{m+1}/u_{m+1}^*$, there exists some $\delta > 0$ such that

$N_m/(u_m^* + \delta) \geq N_{m+1}/(u_{m+1}^* - \delta)$ and $x_m^* + \delta \leq d_m$. Observing that

$$(u_m^* + \delta) + (u_{m+1}^* - \delta) = u_m^* + u_{m+1}^*$$

and that

$$N_m/u_m^* > N_m/(u_m^* + \delta) \geq N_{m+1}/(u_{m+1}^* - \delta) > N_{m+1}/u_{m+1}^*$$

we argue exactly as in the proof of Lemma 2 to get:

$$\theta_m(u_m^* + \delta) + \theta_{m+1}(u_{m+1}^* - \delta) < \theta_m(u_m^*) + \theta_{m+1}(u_{m+1}^*)$$

Note that, with all other constraints remaining the same, $\{u_1^*, \cdots, u_m^* + \delta, u_{m+1}^* - \delta \cdots, u_K^*\}$ is a feasible solution of problem P. Therefore,

$$\begin{aligned}&\sum_{i=1}^{m-1} \theta_i(u_i^*) + \sum_{i=m+2}^{K} \theta_i(u_i^*) + \theta_m(u_m^* + \delta) + \theta_{m+1}(u_{m+1}^* - \delta)\\ &< \sum_{i=1}^{m-1} \theta_i(u_i^*) + \sum_{i=m+2}^{K} \theta_i(u_i^*) + \theta_m(u_m^*) + \theta_{m+1}(u_{m+1}^*)\end{aligned}$$

which contradicts the assumption that $\{u_1^*, \cdots, u_K^*\}$ is the optimal solution and the proof is complete. ■

This result helps us establish Lemma 5 which applies to a sequence of tasks in a BP that starts with a mandatory set task followed by one or more optional set tasks which end this BP. In this case, it becomes easy to determine the optimal controls of these tasks.

Lemma 5 *Let tasks $\{k, \cdots, n\}$ constitute a BP in the optimal state trajectory. If $m \in \mathcal{M} \cap \{k, \cdots, n\}$ and $\{m+1, \cdots, n\} \subseteq \bar{\mathcal{M}} \cap \{k, \cdots, n\}$, then, $x_m^* = d_m$ and $u_i^* = \eta N_i$, $i = m+1, \cdots, n$.*

Proof. We have shown in Lemma 3 that $u_i^* = \eta N_i$ for $i = m+1, \cdots, n$ when $\{m+1, \cdots, n\} \subseteq \bar{\mathcal{M}} \cap \{k, \cdots, n\}$ holds. Since u_m^* must satisfy constraint (4c), we have $u_m^* \leq \eta N_m$.

It is then obvious that adjacent tasks m and $m+1$ satisfy

$$N_m/u_m^* \geq N_{m+1}/u_{m+1}^* = 1/\eta$$

If $N_m/u_m^* > N_{m+1}/u_{m+1}^*$ holds, then it follows from Lemma 4 that $x_m^* = d_m$. If, on the other hand, $N_m/u_m^* = N_{m+1}/u_{m+1}^*$, we have $u_m^* = \eta N_m$. Since, $x_m^* = \max\{a_m, x_{m-1}^*\} + u_m^*$, we get

$$x_m^* = \max\{a_m, x_{m-1}^*\} + u_m^* \geq a_m + \eta N_m \geq d_m$$

where the last inequality is due to Assumption 1. At the same time, constraint (4d) requires that $x_m^* \leq d_m$, which, combined with the previous inequality, implies $x_m^* = d_m$. ■

We can now show how to determine if an optional set task ends a BP of type (iv) without explicitly solving problem P.

Theorem 4 *Let contiguous tasks $\{p, \cdots, q\}$ satisfy $p \in \mathcal{M}$, $d_p \geq a_{p+1}$, and $\{p+1, \cdots, q-1\} \subseteq \bar{\mathcal{M}}$. If there exists some $q \in \bar{\mathcal{M}}$ such that*

$$d_p + \sum_{j=p+1}^{i} \eta N_j \geq a_{i+1}, \quad i = p+1, \cdots, q-1 \tag{14}$$

$$d_p + \sum_{j=p+1}^{q} \eta N_j < a_{q+1} \tag{15}$$

then tasks $\{p, \cdots, q\}$ belong to the same BP and task q ends this BP in the optimal state trajectory. If such q does not exist, then, tasks $\{p, \cdots, q\}$ still belong to the same BP in the optimal state trajectory.

Proof. Since $d_p \geq a_{p+1}$, based on part 2 of Lemma 1 we have $a_{p+1} \leq x_p^*$, i.e., task p cannot end a BP in the optimal state trajectory. Next, we use a contradiction argument and assume that there exists an idle period after task s such that $p+1 \leq s \leq q-1$. From Lemma 5, the optimal solution satisfies $x_p^* = d_p$ and $u_i^* = \eta N_i$, $i = p+1, \cdots, s$. Then, $x_s^* = d_p + \sum_{j=p+1}^{s} \eta N_j$.

Since task s ends a BP, by definition $x_s^* < a_{s+1}$, which contradicts (14). Thus, we conclude that there is no idle period in between processing tasks $p, \cdots, q$ and they all belong to the same BP in the optimal state trajectory.

Note that we established the fact that $\{p, \cdots, q\}$ belong to the same BP when (14) holds regardless of whether $q \in \mathcal{M}$ or $q \in \bar{\mathcal{M}}$. Now we proceed by contradiction to show that if a task $q \in \bar{\mathcal{M}}$ exists satisfying (15) it must end this BP. Assume task $q+1$ also belongs to this BP. Then, by definition, we have $a_{q+1} \leq x_q^*$. Together with $x_p^* \leq d_p$ for $p \in \mathcal{M}$ and $u_j^* \leq \eta N_j$, $j = p+1, \cdots, q$, the optimal service completion time of task q satisfies

$$x_q^* = x_p^* + \sum_{j=p+1}^{q} u_i^* \leq d_p + \sum_{j=p+1}^{q} \eta N_j$$

Therefore,

$$a_{q+1} \leq d_p + \sum_{j=p+1}^{q} \eta N_j$$

which contradicts (15). It follows that tasks $\{p, \cdots, q\}$ must belong to the same BP and this BP indeed ends with $q \in \bar{\mathcal{M}}$.

On the other hand, if such $q \in \bar{\mathcal{M}}$ does not exist, we must conclude that either $q = K$ or $q \in \mathcal{M}$ is true. In either case, since tasks $\{p, \cdots, q\}$ satisfy (14), they belong to the same BP.

■

It follows from Theorem 4 that if $q \in \bar{\mathcal{M}}$ satisfying (15) can be found, this identifies the end of a type (*iv*) BP. If such q does not exist, the next mandatory set task, $q \in \mathcal{M}$, belongs to the same BP as $\{p, \cdots, q-1\}$. To ascertain if such a task q also ends the BP or not, we can invoke Lemma 1. In short, Theorem 4 provides an iterative way to determine if a sequence of optional set tasks belongs to the same BP as mandatory set tasks arriving before and after them.

We may now uniquely identify the start and end of a BP through the following conditions C1-C3:

C1 (Lemma 1 and Theorem 2) If task $n \in \mathcal{M}$ satisfies $d_n < a_{n+1}$, then n ends a BP of type (i) or (iii) in the optimal state trajectory.

C2 (Theorem 3) If contiguous tasks $\{k, \cdots, n\} \subseteq \bar{\mathcal{M}}$ satisfy (11), (12) and (13), then task n ends a type (ii) BP in the optimal sample path.

C3 (Theorem 4) If task $n \in \bar{\mathcal{M}}$ satisfies (14) and (15), and $d_m \geq a_{m+1}$ where $m = \arg\max\{i : i < n, i \in \mathcal{M}\}$, then task n ends a BP of type (iv) in the optimal sample path.

In order to make use of Theorems 2 and 3, we also need to resolve the question that was raised earlier, i.e., checking if the inequality $x^*_{k-1} < a_k$ holds (which appears as a condition in both theorems) without knowing x^*_{k-1}. This is accomplished by proceeding forward in time, starting with the initial condition $x^*_0 = 0$. If task k initiates a BP and we have determined that $\{k, \cdots, n\}$, $n \geq k$, belong to the same BP, then there are four cases to consider:

Case 1: If $\{k, \cdots, n\} \subseteq \mathcal{M}$, we know from Lemma 1 that n ends a type (i) BP if $d_n < a_{n+1}$, in which case $x^*_n = d_n$. Thus, checking if $x^*_n < a_{n+1}$ is equivalent to checking if $d_n < a_{n+1}$.

Case 2: If $\{k, \cdots, n\} \subseteq \bar{\mathcal{M}}$, then, from Lemma 3, if n ends this type (ii) BP we have $x^*_n = a_k + \sum_{j=k}^{n} \eta N_n$. Thus, checking if $x^*_n < a_{n+1}$ is equivalent to checking if $a_k + \sum_{j=k}^{n} \eta N_n < a_{n+1}$.

Case 3: If some tasks in $\{k, \cdots, n-1\}$ belong to the optional and some to the mandatory set and $n \in \mathcal{M}$, then, again using Lemma 1 n ends a type (iii) BP if $d_n < a_{n+1}$. As in Case 1, checking if $x^*_n < a_{n+1}$ is equivalent to checking if $d_n < a_{n+1}$.

Case 4: If some tasks in $\{k, \cdots, n-1\}$ belong to the optional and some to the mandatory set and $n \in \bar{\mathcal{M}}$, then we can determine $m = \arg\max\{i : i < n, i \in \mathcal{M}\}$. By Theorem 4, if n ends this type (iv) BP we have $x^*_n = d_m + \sum_{j=m+1}^{n} \eta N_j$. Thus, checking if $x^*_n < a_{n+1}$ is equivalent to checking if $d_m + \sum_{j=m+1}^{n} \eta N_j < a_{n+1}$.

Table 1 summarizes the observations above in the form of a detailed procedure for identifying how the optimal state trajectory is partitioned into BPs.

Table 1: Busy Period Decomposition Algorithm

Step 1:	Initialization
	$k = 1, n = 1$
Step 2:	Loop while $n \leq K$.
Step 3.1	If $n \in \mathcal{M}$, then
Step 4.1.1	If C1 holds,
	tasks $\{k, \cdots, n\}$ constitute a BP. Go to Step 6
Step 4.1.2	Else, $n = n + 1$. Go to Step 2
Step 3.2	Else, $n \in \bar{\mathcal{M}}$,
Step 4.2.1	If there exist task $m = \arg\max\{i : k \leq i \leq n, i \in \mathcal{M}\}$
Step 5.1.1	If C3 holds for task $n \in \bar{\mathcal{M}}$,
	Tasks $\{k, \cdots, n\}$ constitute a BP. Go to Step 6
Step 5.1.2	Else, $n = n + 1$. Go to Step 2
Step 4.2.2	Else, no such m
Step 5.2.1	If C2 holds for task $n \in \bar{\mathcal{M}}$
	Tasks $\{k, \cdots, n\}$ constitute a BP. Go to Step 6
Step 5.2.2	Else, $n = n + 1$. Go to Step 2
Step 6	$k = n + 1$, $n = n + 1$. Go to Step 2.

To illustrate the optimal BP structure identification process, let us consider an example of 10 tasks, $\{1, \cdots, 10\}$. Let $\{1, 2, 4, 8, 10\} = \mathcal{M}$ and $\{3, 5, 6, 7, 9\} = \bar{\mathcal{M}}$. Assume the following conditions hold:

(a) $d_1 > a_2$; (b) $d_2 < a_3$;

(c) $a_3 + \eta N_3 > a_4$; (d) $d_4 < a_5$;

(e) $a_5 + \eta N_5 > a_6$; (f) $a_5 + \eta N_5 + \eta N_6 < a_7$;

(g) $a_7 + \eta N_7 > a_8$; (h) $d_8 > a_9$; (i) $d_8 + \eta N_9 < a_{10}$

In this case, we can identify the following optimal BP structure. First, $\{1, 2\} \subseteq \mathcal{M}$ is a BP of type (i) since (a) and (b) satisfy C1. Next, $\{3, 4\}$ is a BP of type (iii) since (c) and (d) satisfy C1, while $\{5, 6\} \subseteq \bar{\mathcal{M}}$ is a BP of type (ii) since (e) and (f) satisfy C2. Finally, $\{7, 8, 9\}$ is a BP of type (iv) since (g) (h) and (i) satisfy C3, and $\{10\}$ is trivially a BP of type (i).

In the next section we discuss how to obtain the complete solution of problem P without invoking any nonlinear programming solver.

4 Complete Solution of Problem P

Our starting point is the optimal BP structure as derived in the previous section. Thus, we can partition the set $\{1, \cdots, K\}$ into subsets each corresponding to a specific BP. Let $\{k, \cdots, n\}$ define some such BP which belongs to one of the four types we have defined. From Theorem 1, the solution of P is obtained by solving each problem $P(k, n)$ corresponding to a BP $\{k, \cdots, n\}$.

First, suppose $\{k, \cdots, n\}$ is a BP of type (i), i.e., it contains only mandatory set tasks. In this case, constraints (5c) in problem $P(k, n)$ reduce to $x_i \leq d_i$ for all $i = k, \cdots, n$ and the problem becomes

$$\min_{u_k, \cdots, u_n} \sum_{i=k}^{n} \theta_i(u_i) \tag{16}$$

$$\text{s.t.} \quad x_i = a_k + \sum_{j=k}^{i} u_j \geq a_{i+1}, \quad i = k, \cdots, n-1$$

$$\gamma N_i \leq u_i \leq \eta N_i, \quad i = k, \cdots, n$$

$$x_i \leq d_i, \quad i = k, \cdots, n \tag{17}$$

This is now identical to the problem extensively studied in (Mao et al., 2007) and explicitly solved through the scalable, computationally efficient Critical Task Decomposition Algorithm (CTDA) that requires no nonlinear programming solver. In fact, $P(k, n)$ can also be solved as a constrained convex programming problem through efficient software such as CVX (Grant et al., 2008) as well. It can also be cast as a Dynamic Programming (DP) problem. However, neither of these approaches takes advantage of the specific decomposition properties established in the CTDA (briefly discussed in Section 1) which make it scalable in the number of tasks; extensive numerical results on the computational complexity of the CTDA relative to alternative methods are provided in (Mao and Cassandras, 2007b) and (Mao and Cassandras, 2006).

Second, suppose $\{k, \cdots, n\}$ is a BP of type (*ii*), i.e., containing only optional set tasks. Since $\mathcal{M} \cap \{k, \cdots, n\} = \varnothing$, the constraints (5c) of problem $P(k, n)$ can be removed. From Lemma 3, we know the solution to this specific problem is simply

$$u_i^*(k, n) = \eta N_i, \quad i = k, \cdots, n$$

Next, let $\{k, \cdots, n\}$ be a BP with both mandatory and optional set tasks. Note that the duration of the BP is defined by a_k and x_n^*. If $n \in \mathcal{M}$, i.e, a mandatory set task ends this BP, we have $x_n^* = d_n$ based on Lemma 1; otherwise, $n \in \bar{\mathcal{M}}$ and $x_n^* = x_{n-1}^* + \eta N_n$ based on Lemma 3. Denote the index of the last mandatory set task in this BP by

$$L(k, n) = \arg\max\{i : k \leq i \leq n,\ i \in \mathcal{M}\}$$

We then have $x_n^* = d_{L(k,n)} + \sum_{j=L(k,n)+1}^{n} \eta N_n$ from Lemma 3. This BP thus ends at time D_n such that

$$D_n = \begin{cases} d_n & \text{if } L(k, n) = n \in \mathcal{M} \\ d_{L(k,n)} + \sum_{j=L(k,n)+1}^{n} \eta N_j & \text{if } L(k, n) < n \in \bar{\mathcal{M}} \end{cases}$$

Observe that all optional set tasks in this BP must complete service by time D_n. Therefore, D_n can be treated as the "deadline" assigned to these tasks. That is, we set $d_j = D_n$ for $k \leq j \leq n$ and $j \in \bar{\mathcal{M}}$. The resulting optimization problem is of the same form as $P(k, n)$ in (16) with the constraints (17) replaced by $x_i \leq d_i$, for $i \in \mathcal{M} \cap \{k, \cdots, n\}$ and $d_i \leq D_n$ for

$i \in \bar{\mathcal{M}} \cap \{k, \cdots, n\}$:

$$\begin{aligned}
&\min_{u_k,\cdots,u_n} \sum_{i=k}^{n} \theta_i(u_i) \\
\text{s.t.} \quad & x_i = a_k + \sum_{j=k}^{i} u_j \geq a_{i+1}, \quad i = k, \cdots, n-1 \\
& \gamma N_i \leq u_i \leq \eta N_i, \quad i = k, \cdots, n \\
& x_i \leq d_i, \quad i \in \mathcal{M} \cap \{k, \cdots, n\} \\
& x_i \leq D_m, \quad i \in \bar{\mathcal{M}} \cap \{k, \cdots, n\}
\end{aligned}$$

Thus, this optimization problem can again be efficiently solved by the aforementioned CTDA. In summary, by first determining the optimal BP structure and then appropriately using the CTDA (to explicitly determine the optimal controls) provides a complete solution of problem P without solving any nonlinear programming problem.

5 Numerical Examples

In order to illustrate our solution approach, we consider its application to a problem arising in Dynamic Voltage Scaling (DVS) (Mao et al., 2007) where the cost function represents the energy consumption of a CMOS microprocessor:

$$\begin{aligned}
&\min_{u_1,\cdots,u_K} \sum_{i=1}^{K} C_1 N_i \left(\frac{V_t u_i}{u_i - N_i C_2} \right)^2 \\
\text{s.t.} \quad & x_i = \max\{a_i, x_{i-1}\} + u_i, \quad i = 1, \cdots, K \\
& \frac{V_{\max} C_2}{V_{\max} - V_t} N_i \leq u_i \leq \frac{V_{nt} C_2}{V_{nt} - V_t} N_i, \quad i = 1, \cdots, K \\
& x_m \leq d_m, \ m \in \mathcal{M}
\end{aligned}$$

Here $V_{\max}$ and V_t are the maximum voltage and threshold (minimum possible) voltage of the microprocessor, respectively. In addition, V_{nt} is the cut-off threshold voltage when the processor is operating: When the operating voltage level is below this value, the processor is considered sleeping or in stand-by mode and cannot process any tasks. C_1, C_2 are device parameters, and N_i is the number of operations for task i. It is easy to verify that the cost function above satisfies Assumption 2. In what follows, we set $V_{\max} = 5$, $V_t = 1$, $C_1 = 1$, $C_2 = 0.1$,and $N_i = 10$ for all i.

For the purpose of simulation, all task arrival times and deadlines are generated according to given distributions. In order to specify the mandatory task set $\mathcal{M}$, we consider four different tagging policies:

- *Policy 1*: For every k *consecutive* tasks, there are m mandatory set tasks. The assignment follows the algorithm in (Ramanathan, 1999), i.e., task i belongs to $\mathcal{M}$ when it satisfies
$$i = \left\lfloor \left\lceil \frac{i \times m}{k} \right\rceil \times \frac{k}{m} \right\rfloor$$
- *Policy 2*: The first m out of k tasks are assigned to $\mathcal{M}$. The remaining $k - m$ tasks are placed in the optional task set. For the next k tasks, the same procedure repeats.
- *Policy 3*: The first $k - m$ out of k tasks are assigned to $\bar{\mathcal{M}}$. The remaining m tasks are placed in $\mathcal{M}$. For the next k tasks, the same procedure repeats.
- *Policy 4*: With probability $m/k \in [0, 1]$, a task is placed in $\mathcal{M}$. Otherwise, it is placed in $\bar{\mathcal{M}}$.

It is obvious that when $m = k$, all tasks belong to the mandatory task set $\mathcal{M}$ and the problem is the same as the one with hard deadline requirements studied in (Mao et al., 2007). On the other hand, when $m = 0$, all tasks are optional and all policies result in identical $\bar{\mathcal{M}}$.

In order to compare the performance of the system under the optimal control proposed in this paper to other common controllers, we also study the following two common policies for allocating processing times to tasks:

- *Minimum processing time* (uncontrolled system): For all tasks, whether in $\mathcal{M}$ or $\bar{\mathcal{M}}$, the processing rate is set to the maximum value and is, therefore, independent of the task arrival and deadline structures. The cost value only varies with the total number of tasks in the system.

- *Best effort control*: The maximum feasible processing rate is applied to the mandatory set tasks. For the optional set tasks, the controller maintains a lower processing rate such that a task departs at the arrival time of the next incoming task. This controller uses the structural information of tasks in the system; however, there is no optimization effort made to improve performance. One can expect an improvement over the previous (uncontrolled) case, but less than the optimal performance obtained through the approach described in this paper.

The cost value obtained under the minimum processing time policy is always $250,000$ since it is independent of the task set configuration and all task arrival times and deadlines. Thus, we omit the corresponding column in the tables. We first consider four different (m, k) pair: $(1, 4)$, $(1, 10)$, $(2, 7)$, and $(3, 4)$. The cost comparison results are shown in Table 2 and indicate an order of magnitude in cost improvement. This is significant as it may translate into an order of magnitude increase in the lifetime of a wireless device which "dies" when its energy supply is exhausted. Table 3 shows one particularly interesting case when $(m, k) = (1, 2)$ and approximately half of the incoming tasks are classified as mandatory. In this case, tagging policies 1 and 2 result in identical mandatory and optional sets, thus, giving the same cost.

When $m = 0$, the set $\mathcal{M}$ is empty and all tasks are optional. We know from Lemma 2 that it is optimal to process all tasks using the minimum rate. The improvement over the Best Effort policy is significant as illustrated in Table 4.

Recall that the $(1, 1)$-firm requirement is equivalent to a purely hard real-time system. That is, all policies give the same mandatory task set $\mathcal{M}$ and contain all K tasks. It is

Table 2: Cost comparison with $K = 1000$ for different (m, k) constraints

		Cost Value	
(m, k)	**Tagging Policy**	**Best Effort**	**Optimal Control**
$m = 1$	**1**	71,376.1	5,003.8
	2	71,376.1	5,003.8
$k = 4$	**3**	71,135.2	4,999.2
	4	72,097.3	4,393.0
$m = 1$	**1**	35,607.7	2,034.5
	2	35,607.7	2,034.5
$k = 10$	**3**	35,364.6	1,968.7
	4	37,280.1	2,061.4
$m = 2$	**1**	79,962.8	5,593.6
	2	79,953.2	4,567.7
$k = 7$	**3**	79,492.7	4,527.2
	4	79,966.7	4,871.5
$m = 3$	**1**	190,376.8	10,221.2
	2	190,619.9	10,213.7
$k = 4$	**3**	190,379.1	10,260.1
	4	191,814.3	9,810.9

Table 3: Cost comparison of different control with $K = 1000$, $(1, 2)$-firm constraint

		Cost Value	
(m, k)	**Tagging Policy**	**Best Effort**	**Optimal Control**
$m = 1$	**1**	130,996.8	8,742.7
	2	130,996.8	8,742.7
$k = 2$	**3**	130,758.4	8,682.5
	4	133,861.1	7,530.9

Table 4: Cost comparison with $K = 1000$, soft real-time requirement

		Cost Value	
(m, k)	**Tagging Policy**	**Best Effort**	**Optimal Control**
$m = 0$	**1**	11,755.2	11.1
	2	11,755.2	11.1
$k = 1$	**3**	11,755.2	11.1
	4	11,755.2	11.1

noteworthy that the performance improvement under the optimal controller in all such cases is again significant, i.e., an order of magnitude, as depicted in Table 5.

Table 5: Cost comparison of different control with $K = 1000$, $(1, 1)$-firm constraint

		Cost Value	
(m, k)	**Tagging Policy**	**Best Effort**	**Optimal Control**
$m = 1$	**1**	250,000	11,692.1
	2	250,000	11,692.1
$k = 1$	**3**	250,000	11,692.1
	4	250,000	11,692.1

The performance of weakly hard real-time systems is often also measured through the *miss ratio* metric, defined as the fraction of tasks that miss their deadline. Recall that in our system all tasks are processed under a first-come-first-served discipline and the deadlines of optional set tasks are simply ignored. Nonetheless, optional set tasks may still meet their original deadlines due to the presence of mandatory set tasks whose deadlines affect the processing time of optional set tasks arriving before them. The (m, k)-firm constraints in our optimization problem guarantee that the deadline requirements of mandatory set tasks cannot be violated. In our simulation, we take into consideration the original deadlines of the optional set tasks and Table 6 provides miss ratio data for optional set tasks when the system operates under optimal control. Note that we omit the columns of data for $(0, 1)$ and $(1, 1)$. In the former case, the set $\mathcal{M}$ is empty and the deadline miss ratio is 1 since there is no mandatory set task in the system. In the latter case, all tasks meet their deadlines and the miss ratio is 0.

6 Conclusion and Future Work

DES with weakly hard real-time constraints provide a natural extension to previous work done for hard real-time and soft real-time constraints. As in the case of hard real-time constraints, we show that the optimization problem we formulate can be decomposed into a set of simpler

Table 6: Miss Ratio of Optional Set Tasks, $K = 1000$

		Tagging Policy			
(m, k)	**Number of Tasks**	**1**	**2**	**3**	**4**
$(1, 2)$	optional set	499	499	500	487
	misses	55	55	47	78
$(1, 10)$	optional set	899	899	900	892
	misses	40	40	60	75
$(2, 7)$	optional set	713	713	713	713
	misses	92	100	120	108
$(3, 4)$	optional set	250	249	250	244
	misses	21	26	17	33

ones and we identify a number of structural properties of its solution. This leads to a low complexity scalable algorithm for obtaining the explicit values of all optimal task processing times which capitalizes on the Critical Task Decomposition Algorithm (CTDA) of (Mao et al., 2007) derived when all hard real-time constraints must be guaranteed. Our ongoing work aims at extending the analysis to a multi-stage DES, where the bottleneck server can better manage the resources by scheduling tasks efficiently and, at the same time, meet end-to-end real-time constraints. An additional challenge is the on-line control of systems where no a priori information on task arrivals or deadlines is available. Although the low complexity of our solution makes it realistic to re-solve problem P at every task completion event using the available task information at that time, it is also desirable to make use of statistical information regarding the task arrival process, as well as the deadline and size characteristics of tasks.

Finally, as mentioned in Section 2, the optimization problem P may not have a feasible solution, in which case an auxiliary problem arises: Minimizing the number of tasks that need to be eliminated in order to obtain a feasible problem. Solving this problem efficiently is the subject of ongoing research with some results applicable to the hard real-time constraint case provided in (Mao and Cassandras, 2007a).

References

Aydin, H., Melhem, R., Mossé, D., and Mejia-Alvarez, P. (May 2004). Power-aware scheduling for periodic real-time tasks. *IEEE Trans. on Computers*, 53(5):584 – 600.

Bernat, G. and Burns, A. (1997). Combining (m, n)-hard deadlines and dual priority scheduling. In *RTSS '97: Proceedings of the 18th IEEE Real-Time Systems Symposium (RTSS '97)*, page 46. IEEE Computer Society.

Bernat, G., Burns, A., and Llamosi, A. (2001). Weakly hard real-time systems. *IEEE Transactions on Computers*, 50(4):308–321.

Cassandras, C. G. and Zhuang, S. (2005). Optimal dynamic voltage scaling for wireless sensor nodes with real-time constraints. In *Proceedings of SPIE – Intelligent Systems in Design and Manufacturing VI*.

Gamal, A. E., Nair, C., Prabhakar, B., Uysal-Biyikoglu, E., and Zahedi, S. (2002). Energy-efficient scheduling of packet transmissions over wireless networks. In *Proceedings of IEEE INFOCOM*, volume 3, 23-27, pages 1773–1782, New York City, USA.

Grant, M., Boyd, S., and Ye, Y. (2008). Cvx: Matlab software for disciplined convex programming (web page and software). Technical report, Stanford University.

Hua, S. and Qu, G. (2004). Energy-efficient dual-voltage soft real-time system with (m,k)-firm deadline guarantee. In *CASES '04: Proceedings of the 2004 international conference on Compilers, architecture, and synthesis for embedded systems*, pages 116–123. ACM Press.

Jeffay, K., Stanat, D. F., and Martel, C. U. (1991). On non-preemptive scheduling of periodic and sporadic tasks. In *Proc. of the IEEE Real-Time Systems Symposium*, pages 129–139.

Jonsson, J., Lonn, H., and Shin, K. G. (1999). Non-preemptive scheduling of real-time threads on multi-level-context architectures. In *Proceedings of the IEEE Workshop on Parallel and Distributed Real-Time Systems*, volume 1586, pages 363–374. Springer Verlag.

Liu, J. (2000). *Real - Time System*. Prentice Hall Inc.

Mao, J. and Cassandras, C. G. (2007b). Optimal control of two-stage discrete event systems with real-time constraints. *Journal of Discrete Event Dynamic Systems*. To Appear.

Mao, J. and Cassandras, C. G. (Dec. 2006). Optimal control of multi-stage discrete event systems with real-time constraints. In *Proc. of 45rd IEEE Conf. Decision and Control*, pages 1057–1062.

Mao, J. and Cassandras, C. G. (Dec. 2007a). Optimal admission control of discrete event systems with real-time constraints. In *Proc. of 46rd IEEE Conf. Decision and Control*. To appear.

Mao, J., Cassandras, C. G., and Zhao, Q. C. (June 2007). Optimal dynamic voltage scaling in power-limited systems with real-time constraints. *IEEE Trans. on Mobile Computing*, 6(6):678–688.

Miao, L. and Cassandras, C. G. (2006). Optimal transmission scheduling for energy-efficient wireless networks. In *Proceedings of INFOCOM*.

Miao, L. and Cassandras, C. G. (Sep. 2005). Optimality of static control policies in some discrete event systems. *IEEE Transactions on Automatic Control*, 50(9):1427 – 1431.

Pepyne, D. and Cassandras, C. (2000). Optimal control of hybrid systems in manufacturing. In *Proceedings of the IEEE*, volume 88, pages 1108–1123.

Quan, G., Niu, L., and Davis, J. P. (January 10-12, 2004). Power aware scheduling for real-time systems with (m,k)-guarantee. In *Proceedings CNDS-04: Communication Net-*

works and Distributed Systems Modeling and Simulation, San Diego, CA. The Society for Modeling and Simulation International.

Ramanathan, P. (1997). Graceful degradation in real-time control applications using (m, k)-firm guarantee. In *Proceedings of the 27th International Symposium on Fault-Tolerant Computing (FTCS '97)*, page 132, Washington, DC, USA. IEEE Computer Society.

Ramanathan, P. (1999). Overload management in real-time control applications using (m, k)-firm guarantee. *IEEE Trans. Parallel Distrib. Syst.*, 10(6):549–559.

Ramanathan, P. and Hamdaoui, M. (1995). A dynamic priority assignment technique for streams with (m, k)-firm deadlines. *IEEE Trans. on Computers*, 44(12):1443–1451.

Yao, F., Demers, A., and Shenker, S. (1995). A scheduling model for reduced CPU energy. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS'95)*, pages 374–382. IEEE Computer Society.