Optimal Control of Hybrid Systems in Manufacturing

DAVID L. PEPYNE, MEMBER, IEEE, AND CHRISTOS G. CASSANDRAS, FELLOW, IEEE

Invited Paper

Hybrid systems combine time-driven and event-driven dynamics. This is a natural framework for manufacturing processes: The physical characteristics of production parts undergo changes at various operations described by time-driven models, while the timing control of operations is described by event-driven models. Accordingly, in the framework we propose, manufactured parts are characterized by physical states (e.g., temperature, geometry) subject to time-driven dynamics and by temporal states (e.g., operation start and stop times) subject to event-driven dynamics. We first provide a tutorial introduction to this hybrid system framework and associated optimal control problems through a single-stage manufacturing process model. We then show how the structure of the problem can be exploited to decompose what is a hard nonsmooth, nonconvex optimization problem into a collection of simpler problems. Next, we present extensions to multistage manufacturing processes for which we develop solution algorithms that make use of Bezier approximation techniques. Emphasis is given to the issue of deriving solutions through efficient algorithms, and some explicit numerical results are included.

Keywords*—*Hybrid system, nonsmooth optimization, optimal control.

I. INTRODUCTION

Hybrid systems combine continuous with discrete dynamic behavior. Several different frameworks have been proposed for describing how the continuous and discrete aspects of such systems interact [1]–[5]. Some hybrid system formalisms extend event-driven models to allow time-driven activities to take place between event occurrences, or to determine event occurrence times. Other formalisms extend time-driven models to allow events to be injected, causing jumps in the state or switching in the dynamics. Regardless of the framework adopted, it is becoming clear that the analysis and design of controllers for hybrid systems will require tools that are more than just simple extensions of those for continuous or discrete systems.

In this paper, we describe a specific hybrid system modeling framework for which optimal control problems are formulated and solved by combining time-driven and event-driven methodologies. This framework is motivated by problems encountered in manufacturing settings. A hybrid system description is a natural one for manufacturing processes that perform a sequence of operations on a set of production parts, with the purpose of each operation to change some physical characteristics of the part. In the context of hybrid systems, the changes in the physical characteristics are described by *time-driven* (continuous) dynamics, while the timing control of the operations is described by *event-driven* (discrete-event) dynamics. Accordingly, in the framework we propose, manufactured parts are characterized by *physical* states (temperature, geometry, bacteria level, or some indirect measure of quality) subject to time-driven dynamics and by *temporal* states (operation start and stop times) subject to event-driven dynamics. A common problem in manufacturing is to design a control strategy to trade off job completion deadlines (satisfaction of customer demand) against the quality of the completed jobs. These two objectives are normally conflicting; when a product is in high demand, a natural tendency is to try to produce as many as possible as quickly as possible. Under such circumstances, quality often suffers. The hybrid system modeling framework described in this paper is designed specifically to address this optimal control problem.

Our hybrid system framework can be viewed as an extension of queueing network models, with the physical state of the jobs undergoing service determining the occurrence

Manuscript received October 29, 1999; revised March 18, 2000. The work of D. L. Pepyne was supported by the U.S. Army under Contracts DAAL03-92-G-0115 and DAAH04-0148, by the U.S. Air Force under Grant F49620-98-1-0387, by the Office of Naval Research under Contract N00014-98-10720, and by EPRI/DoD under Contract WO8333-03. The work of C. G. Cassandras was supported by the National Science Foundation under Grants EEC-9527422 and ACI-9873339, by AFOSR under Grant F49620-98-1-0387, by the Air Force Research Laboratory under Contract F30602-99-C-0057, and by EPRI/DoD under Contract WO8333-03.

D. L. Pepyne is with the Division of Engineering and Applied Science, Harvard University, Cambridge, MA 02138 USA (e-mail: pepyne@hrl.harvard.edu).

C. G. Cassandras is with the Department of Manufacturing Engineering, Boston University, Boston, MA 02215 USA (e-mail: cgc@bu.edu).

Publisher Item Identifier S 0018-9219(00)06467-7.

times of certain departure and arrival events. Alternatively, our framework may be viewed as an extension of the usual time-driven models, and can be described as a switched system consisting of an indexed set of (generally nonlinear) differential equations representing the way the physical state of each job evolves during processing. In this alternative viewpoint, event-driven dynamics determine the processing start and stop times, as well as the order in which the jobs are processed and the sequence of workcenters visited by each job. Accordingly, we adopt a solution approach that combines ideas from queueing networks with ideas from nonlinear optimization.

The purpose of this paper is to give a tutorial presentation to the present state-of-the-art of our modeling and optimal control framework, including its formulation, analysis, and synthesis of optimal control policies. In addition to addressing real manufacturing problems, this presentation also illustrates many of the difficulties inherent in hybrid systems. In particular, the introduction of discrete-event phenomena into an otherwise continuous system takes it off the well-beaten path of differentiable calculus and onto the less well-traveled path of nonsmooth calculus (Lipschitz continuous functions in our case). Moreover, since occurrences of discrete events are often random, hybrid systems are often stochastic, even when the continuous dynamics are completely deterministic. In describing our framework, we will show how these difficulties arise and how tools from nonsmooth calculus and optimal control can be combined to solve some broad classes of optimal control problems relevant to manufacturing. Crucial to the solution approach is the exploitation of structural properties in the optimal state trajectories, which are due to the event-driven dynamics and the nature of the cost functions of interest. In particular, an optimal state trajectory can be decomposed into independent segments, termed "busy periods." Moreover, each busy period can be further decomposed into "blocks" defined by certain events termed "critical." This decomposition allows us to replace a single computationally complex problem by a set of smaller and simpler problems and becomes the key to the development of algorithms for obtaining explicit solutions.

This paper is organized as follows**.** We begin with a high-level description of the proposed hybrid system framework in Section II, followed by a detailed model and optimal control problem formulation for single-stage processes in Section III. In Section III-A, we consider two specific classes of problems and explain the main difficulties involved, derive necessary conditions for optimality, introduce the concept of "critical jobs," and identify several structural properties that allow for the decomposition of the overall nonconvex and nonsmooth problem into smaller and simpler ones. In Section III-B, we present an explicit algorithmic procedure for obtaining the optimal control solution which takes advantage of the structural properties presented in Section III-A. In Section IV, we extend the framework to two-stage processes and describe a solution approach based on replacing the nondifferentable component of the hybrid system dynamics by a Bezier function approximate. In Section V, we outline a number of ongoing research directions stemming from the hybrid system framework presented in this paper.

II. The Hybrid System Framework

The original motivation for the hybrid system model developed in this paper came from our interaction with a metal manufacturing company seeking to integrate its metallurgical process control operations with its plantwide planning and scheduling operations. In this context, individual ingots undergo various operations to achieve certain metallurgical properties that define the "quality" of the finished metal. A common operation in metalmaking involves slowly heating ingots to some desired temperature and holding them at that temperature for some relatively long period of time. This is done to remove impurities, add beneficial elements to the metal, or achieve a desired grain structure (hardness). A "dual" to this operation is controlled cooling (annealing). In either case, the process control operation determines the rate at which the temperature is changed (more generally, a "heating profile"), as well as the amount of time that this temperature is held at each level. The objective of achieving high *quality* here is clearly in conflict with the plantwide planning and scheduling objective of *timely delivery* of finished metal products to clients. A similar situation arises in operations such as rolling, milling, or machining metals to achieve desired quality-defining shapes. We also mention in passing that semiconductor manufacturing processes exhibit several features similar to those of metal manufacturing mentioned here, with the added importance of timely product delivery due to the time-sensitive competitive nature of the semiconductor product market. The hybrid system framework we describe next is designed to deal with these tradeoffs between quality and satisfaction of customer demand. We remark, however, that our proposed framework is more generally applicable to any manufacturing process where there are a number of tasks that need to be completed by processing them through a network of workcenters. To reflect the generality of our framework, we will use the generic term "jobs" for the tasks, and "servers" to describe the devices that process the tasks. Whereas in metalmaking the jobs were ingots and the servers ovens, rolling mills, and lathes, more generally, the jobs might be food products or computer programs and the servers pasturizers and human computer programmers.

What distinguishes our model from a standard discrete-event queueing model (see, e.g., [6] and [7] for some background) and endows it with hybrid properties is that while a job is being processed by a server, its physical characteristics (functionality, shape, quality) are changing according to some time-driven dynamics (e.g., differential equations). In our model, the role of the server is to transform the physical characteristics of a job from some initial "raw" state to some final "completed" state. To represent the hybrid nature of the model, we characterize each job by a *physical* state and a *temporal* state (in general, these are both vectors). The physical state represents the physical characteristics of interest

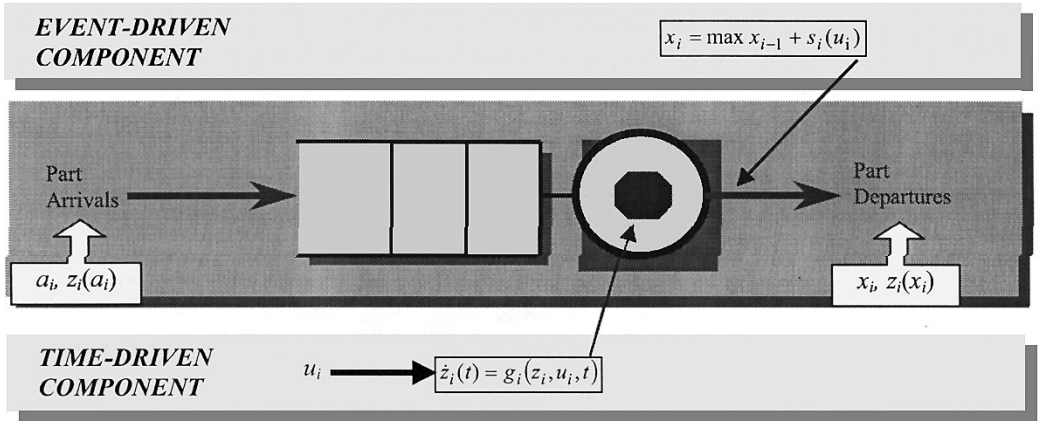


Fig. 1. A single-stage manufacturing process.

and evolves according to time-driven differential equations while the job is being processed by a server. The temporal state represents processing start and stop times and evolves according to discrete-event dynamics (queueing dynamics in our case).

Our objective is to formulate and solve optimal control problems that trade off costs on both the physical and temporal states of the completed jobs while taking into account associated control costs. We assume the sequence of jobs and their arrival times are assigned by an external source. In general, a control policy determining how these jobs are processed through the system involves the selection of 1) the waveforms that control the evolution of the physical state of each job, 2) the processing time for each job, 3) the order in which the jobs are to be processed, and, in the case of a network of servers, 4) the sequence of servers each job will visit. The control policy chosen must also satisfy various constraints on the allowable physical states, the allowable control inputs to the physical process, the order in which the jobs are processed, and the sequence of machines each job must visit. Taken separately, methods exist for solving each of the individual subproblems 1)–4) above: nonlinear optimal control for 1); discrete-event dynamic system performance optimization techniques for 2); and scheduling methods for 3) and 4). However, when these four subproblems are combined and tightly coupled, as they are in a hybrid system, the resulting problem is one we cannot generally solve using available methods.

In what follows, we describe our initial efforts toward a complete solution of the above four subproblems. Following a natural progression, our work so far has focused on manufacturing processes consisting of a single operation, with extensions to a series of operations performed in tandem. To preserve the tutorial nature of this paper and keep it from becoming too long, we do not provide complete technical proofs but refer the reader to appropriate references.

III. Single-Stage Processes

We begin with a simple manufacturing process consisting of a single operation. We represent this process as a single-server queueing system as shown in Fig. 1. As in standard queueing theory, the server is shown as a circle and the waiting area, or buffer, as an open rectangle.

A total of N jobs are assigned by an external source and arrive for processing at known times $0 \leq a_1 \leq a_2 \leq \cdots \leq a_N < \infty$. Let $i = 1, 2, \cdots, N$ be the index used to identify each job. The jobs are processed first-come, first-served (FCFS) by a work-conserving, nonpreemptive server, i.e., the server never idles when there are jobs in the queue and service is never interrupted. The processing time is $s_i(u_i)$, which is a function of a control variable $u_i(t)$. In general, the control is time-varying over the course of the processing time s_i. We limit ourselves here, however, to controls constrained to be constant over the duration of service, varying only with each new job, and chosen to ensure that processing times are nonnegative, i.e., $s_i(u_i) \geq 0$ (work on problems involving time-varying control may be found in [8]).

1) *Time-Driven Dynamics:* While a job is being processed by the server, its physical state z_i evolves according to a deterministic differential equation

$$\dot{z}_i(t) = g_i(z_i, u_i, t) \qquad z_i(\tau_i) = \zeta_i \tag{1}$$

where τ_i is the processing start time for job i and ζ_i is the state of the job at that time.

2) *Event-Driven Dynamics:* The completion time of each job is denoted by x_i and is given by the standard Lindley equation for a FCFS, work-conserving, nonpreemptive queue [7]

$$x_i = \tau_i + s_i(u_i) = \max(x_{i-1}, a_i) + s_i(u_i) \tag{2}$$

where we assume $a_0 = -\infty$ so that $x_1 = a_1 + s_1(u_1)$.

Note that the control u_i affects both time-driven dynamics (1) and event-driven dynamics (2), justifying the hybrid nature of the system. There are two alternative ways to view this hybrid system. The first is as a discrete event queueing system with time-driven dynamics evolving during processing in the server, as shown in Fig. 2. The other viewpoint interprets the model as a switched system. To illustrate, assume each job must be processed until it reaches

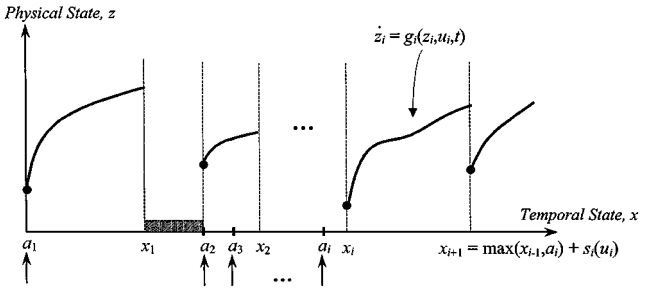


Fig. 2. Typical state trajectory.

a certain "quality" level denoted by Γ_i. That is, choose the processing time for each job such that

$$s_i(u_i) = \min\Big[t \geq 0 \colon z_i(\tau_i + t) = \int_{\tau_i}^{\tau_i + t} g_i(s, u_i, t)\, ds + \zeta_i \in \Gamma_i\Big]. \quad (3)$$

Fig. 2 shows the evolution of the physical state as a function of time (the "temporal state"). As illustrated in the figure, the dynamics of the physical state experience a "switch" when certain events occur. These events are of two types: *uncontrolled* (exogenous) arrival events, and *controlled* departure events. For the example in Fig. 2, the first event is an exogenous arrival event at time a_1. When this event occurs, the physical state begins evolving according to $\dot{z}(t) = g_1(z, u_1, t)$ until it reaches the desired target Γ_1. At this time, denoted by x_1, we remove job 1 from the server, which generates a controlled departure event. Note that the departure event at time x_1 occurs before the second job arrives at time a_2. Thus, there is a period of time, an idle period defined by the event times x_1 and a_2, during which the server has no jobs to process. At time a_2 processing on the second job commences, and the dynamics of the physical state switch to $\dot{z}(t) = g_2(z, u_2, t)$. The physical state evolves according to these dynamics until time x_2, when the target Γ_2 is reached and the job is removed from the server. Note, however, that the third job arrived before processing on the second job was completed. This job was forced to wait in the queue until time x_2, at which point the physical dynamics switch to $\dot{z}(t) = g_3(z, u_3, t)$, and work immediately begins on the third job. As indicated in Fig. 2, not only do the arrival and departure events cause switching in the physical dynamics according to (1) but the sequence in which these events occur is governed by their own dynamics given in (2).

For the above single-operation framework defined by (1) and (2), the optimal control objective is to choose a control policy $\pi = \{u_1, u_2, \cdots, u_N\}$ to minimize an objective function of the form

$$J = \sum_{i=1}^{N} \{\theta_i(u_i) + \psi_i(x_i)\}. \quad (4)$$

Note that although, in general, the state variables evolve continuously with time, (4) is a multistage optimization problem since we are concerned with the values of the state variables only at the job completion times $x_1, \cdots, x_N$. Note also that we do not include a cost on the physical state $z_i(x_i)$. Clearly, when the stopping criterion in (3) is used to obtain the service times, a cost on the physical state is unnecessary because in that case, we know that the physical state of each completed job satisfies our quality objectives, i.e., $z_i(x_i) \in \Gamma_i$. More generally, as we describe next, we can indirectly impose a cost on the physical state by appropriate choice of the functions $\theta_i(\cdot)$ in (4) and $s_i(\cdot)$ in (2). This is an "engineering" approximation that makes the problem somewhat more tractable, yet it still captures the essence of the original problem. In particular, we have studied two different classes of problems, each defined in terms of the form of the functions $\theta_i(\cdot)$, $\phi_i(\cdot)$, and $s_i(\cdot)$ in (4) and (2). For the first class, the control is interpreted as the processing time for the job, and the cost function trades off the quality of the completed jobs against the job completion times. We refer to these as Class 1 problems (see also [9]–[11]). For Class 2, the control is interpreted as the effort applied to the job, and the cost function trades off processing speed (which is related to the monetary cost of producing the job) against the job completion times (see [12]). Mathematically, these two classes of problems are defined as follows.

Class 1 Problems:

1) For every $i = 1, \cdots, N$, $\theta_i(\cdot)$: $\mathbb{R}^+ \longrightarrow \mathbb{R}^+$ is twice continuously differentiable, strictly convex, monotonically decreasing, and the following limits hold:

$$\lim_{u_i \to 0^+} \theta_i(u_i) = -\lim_{u_i \to 0^+} \frac{d\theta_i}{du_i} = \infty$$

and

$$\lim_{u_i \to \infty} \theta_i(u_i) = \lim_{u_i \to \infty} \frac{d\theta_i}{du_i} = 0.$$

2) For every $i = 1, \cdots, N$, $\psi_i(\cdot)$: $\mathbb{R} \longrightarrow \mathbb{R}$ is twice continuously differentiable, strictly convex, and its minimum is obtained at a finite point δ_i.
3) For every $i = 1, \cdots, N$, $s_i(\cdot)$: $\mathbb{R}^+ \longrightarrow \mathbb{R}^+$ is linear with $s_i(u_i) = \alpha u_i$ and $\alpha > 0$.

Class 2 Problems:

1) For every $i = 1, \cdots, N$, $\theta_i(\cdot)$: $\mathbb{R}^+ \longrightarrow \mathbb{R}$ is twice continuously differentiable, strictly convex, and monotonically increasing.
2) For every $i = 1, \cdots, N$, $\psi_i(\cdot)$: $\mathbb{R} \longrightarrow \mathbb{R}$ is twice continuously differentiable, strictly convex, and monotonically increasing.
3) For every $i = 1, \cdots, N$, $s_i(\cdot)$: $\mathbb{R}^+ \longrightarrow \mathbb{R}^+$ is twice continuously differentiable, strictly convex, monotonically decreasing, and the following limits hold:

$$\lim_{u_i \to 0^+} s_i(u_i) = -\lim_{u_i \to 0^+} \frac{ds_i}{du_i} = \infty$$

and

$$\lim_{u_i \to \infty} s_i(u_i) = \lim_{u_i \to \infty} \frac{ds_i}{du_i} = 0.$$

While the above definitions are somewhat technical, their interpretation is consistent with the previous discussion regarding time–quality tradeoffs in manufacturing systems. In

the case of Class 1 problems, we interpret the physical state z_i as a measure of the "quality" of the finished ith job. Beyond a certain minimum processing time, there are decreasing returns insofar as further improvement in quality is concerned. A common manufacturing problem is to produce jobs that meet certain minimum quality standards and deliver them by specified due dates. To achieve this, we place a cost on poor quality and a cost on missing the due date. As an example, let

$$\theta_i(u_i) = \frac{1}{u_i} \quad \psi_i(x_i) = (x_i - \delta_i)^2 \quad s_i(u_i) = u_i. \tag{5}$$

This set of functions satisfies the conditions of Class 1. Here, the control u_i is simply the service time. The cost on the control penalizes short service times, since this generally results in poor quality. Letting δ_i be the due date of job i, the cost on the departure time penalizes job earliness and tardiness.

Next, consider a Class 2 problem. In this case, our goal is to process each job so that it achieves a certain desired final state $z_i(x_i) = q$ from an initial state $z_i(\tau_i) = \zeta_i = 0$. Interpreting u_i as the control effort applied to job i, a simple form of the physical dynamics is

$$\dot{z}_i = u_i \tag{6}$$

where the effort determines the rate at which the quality z_i evolves and where, for simplicity, we take this effort to be constant over the entire processing time. Then, the amount of time we must process a job to reach the desired quality level q is

$$s_i(u_i) = \frac{q}{u_i} \tag{7}$$

where the requirement that $s_i(u_i) \geq 0$ implies that $u_i \geq 0$. In this setting, it is typical to consider a quadratic cost on the amount of effort involved, hence choosing

$$\theta_i(u_i) = u_i^2 \tag{8}$$

and if there is a due date δ_i for job i, we may choose a cost penalizing tardiness only as follows:

$$\psi_i(x_i) = \begin{cases} 0, & x_i < \delta_i \\ (x_i - \delta_i)^2, & x_i \geq \delta_i. \end{cases} \tag{9}$$

Observe that (7)–(9) satisfy the Class 2 conditions.

The analysis to follow is limited to Class 1 problems. As previously stated, our purpose is to give the intuition and insight behind our main results rather than provide all technical details. Readers interested in the details are referred to [9]–[11] and [13] regarding Class 1 problems, and to [12] regarding Class 2 problems.

A. Analysis of Class 1 Problems

In the following, we present some of the interesting properties of Class 1 problems. These properties will form the theoretical foundation of the numerical algorithm presented at the end of the section.

We begin by restating the Class 1 optimization problem, which we will hereafter refer to as Problem P1:

$$\min_{\pi} J = \sum_{i=1}^{N} \{\theta_i(u_i) + \psi_i(x_i)\}, \qquad \pi = \{u_1, u_2, \cdots, u_N\} \tag{10}$$

subject to

$$x_i = \max(x_{i-1}, a_i) + s_i(u_i) \tag{11}$$

and

$$s_i(u_i) = \alpha u_i \geq 0, \qquad \alpha > 0 \tag{12}$$

with a known arrival sequence $0 \leq a_1 \leq a_2 \leq \cdots \leq a_N < \infty$. The functions $\theta_i(\cdot)$, and $\psi_i(\cdot)$ are assumed to conform to the Class 1 conditions stated earlier. Observe that although this looks like a standard discrete-time optimal control problem, it is worth pointing out that the index i in (10) does not correspond to time steps; instead, it counts occurrences of the asynchronous job departure events involved in our hybrid system.

Although dynamic programming (DP) is a general-purpose methodology that can be used to solve this problem, and has been used for other types of hybrid system models, such as those studied in [14] and [15], the fundamental limitations arising from the notorious "curse of dimensionality" are well documented. For problems of any real sophistication, the policy space over which one needs to search for the optimal control policy is so large that it makes the computational burden unmanageable, even with today's fast computers. Furthermore, the DP algorithm involves storing historical search information and elements of the control policy, which makes the memory required for problems of even modest dimension prohibitive. Consequently, in practice, one tries to avoid numerically solving the DP equations.

For deterministic optimal control problems with real-valued states and controls, an alternative to DP is provided by variational (gradient-based) approaches (e.g., [16] and [17]). Variational techniques, however, are predicated on the differentiability of the objective function. Due to the nature of event-driven dynamics, however, hybrid systems typically do not yield objective functions with the requisite "smoothness." It is, in fact, the nondifferentiability introduced by the event generating mechanism in hybrid systems that makes associated optimal control problems so difficult to solve (see also [15]).

In the case of Problem P1, the root of the nondifferentiability lies with the max function in the event-driven dynamics (11). The function $\max(x_i, a_{i+1})$ is clearly not differentiable at the point where $x_i = a_{i+1}$; at all other points, however, it is differentiable with

$$\frac{d}{dx_i} \max(x_i, a_{i+1}) = \begin{cases} 0, & \text{if } x_i < a_{i+1} \\ 1, & \text{if } x_i > a_{i+1}. \end{cases}$$

To illustrate this difficulty and gain some insight as to the nature of a typical objective function in (10), consider a simple

Class 1 example with $N = 2$. Let $a_1 = 2$ and $a_2 = 3$, and define

$$\theta_1(u_1) = \frac{1}{u_1}, \quad \theta_2(u_2) = \frac{1}{u_2},$$
$$\psi_1(x_1) = x_1^2, \quad \psi_2(x_2) = (x_2 - 30)^2.$$

Making use of (11), this gives the cost function

$$\begin{aligned} J(u_1, u_2) &= \frac{1}{u_1} + \frac{1}{u_2} + x_1^2 + (x_2 - 30)^2 \\ &= \frac{1}{u_1} + \frac{1}{u_2} + (a_1 + u_1)^2 \\ &\quad + [\max(a_1 + u_1, a_2) + u_2 - 30]^2 \\ &= \frac{1}{u_1} + \frac{1}{u_2} + (2 + u_1)^2 \\ &\quad + [\max(2 + u_1, 3) + u_2 - 30]^2 \end{aligned} \tag{13}$$

which is plotted in Fig. 3. As seen in the figure, the objective function is generally not smooth: there is evidence of a "crease" in the surface running parallel to the u_2 axis at $u_1 = 1$. This corresponds to all points with $x_1 = a_2$. The surface is not differentiable across this crease, although it is differentiable everywhere else. Another interesting observation is that $J(u_1, u_2)$ is *not* a convex function of u_1, although it is convex in u_2. This is due to the last term in (13) involving the max function. Nonetheless, we can show [10] that Class 1 problems always have a unique global optimal solution (in this example, it turns out to be $u_1 = 0.452$ and $u_2 = 6.965$.) However, the fact that $J(\cdot)$ in (10) is generally not a convex function in the control sequence $\{u_1, \cdots, u_N\}$, despite the convexity assumptions on $\theta_i(\cdot)$, and $\psi_i(\cdot)$, adds to the difficulty of obtaining numerical solutions.

The points of nondifferentiability illustrated by this example form a critical component of the analysis to follow, and, for this reason, jobs associated with them are singled out and termed "critical."

Definition 1: A job $i = 1, \cdots, N-1$ is called *critical* if $x_i = a_{i+1}$.

Intuitively, critical jobs in the optimal policy correspond to the idea of processing job i as long as possible (to ensure high quality) and releasing it "just in time" for the next arriving job to start processing, i.e., precisely at time $x_i = a_{i+1}$. As we will see, it is indeed optimal to control the service times for some selected jobs in this way; determining which ones in particular is an important part of the problem solution.

Critical jobs are important for the following reason: If the optimal solution does not contain any critical jobs, then the objective function is differentiable at its global minimum. In this case, the problem can be solved using gradient-based methods, such as a standard two-point boundary-value problem (TPBVP) solver (e.g., see [17]). If the solution does contain critical jobs, however, a gradient-based method will not converge, but will "chatter," jumping back and forth across the crease at the minimum. We have developed an algorithm that detects this chattering and responds by taking action to force certain jobs to be critical (see [18]). Although this algorithm works well most of the time, it is heuristic and may fail to find the optimal solution.

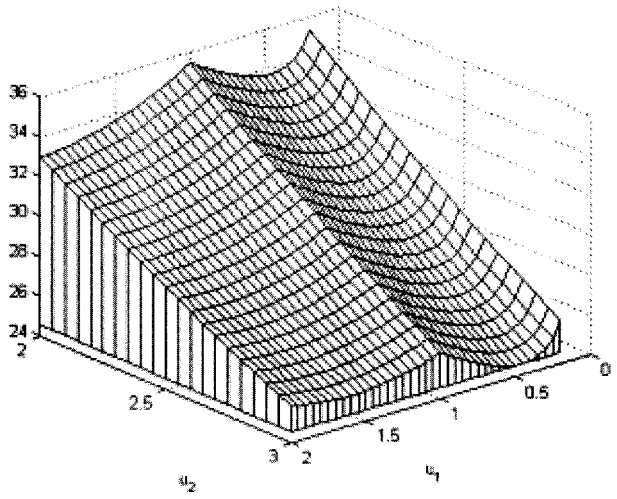


Fig. 3. Plot of a portion of $J(u_1, u_2)$ showing that it is nonconvex and nonsmooth.

1) Nonsmooth Optimization: To develop a rigorous solution approach that takes into account the nondifferentiability associated with critical jobs, we use some basic results from nonsmooth optimization theory (see [19] and [20]). Nonsmooth optimization theory deals with the optimization of Lipschitz continuous functions f: $E \to \mathbb{R}^n$, which satisfy $|f(x) - f(y)| \leq K|x - y|$, where K is a positive constant and $x, y \in E$, an open subset of $\mathbb{R}^n$. Note that Lipschitz functions are continuous, but need not be everywhere differentiable. According to Radmacher's theorem, they are, however, differentiable almost everywhere. The max function is a Lipschitz function, continuous and differentiable everywhere except at the point where its arguments are equal. Since the sum of Lipschitz functions is Lipschitz, and a composite function of Lipschitz functions is Lipschitz, our objective function in (10) is Lipschitz continuous. While objective functions in hybrid optimal control problems can have discontinuities (see [15]), they are more often Lipschitz, and this is due to the event-generating mechanism.

For continuously differentiable (smooth) functions, a necessary condition for a point to be a local extremum is that the gradient be zero there, and gradient-based methods can be used to search for points satisfying this condition. The necessary condition for general Lipschitz functions, which are often referred to as *nonsmooth* functions, cannot be phrased in terms of the gradient, since the gradient may not exist at the local extrema (as in our problem when the optimal solution contains critical jobs). Instead, the conditions for optimality are phrased in terms of a generalization of the gradient. Specifically, suppose f: $\mathbb{R}^n \to \mathbb{R}$ is a locally Lipschitz continuous function of $u \in \mathbb{R}^n$, and let $S(u)$ denote the set of all indexed sequences $\{u_m\}_{m=1}^{\infty} \subset \mathbb{R}^n$that satisfy the following three conditions: i) $u_m \to u$ as $m \to \infty$, ii) the gradient $\nabla f(u_m)$ exists for all $m = 1, 2, \cdots$, and iii) $\lim_{m\to\infty} \nabla f(u_m) = \varphi$ exists. Then, the *subdifferential* of f at u is denoted by $\partial f(u)$ and is defined as the convex hull of all limits φ corresponding to every sequence $u_m \in S(u)$. The subdifferential has the following three fundamental properties (see [19]): i) $\partial f(u)$ is a nonempty, com-

pact, and convex set in $\mathbb{R}^n$, ii) $\partial f(u)$ is a singleton *iff* f is continuously differentiable in some open set containing u, in which case $\partial f(u) = \nabla f(u_m)$, and iii) if u is a local extremum of f, then $0 \in \partial f(u)$. Each element φ of the subdifferential is referred to as a *generalized gradient*, or, in the convex case, a *subgradient*.

As a simple illustrative example, let us evaluate the subdifferential for the absolute value function, $f(x) = |x|$, at $x = 0$, where it is not differentiable. Since $x = 0$ is the global minimum, it must satisfy the necessary condition for optimality, i.e., $0 \in \partial f(0)$. In this case, the subdifferential is nothing but a closed interval on the real line, whose left and right endpoints are given respectively by the left derivative, $\lim_{x \uparrow 0^-}((df(x))/dx) = -1$, and the right derivative, $\lim_{x \downarrow 0^+}((df(x))/dx) = 1$. The subdifferential is, therefore, given by $\partial f(0) = [-1, 1]$, and clearly, $0 \in \partial f(0)$.

2) Subdifferential Derivation: The third property of the subdifferential presented in the previous section is the most important to us, since it provides a way to check if a candidate solution satisfies the necessary conditions for optimality. Solving our optimization problem (10)–(12), therefore, requires deriving an expression for the subdifferential $\partial J(u_1, \cdots, u_N)$. As we will see, the special structure of the event-driven dynamics (11) makes the derivation of the subdifferential rather straightforward and not much different than the simple example involving $f(x) = |x|$ in the previous section.

Referring back to Fig. 2, we see that, as the hybrid system operates, it generates a sequence of departure times in response to the given sequence of arrival times. These two sequences taken together define a *sample path*. It is convenient to divide a sample path into *busy* periods, during which the server is actively processing jobs, and *idle* periods, during which the system is empty waiting for jobs to arrive. Formally, we have the following.

Definition 2: An *idle period* is a time interval $(x_k, a_{k+1}]$ such that $x_k < a_{k+1}$ for any $k = 1, \cdots, N-1$.

Definition 3: A *busy period* is a time interval $(a_k, x_n]$ defined by a subsequence $\{k, k+1, \cdots, n\}$ such that i) $x_{k-1} < a_k$, ii) $x_i \geq a_{i+1}$ for all $i = k, \cdots, n-1$, and iii) $x_n < a_{n+1}$.

It is also useful to partition each busy period into *blocks*. A block starts with the first job after either a critical job or an idle period and it ends either with the next critical job or with the last job in the busy period. For any job $i = 1, \cdots, N$ let us define

$$n(i) = \min\{n \geq i \colon x_n < a_{n+1}\} \tag{14}$$

and

$$m(i) = \min\{m \geq i \colon x_m \leq a_{m+1}\}. \tag{15}$$

In words, $n(i)$ is the index of the last job in the *busy period* that contains job i, and $m(i)$ is the index of the last job in the *block* that contains job i. If the last job in the block that contains job i is *critical*, then $m(i) < n(i)$ and $x_{m(i)} = a_{m(i)+1}$. On the other hand, if the last job in the block containing job i is *not critical*, then $m(i) = n(i)$ and $x_{m(i)} < a_{m(i)+1}$.

These definitions are helpful in evaluating the subdifferential $\partial J(u_1, \cdots, u_N)$, which is given in terms of the left and right derivatives of $J(u_1, \cdots, u_N)$ with respect to $u_1, \cdots, u_N$. To see this, let us fix all controls on some sample path such that jobs $i, \cdots, m(i)$ are all in the same block and $x_{m(i)} < a_{m(i)+1}$. Since the service times are given by $s_i = \alpha u_i$, by adjusting the control u_i (keeping all other controls fixed), we change the departure times x_j of all jobs $j = i, \cdots, m(i)$ through

$$x_j = \max(x_{i-1}, a_i) + s_i(u_i) + \sum_{k=i+1}^{j} s_k(u_k) \tag{16}$$

where the max accounts for the fact that job i may be the first in its busy period. Moreover,

$$\frac{d\max(x_j, a_{j+1})}{dx_j} = \frac{dx_j}{dx_j} = 1 \qquad \text{for all } j = i, \cdots, m(i)-1$$

and

$$\lim_{x_{m(i)} \uparrow a_{m(i)+1}} \frac{d\max(x_{m(i)}, a_{m(i)+1})}{dx_{m(i)}} = \frac{da_{m(i)+1}}{dx_{m(i)}} = 0 \tag{17}$$

where the limit above is obtained by increasing u_i in (16) in such a way that x_j approaches a_{j+1} from the left for $j = m(i)$. In addition, since $x_{m(i)} < a_{m(i)+1}$, the state equation (11) implies that $((d\max(x_j, a_{j+1}))/dx_j) = 0$ for all $j > m(i)$. Therefore, recalling (16)

$$\begin{aligned}
&\lim_{x_{m(i)} \uparrow a_{m(i)+1}} \frac{\partial J}{\partial u_i} \\
&\quad = \lim_{x_{m(i)} \uparrow a_{m(i)+1}} \frac{\partial}{\partial u_i} \left[\sum_{j=1}^{N} [\theta_j(u_j) + \psi_j(x_j)] \right] \\
&\quad = \frac{d\theta_i}{du_i} + \sum_{j=i}^{m(i)} \frac{d\psi_j}{dx_j} \frac{dx_j}{du_i} \\
&\quad = \frac{d\theta_i}{du_i} + \frac{ds_i}{du_i} \sum_{j=i}^{m(i)} \frac{d\psi_j}{dx_j} \\
&\quad = \frac{d\theta_i}{du_i} + \alpha \sum_{j=i}^{m(i)} \frac{d\psi_j}{dx_j}.
\end{aligned} \tag{18}$$

Denoting this left derivative with respect to u_i by ξ_i^-, we set

$$\xi_i^- = \frac{d\theta_i}{du_i} + \alpha \sum_{j=i}^{m(i)} \frac{d\psi_j}{dx_j}. \tag{19}$$

A similar argument can be made for the right derivative by starting with u_i such that $x_{m(i)} > a_{m(i)+1}$, and noting that

$$\lim_{x_j \downarrow a_{j+1}} \frac{d\max(x_j, a_{j+1})}{dx_j} = \frac{dx_j}{dx_j} = 1 \tag{20}$$

for any critical job j between i and the end of the busy period that contains it, given by $n(i)$. Hence

$$\frac{d\max(x_j,\, a_{j+1})}{dx_j} = \frac{dx_j}{dx_j} = 1 \qquad \text{for all } j = i, \cdots, n(i){-}1$$

in which case we get

$$\lim_{x_{m(i)} \downarrow a_{m(i)+1}} \frac{\partial J}{\partial u_i} = \frac{d\theta_i}{du_i} + \alpha \sum_{j=i}^{n(i)} \frac{d\psi_j}{dx_j}. \tag{21}$$

Denoting this right derivative with respect to u_i by ξ_i^+, we set

$$\xi_i^+ = \frac{d\theta_i}{du_i} + \alpha \sum_{j=i}^{n(i)} \frac{d\psi_j}{dx_j}. \tag{22}$$

3) Necessary Conditions for Optimality: For Problem P1, it is easy to establish, using (19) and (22), the inequality $\xi_i^- \leq \xi_i^+$ for all $i = 1, \cdots, N$ (a proof may be found in [10]). Thus, all intervals $[\xi_i^-, \xi_i^+]$ are well defined. Then, the subdifferential of our objective function (10) is given by

$$\partial J = [\xi_1^-, \xi_1^+] \times \cdots \times [\xi_N^-, \xi_N^+] \subset \mathbb{R}^N. \tag{23}$$

A necessary condition for optimality then follows directly from the fact presented in Section III-A1 that for a locally Lipschitz continuous function f, if u is a local extremum of f, then $0 \in \partial f(u)$ [19].

Theorem 3.1: For Problem P1, an optimal control sequence u_i, $i = 1, \cdots, N$, must satisfy

$$0 \in [\xi_i^-, \xi_i^+] \subset \mathbb{R}^1 \tag{24}$$

for each $i = 1, \cdots, N$.

Using this necessary condition, we were able to show (see [10]) that the optimal solution for Problem P1 is unique.

Theorem 3.2: For Problem P1, the optimal control sequence u_i, $i = 1, \cdots, N$, is unique.

Establishing this uniqueness result is not trivial and it is particularly interesting because, as Fig. 3 clearly shows, the objective function is generally not convex and, therefore, may have local minima. Not only does nonconvexity make it difficult to establish uniqueness of the optimal solution but it also complicates the derivation of numerical algorithms. We remark that this difficulty does not arise in Class 2 problems, since it is easy to show using basic results from convexity theory [21] that these problems yield a strictly convex objective function, in which case there is only a single minimum.

4) Decoupling Properties: Due to the nature of the event-generating mechanism (queueing dynamics), our hybrid system has two useful decoupling properties that help to simplify its analysis and aid in the development of explicit algorithms. These properties are stated below as lemmas, proofs for which can be found in [10]. The first decoupling property is the *idle period decoupling* property.

Lemma 3.1: Consider a busy period consisting of jobs $\{k, \cdots, n(k)\}$ and let $i \in \{k, \cdots, n(k)\}$. The optimal control u_i^* depends only on the arrival times $a_k, \cdots, a_{n(k)}$ (it does not depend on the arrival times of jobs in other busy periods).

A related, but weaker, property applies to blocks and is referred to as *partial coupling*.

Lemma 3.2: Consider a block consisting of jobs $\{j, \cdots, m(j)\}$ and let $i \in \{j, \cdots, m(j)\}$. The optimal control u_i^* depends only on the arrival times a_j and $a_{m(j)+1}$ (it does not depend on any other arrival times).

These decoupling properties are central to the numerical algorithm presented in Section III-B. The idle period decoupling property tells us that the controls for individual busy periods can be determined independently of each other. This decomposes a large optimization problem consisting of N jobs into several smaller subproblems, one for each busy period. Further, partial coupling tells us that the controls for the jobs that come after a critical one are independent of the controls for the jobs that precede the critical job. This decomposes each busy period into subproblems, one for each block. Consequently, if we can identify the busy period structure (i.e., which jobs will be in which busy periods), and the block structure within each busy period (i.e., which jobs will be critical), then the problem can be solved as a collection of smaller independent subproblems. Therefore, of obvious importance is the identification of the busy period structure and the critical jobs within the busy periods.

5) Critical Jobs and Critical Intervals: As we mentioned already, obtaining solutions for our hybrid system model is complicated by the possibility that the optimal solution may contain critical jobs. At first sight, it might seem that the possibility of obtaining a solution with x_i precisely equal to a_{i+1} would be a very rare, even pathological, situation. However, this is not the case; in fact, almost any sample path will have some critical jobs within its busy periods. To see why this is and gain some more insight into critical jobs, let us explain how they occur.

Consider a busy period on an optimal sample path. Without loss of generality, because of Lemma 3.1, consider the first busy period on the sample path, starting at time a_1. Suppose this busy period contains B jobs and that none of these jobs are critical, i.e., $m(1) = n(1)$. Denote the optimal departure times for the jobs in this busy period by $x_{i,B}$, $i = 1, \cdots, B$. Here, i is the index of the job and B is the number of jobs in the busy period that contains job i. Assuming there exists an arrival sequence that gives such a busy period, we have the following two properties (detailed proofs given in [10]).

Lemma 3.3: The optimal departure times $x_{i,B}$, $i = 1, \cdots, B$, depend only on a_1 and B.

Lemma 3.4: The optimal departure times $x_{i,B}$, $i = 1, \cdots, B$, are monotonically decreasing in B, i.e., $x_{i,B} \leq x_{i,\overline{B}}$ for all $B \geq \overline{B}$.

The significance of Lemma 3.3 is that it allows us to precompute $x_{i,B}$ for any given a_1 and positive integer B. Lemma 3.4 is needed in order to explain the mechanism responsible for critical jobs. In particular, under this

lemma intervals of the form $[x_{i,B}, x_{i,i}]$ for any B and $i = 1, \cdots, B-1$ are well defined since $x_{i,B} \leq x_{i,i}$. Then, the following provides a criterion for checking the presence of critical jobs in a busy period (proof given in [10]).

Lemma 3.5: A busy period beginning with job $i = 1$ and containing B jobs on an optimal sample path includes at least one critical job if $a_{i+1} \in [x_{i,B}, x_{i,i}]$ for one or more jobs $i = 1, \cdots, B-1$.

We refer to the time intervals $[x_{i,B}, x_{i,i}]$ as *critical intervals*. Clearly, the wider the critical intervals, the greater the likelihood that the busy period will contain critical jobs. For typical Problem P1 examples we have considered, these intervals are, in fact, quite wide and result in the frequent occurrence of critical jobs.

While Lemma 3.5 can be used to determine *whether or not* a busy period will contain critical jobs, it cannot be used to determine *which* jobs in the busy period will be critical. To answer this question, one must actually explicitly solve the problem. One exception arises when the following sufficient condition is satisfied (again, a proof is given in [10]):

Lemma 3.6: Consider a busy period beginning with job $i = 1$, and suppose there are a total of N jobs remaining to be processed. If there exists some $L \leq N$ such that $a_{i+1} < x_{i,N}$ for all $i = 1, \cdots, L-1$ and $x_{L,L+1} \leq a_{L+1} \leq x_{L,L}$, then job L is critical.

Thus, Lemma 3.6 is sufficient to identify a critical job, whereas Lemma 3.5 implies that there are other situations where critical jobs can occur. To illustrate, consider the example shown in Fig. 4 for the case $N = 3$. In the figure, $x_{1,1}$, $x_{1,2}$, $x_{2,2}$, $x_{1,3}$, $x_{2,3}$, $x_{3,3}$ have been computed for a given arrival time a_1 and $B = 1$, 2, and 3. First, consider the implications of Lemma 3.6. With $N = 3$ and $L = 1$, according to the lemma, if $x_{1,2} \leq a_2 \leq x_{1,1}$, as shown in Fig. 4(a), then job 1 is critical (regardless of a_3). Therefore, the optimal departure time for job 1 is $x_1 = a_2$. Note that if $a_2 < x_{1,2}$ then job 2 is definitely in the same busy period as job 1, whereas if $a_2 > x_{1,1}$ then job 2 must start a separate busy period. Thus, the location of a_2 relative to the critical interval $[x_{1,2}, x_{1,1}]$ allows us to determine whether job 1 is critical, whether it ends the first busy period, or whether it is included in a busy period containing at least the first two jobs. Similarly, for $L = 2$, if $a_2 < x_{1,3}$ and $x_{2,3} \leq a_3 \leq x_{3,3}$, as shown in Fig. 4(b), then job 2 is critical.

Next, consider Lemma 3.5. Suppose that $a_2 < x_{1,2}$ and $a_3 < x_{2,3}$. Then, with $j = 2$ and $B = 3$, if $x_{1,3} \leq a_2 < x_{1,2}$ and $a_3 < x_{2,3}$, job 1 is the only job in the busy period satisfying the condition of this lemma, and hence must be critical. On the other hand, suppose $x_{1,3} \leq a_2 < x_{1,2}$ and $x_{2,3} \leq a_3 < x_{2,2}$, as shown in Fig. 4(c). In this case, both $j = 1, 2$ satisfy the conditions of the lemma; therefore, either or both of jobs 1 and 2 might be critical. Until we actually solve the problem, however, it is not possible to make a final determination.

When a case such as the one shown in Fig. 4(c) arises, we need a different criterion for identifying critical jobs. One such criterion is given by the following property:

$$\xi_i^- \cdot \xi_i^+ < 0. \tag{25}$$

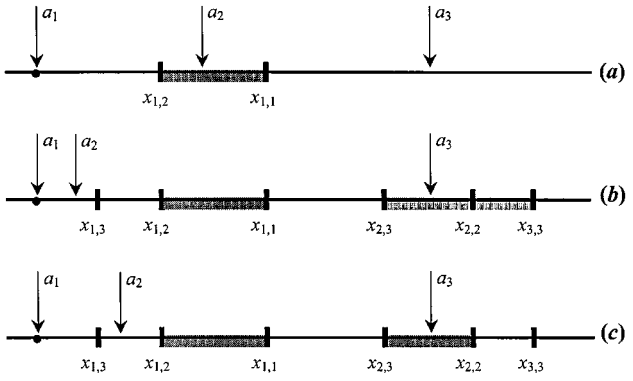


Fig. 4. Critical intervals for an example with $N = 3$.

The intuition behind this fact is not hard to see. From Theorem 3.1, we know that each critical job must satisfy $0 \in [\xi_i^-, \xi_i^+]$. Thus, it is expected that ξ_i^- and ξ_i^+ should have opposite signs, unless one of them were exactly zero. Because of uniqueness (Theorem 3.2), this can only happen in regions of measure zero, i.e., the objective function cannot have a flat region (i.e., a region with zero slope) either along the nondifferentiable "crease" (see Fig. 3) or to either side of it when the crease includes the global minimum. In fact, a stronger, necessary, and sufficient condition for critical jobs may be derived, which includes (25) as part of it; we omit it, since we do not make use of it in this paper, but the reader is referred to [10] for details. Equation (25) leads to a "sign test," which will be used in the algorithm to be described next to determine which jobs will be critical.

B. A Backward-Recursive Algorithm

As already mentioned, if the optimal solution to our hybrid optimal control problem could never contain critical jobs, then the objective function would be differentiable at its (unique) global minimum, and a standard gradient-based TPBVP solver, like those described in the classical texts (e.g., [17]), could be used to obtain numerical solutions. As just described, however, critical jobs not only can occur, but are, in fact, a usual feature of our optimal solutions. As a consequence, our hybrid system optimal control problem will typically have an objective function that is not differentiable at its minimum, in which case gradient-based methods will not work. Of course, other methods designed for solving nonsmooth optimization problems could be used. However, these algorithms are either computationally very complex or use first-order necessary conditions to guide their search, in which case they may get trapped in a local, instead of the global, minimum of this nonconvex optimization problem. In contrast, the algorithm we develop next is specifically tailored to our hybrid system optimal control problem, and, as such, it is efficient. Although not yet formally shown, we conjecture that it is guaranteed to return the unique global optimal solution (as opposed to a potentially local minimum).

The algorithm is based on the following fundamental observation: Even though the *overall* optimization problem is nonsmooth (recall Fig. 3), the controls within each *block*

can be obtained by solving a smooth nonlinear optimization problem with terminal constraints on the departure time of the last job in the block. For example, consider a busy period consisting of jobs $k, \cdots, n(k)$, and suppose that this busy period contains two blocks, i.e., $m(k) < n(k)$ and $m(m(k)+1) = n(k)$, as shown in Fig. 5. Then, by the idle period decoupling property (Lemma 3.1) and the partial coupling property (Lemma 3.2), we can determine the optimal controls for the jobs in this busy period by solving two independent TPBVPs: the first for jobs $k, \cdots, m(k)$, with terminal constraint $x_{m(k)} = a_{m(k)+1}$ (e.g., using a TPBVP solver and the penalty function approach; see [17]) and the second for jobs $m(k)+1, \cdots, n(k)$, with no terminal constraint on the departure time $x_{n(k)}$. That is, if we can identify the *busy period structure* of the optimal solution, and the *block structure* within each busy period, then we can decompose the solution of a computationally difficult nonsmooth optimization problem into a collection of simpler, smooth optimization problems (one for each block). What we need, therefore, is systematic way to identify the busy period and block structures of the optimal solution.

One possibility is to exhaustively search over all possible busy period and block structures for the optimal one. The main difficulty with such an approach is its computational complexity. It can be shown that there are 2^{N-1} different *busy period structures* [11]. By the same argument, a busy period containing B jobs has 2^{B-1} *block structures*. Such an approach is, therefore, infeasible, except for very small problems. Algorithm 1 given below provides a much more efficient way of determining the optimal busy period and block structures.

In describing the algorithm, we use the following definitions.

Problem $P_{j,k}(C)$:

$$\min_{u_j,\cdots,u_k} J = \sum_{i=j}^{k} \{\theta_i(u_i) + \psi_i(x_i)\} \tag{26}$$

subject to

$$x_i = \max(x_{i-1}, a_i) + s_i(u_i) \tag{27}$$

where, if $C = 1$, we include the terminal constraint

$$x_k = a_{k+1}$$

and if $C = 0$, there is no terminal constraint. As usual, the arrivals $a_j, \cdots, a_k$ and the terminal constraint a_{k+1} are assumed known.

We also define

$$\xi_{j,k} = \frac{d\theta_j}{du_j} + \frac{ds_j}{du_j} \sum_{i=j}^{k} \frac{d\psi_i}{dx_i}. \tag{28}$$

In this notation, $\xi_i^- = \xi_{i,m(i)}$ and $\xi_i^+ = \xi_{i,n(i)}$, where ξ_i^- and ξ_i^+ were defined in (19) and (22).

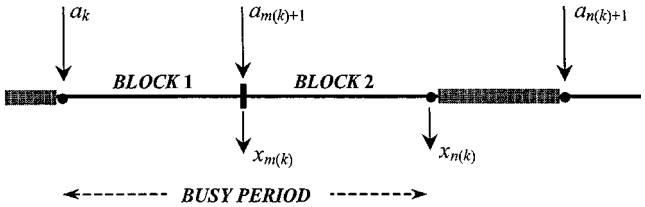


Fig. 5. A busy period consisting of two blocks.

The essential idea of the algorithm is to solve the overall nonsmooth optimization problem, which includes jobs $1, \cdots, N$ by decomposing it into a sequence of subproblems $P_{j,k}(C)$, each of which is simpler (since it involves fewer jobs, i.e., $j, \cdots, k$) and is also smooth. That is, we will only attempt to solve subproblems with $m(j) = m(k)$, in which case, efficient (conjugate) gradient-based methods can be used to solve $P_{j,k}(C)$. To do this, the algorithm proceeds in a backward-recursive manner, starting with job N and adding earlier jobs one at a time, until all jobs have been considered. As each new job i is added, simple sign tests on the right and left derivatives ξ_i^- and ξ_i^+ are performed to identify the busy period and block structures, and smooth optimal control problems are solved (with and without terminal constraints, as required) to obtain the optimal controls. The basic structure of the algorithm is as follows; for complete details, see [18].

Algorithm 1:

- INPUT:
  The arrival sequence, $a_1, \cdots, a_N$ and a terminal constraint flag TC. If $TC = 1$, the terminal constraint, a_{N+1}, is also needed.
- INITIALIZE:
  If $TC = 0$, solve $P_{N,N}(0)$ to get u_N^*, $x_N^* = a_N + s_N(u_N^*)$, and set $m(N) = N$ and $n(N) = N$;
  If $TC = 1$, set $x_N^* = a_{N+1} = a_N + s_N(u_N^*)$, invert the service function to get $u_N^* = s_N^{-1}(a_{N+1} - a_N)$, and set $m(N) = N$ and $n(N) = n(N+1)$.
- FOR ALL $i = N-1, N-2, \cdots, 1$:

Step 1) Set $x_i^* = a_{i+1} = a_i + s_i(u_i^*)$, and invert the service function to get, $u_i^* = s_i^{-1}(a_{i+1} - a_i)$.

Step 2) Compute $\xi_{i,i}$ and $\xi_{i,n(i+1)}$, and conduct a sign test to determine the nature of the coupling between jobs i and the busy period that begins with job $i+1$.

Step 2.1) If $\xi_{i,i} \cdot \xi_{i,n(i+1)} < 0$ (i.e., opposite sign), then job i is critical [recall equation (25)].
Set $m(i) = i$ and $n(i) = n(i+1)$.
Go to Step 1.

Step 2.2) If $\xi_{i,i} > 0$ and $\xi_{i,n(i+1)} > 0$, then job i is decoupled from $i+1$ and forms a separate busy period.

Solve $P_{i,i}(0)$ to get u_i^* and $x_i^* = a_i + s_i(u_i^*)$.

Set $m(i) = i$ and $n(i) = i$.

Go to Step 1.

Step 2.3) If $\xi_{i,i} < 0$ and $\xi_{i,n(i+1)} < 0$, job i is coupled to $i+1$, in which case a block-by-block forward sweep is used to merge job i into the busy period that contains job $i+1$ (details omitted).

Check to see if the busy period formed overlaps with the next (provided there is one to overlap with), i.e., check to see if $n(i) < N$ and $x_{n(i)} > a_{n(i)+1}$.

- If there is no overlap, go to Step 1).

- Otherwise (i.e., there is overlap), make a recursive call to Algorithm 1, with a terminal constraint $x_{n(i)} = a_{n(i)+1}$ to find the controls that make the last job in the busy period containing jobs $i, \cdots, n(i)$ critical.

Conduct a sign test to see if the controls computed above are optimal (details omitted). If so, go to Step 1); else, go to Step 2.3) to merge the two busy periods.

The easiest way to explain the algorithm, including the details omitted in Step 2.3), is through a simple example. Accordingly, let us consider the following $N = 5$ problem:

$$\min_{u_1,\cdots,u_5} J = \sum_{i=1}^{5} \{u_i^{-1} + x_i^2\}$$
$$\text{subject to } x_i = \max(a_i, x_{i-1}) + u_i^2 \qquad (29)$$

for the arrival sequence $\{0.4, 0.5, 0.7, 0.9, 1.3\}$. Note that, although this problem has the same interpretation as Class 1 (trades off quality against completion time), the service functions $s_i(u_i)$ are not linear. Hence, this problem is more general than those in Class 1. However, it is possible to show (using basic facts from convex analysis [21]) that this particular example has a strictly convex objective function, and hence satisfaction of the necessary conditions implies that the unique global solution has been found.

Fig. 6, which plots sample paths (the number of jobs in the system vs. time), shows the progress of Algorithm 1 (implemented through MATLAB) as it proceeds job by job toward the final solution.

Initialization ($i = 5$)*:* The algorithm is initialized by solving $P_{5,5}(0)$ to obtain u_5^* and $x_5^* = a_5 + (u_5^*)^2$. These arrival and departure events are plotted in Fig. 6(a).

$i = 4$: Next, job 4 is introduced, and the algorithm seeks to find the optimal controls u_4^* and u_5^* for these two jobs in isolation. To do so, the algorithm must determine whether or not job 4 should be decoupled from job 5, coupled to it, or be a critical job. To determine this, we assume job 4 is critical, and set $x_4^* = a_5 = a_4 + s_4(u_4^*)$, in which case, $u_4^* = s_4^{-1}(a_5 - a_4)$ [Step 1)]. Then, by computing the quantities $\xi_{4,4}$ and $\xi_{4,5}$ [Step 2)], a simple sign test tells us the nature of the coupling between the two jobs.

1) If $\xi_{4,4}$ and $\xi_{4,5}$ have opposite signs [Step 2.1)], then by (25), job 4 is critical.
2) If $\xi_{4,4} > 0$ and $\xi_{4,5} > 0$, then we know from definitions (19) and (22) that increasing the control increases the cost.
3) Conversely, if $\xi_{4,4} < 0$ and $\xi_{4,5} < 0$, then increasing the control decreases the cost.

Consequently, when the service function $s_i(\cdot)$ is monotonically increasing, as in this example, then both $\xi_{4,4}$ and $\xi_{4,5}$'s being positive implies that the service time is too long, and there should be an idle period between jobs 4 and 5 [Step 2.2)]. On the other hand, these quantities being negative says that the service time is too short, and job 4 should be merged with job 5 [Step 2.3)]. As illustrated in Fig. 6(b), $\xi_{4,4}$and $\xi_{4,5}$ are both positive, implying that jobs 4 and 5 are decoupled into separate busy periods. We, therefore, follow Step 2.2) and solve $P_{4,4}(0)$ to obtain u_4^* and x_4^*; the result is shown in Fig. 6(c). Because of the idle period decoupling property

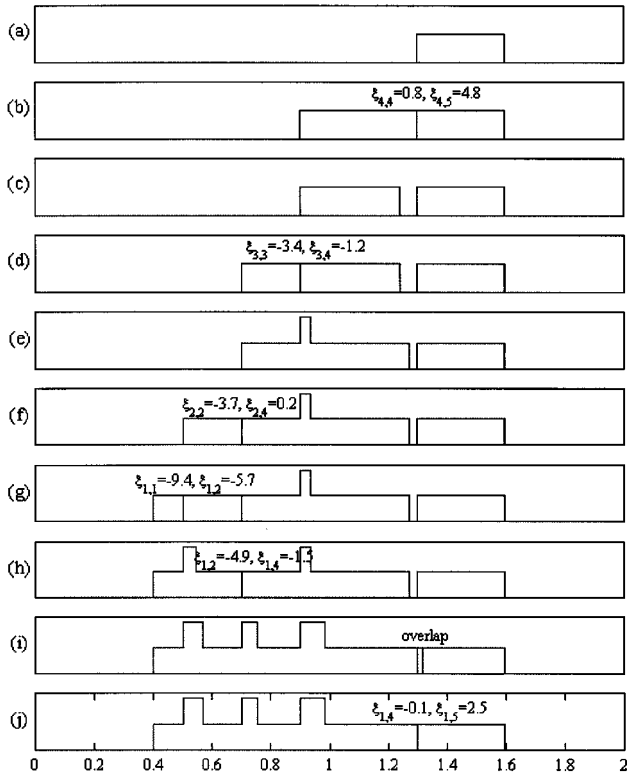


Fig. 6. Example showing the operation of the backward-recursive Algorithm 1.

(Lemma 3.1), the optimal control u_5^* for job 5 does not need to be reevaluated.

Here we make the following remark: if the service function is monotonically decreasing, as for Class 2 problems, Algorithm 1 can still be used if the sign tests in Steps 2.2) and 2.3) are switched, i.e., both positive implies decoupling jobs i and $i+1$, and both negative implies coupling jobs i and $i+1$.

$i=3$: The algorithm continues by introducing job 3, making it critical, computing $\xi_{3,3}$ and $\xi_{3,n(4)} = \xi_{3,4}$, and performing the sign test of Step 2). As shown in Fig. 6(d), $\xi_{3,3}$ and $\xi_{3,4}$ are both negative, indicating that jobs 3 and 4 should be coupled by merging job 3 into the same busy period as job 4. To do this, the algorithm executes Step 2.3) and solves $P_{3,4}(0)$ to get the result shown in Fig. 6(e). Since there is no overlap between x_4 and a_5, the algorithm is done with job 3. Again, because of the idle period decoupling property, the optimal control for job 5 does not change.

$i=2$: When job 2 is introduced, the sign test in Step 2.1) indicates it should be critical [see Fig. 6(f)]. Once more, because of the partial coupling property, the controls for jobs 3–5 are unaffected, and the algorithm is finished with job 2.

$i=1$: When job 1 is introduced, things get interesting. In this case, the sign test of Step 2) indicates that job 1 should be merged into the busy period containing jobs 2–4 [see Fig. 6(g)] and the algorithm executes Step 2.3). Note that the algorithm cannot merge job 1 by simply solving $P_{1,4}(0)$, because it may turn out that job 2 is still critical, in which case $P_{1,4}(0)$ is a nonsmooth optimization problem, which is precisely what we are trying to avoid. To deal with this possibility, merging job 1 into the busy period containing jobs 2–4 requires a block-by-block forward sweep through the busy period. The first part of the forward sweep involves solving $P_{1,2}(1)$, resulting in Fig. 6(h). A sign test with $\xi_{1,2}$ and $\xi_{1,4}$ then indicates that the block just formed by merging jobs 1 and 2 should be merged with the block containing jobs 3 and 4, i.e., we now know that job 2 will *not* be critical. We already know that job 3 is not critical, hence, the algorithm proceeds to solve $P_{1,4}(0)$, obtaining the result shown in Fig. 6(i), and the algorithm is finished with the first part of Step 2.3). The algorithm then proceeds to the second part of Step 2.3) to check if the busy period just formed overlaps with the busy period containing job 5. In this case, there is overlap indicating that the busy period consisting of jobs 1–4 needs to be merged with the busy period consisting of job 5. Merging these busy periods requires some care, because doing so could result in any one of the jobs $1, \cdots, 4$ becoming critical. To merge the two busy periods, this step begins by making a recursive call to Algorithm 1 to solve a fixed endpoint problem for jobs 1–4 with terminal constraint $x_4 = a_5$. Without going into the details, this results in Fig. 6(j). Conducting a sign test with $\xi_{1,4}$ and $\xi_{1,5}$ indicates that job 4 is critical. Consequently, the algorithm is finished with job 1, and the solution is complete.

As just described, Algorithm 1 solves the nonsmooth optimization problem by decomposing it into a collection of *simpler and smooth* optimization problems, essentially identifying the block structure, and solving a smooth optimization problem for each block. This capability, however, comes at the cost of a complex recursive procedure that can require recomputing the controls for each job several times before the final solution is obtained. It should be clear that the algorithm achieves its best performance when all jobs are critical, because in that case it only needs to solve $P_{N,N}(0)$ (in the initialization step), and after that all of the other controls are obtained by inverting the service function [Steps 2.2) and 2.3) are never invoked]. The next best situation is when every job forms a separate busy period. In this case, the solution becomes one of solving N problems $P_{i,i}(0)$, one for each job $i = 1, \cdots, N$. The worst case is when all jobs are in a single busy period, since this generally requires repeated merging of blocks. Even so, the algorithm is much more efficient than examining all possible block structures.

We remark that we have recently developed two other algorithms for the Class 1 problems. The first, described in [11], is another backward algorithm. It is similar to Algorithm 1, except that it searches busy period structures instead of block structures. We have shown that this algorithm does not need to search all 2^{N-1} possible busy period structures, but at most $2N-1$. The second algorithm, described in [13], gives an even better performance, requiring a search of only N busy period structures to solve the problem. Moreover, we can prove that both algorithms are guaranteed to obtain the unique optimal solution. Our ongoing research is investigating the advantages and disadvantages of the various algorithms we have developed.

IV. Two-Stage Processes

Next, we look at the simplest case of a manufacturing process involving multiple operations, i.e., a manufacturing process that requires a sequence of two operations to be performed on each job. A natural extension of the cost function (4) for the single-stage case gives the minimization problem

$$J = \sum_{i=1}^{N} \{\theta_{i,1}(u_{i,1}) + \theta_{i,2}(u_{i,2}) + \psi_{i,1}(x_{i,1}) + \psi_{i,2}(x_{i,2})\}$$

where the first subscripted quantity is the job index ($i = 1, \cdots, N$) and the second is the server index ($j = 1, 2$). Assuming both queues are FCFS, nonidling, and nonpreemptive, the event-driven dynamics are given by two coupled Lindley equations of the same form as (2)

$$x_{i,1} = \max(x_{i-1,1}, a_i) + s_{i,1}(u_{i,1})$$
$$x_{i,2} = \max(x_{i-1,2}, x_{i,1}) + s_{i,2}(u_{i,2}).$$

Note the coupling between the two stages: a departure from the first immediately becomes an arrival at the second. Again we need nonnegative service times, i.e., $s_{i,1}(u_{i,1}) \geq 0$ and $s_{i,2}(u_{i,2}) \geq 0$.

For this tandem operation case, therefore, we have the following optimal control problem, which we will call Problem P2:

$$\min_{\pi} J = \sum_{i=1}^{N} \{\theta_{i,1}(u_{i,1}) + \theta_{i,2}(u_{i,2}) + \psi_{i,1}(x_{i,1}) + \psi_{i,2}(x_{i,2})\},$$
$$\pi = \{u_{1,1}, \cdots, u_{N,1}, u_{1,2}, \cdots, u_{N,2}\} \tag{30}$$

subject to

$$x_{i,1} = \max(x_{i-1,1}, a_i) + s_{i,1}(u_{i,1}) \tag{31}$$
$$x_{i,2} = \max(x_{i-1,2}, x_{i,1}) + s_{i,2}(u_{i,2}) \tag{32}$$

and

$$s_{i,1}(u_{i,1}) \geq 0 \qquad s_{i,2}(u_{i,2}) \geq 0 \tag{33}$$

with a known arrival sequence $0 \leq a_1 \leq a_2 \leq \cdots \leq a_N < \infty$. For the examples in this section, we will assume that the functions $s_{i,j}(\cdot)$, $\theta_{i,j}(\cdot)$, and $\psi_{i,j}(\cdot)$ conform to the Class 2 conditions. Under these conditions, we can show that the objective function (although nonsmooth) is strictly convex, and hence has a unique solution.

For the single operation case, it was relatively easy to obtain the subdifferential as a collection of intervals as shown in (23). The situation is not as simple in the two-stage case. For one, now the control for each job is a vector, consisting of a control for the first operation and another for the second. For another, the coupling between the two operations complicates matters substantially. As a consequence, the resulting subdifferential is, in general, a region in $\mathbb{R}^{2N}$ (the dimension of the control vector). Determining this region, however, is very difficult, except for the most trivial situations. The difficulty in determining the subdifferential, therefore, strongly discourages the use of an algorithm involving generalized gradients (to date, we have been unable to develop any such scheme).

An alternative that provides systematic means for obtaining approximate solutions to our hybrid optimal control problem is based on the following observation. By our Class 2 (as well as Class 1) assumptions, the only function that is not everywhere differentiable is the max function appearing in the event-driven dynamics (31), (32). If we were to replace the max by a continuously differentiable surrogate that closely approximates it, then we would be left with a smooth programming problem, for which well-developed numerical algorithms exist (e.g., conjugate gradient methods).

Accordingly, suppose we replace the max function with a Bezier function as in Fig. 7 (see, e.g., [22]). A Bezier function is constructed using $n+1$ "control points" represented by vectors $\mathbf{v}_1, \cdots, \mathbf{v}_n$ and is parametrically given by

$$\mathbf{v}(t) = \sum_{i=0}^{n} \mathbf{v}_i B_{i,n}(t)$$

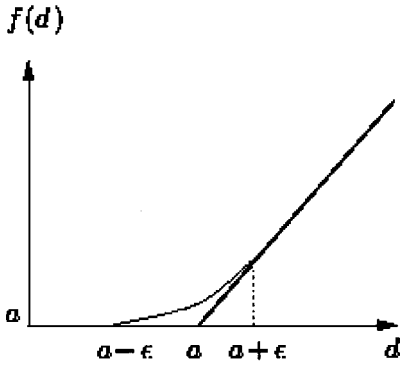


Fig. 7. Bezier approximation of a max function.

where

$$B_{i,n}(t) = \frac{n!}{i!(n-i)!} t^i (1-t)^{n-i}$$

and $0 < t < 1$. The control points define a "characteristic polygon," and the Bezier function has the property that it is contained within the convex hull of this characteristic polygon. In our case, there are three obvious control points to use: the point (a, a) shown in Fig. 7, where the max function is not differentiable, and two points $(a-\epsilon, a)$ and $(a+\epsilon, a+\epsilon)$, which define a neighborhood of a on the d-axis in Fig. 7 (where ϵ is shown as e). An additional property of $\mathbf{v}(t)$ is that the tangents at the first and last control points coincide with the first and last segments of the characteristic polygon. In our case, the two properties imply that the derivative of the Bezier approximation of the max function is always between zero and one. Making this replacement gives a continuously differentiable surrogate, which by our choice of ϵ can be made to approximate the max function to any arbitrary degree of accuracy. For the max function, the Bezier approximation over the range $d = a-\epsilon$ to $d = a+\epsilon$ is given by $f(d) = a + \epsilon t^2$, where $t = (1/2\epsilon)\, d - ((a-\epsilon)/2\epsilon)$. In other words, the max has been replaced by the surrogate

$$\max(x_i, a_{i+1}) = \begin{cases} a_{i+1}, & \text{if } x_i < a_{i+1} - \epsilon \\ a_{i+1} + \dfrac{1}{4\epsilon}(x_i - a_{i+1} + \epsilon)^2, & \text{if } a_{i+1} - \epsilon \leq x_i \leq a_{i+1} + \epsilon \\ x_i, & \text{if } x_i > a_{i+1} + \epsilon \end{cases}$$

with a derivative given by

$$\frac{d}{dx_i}\max(x_i, a_{i+1}) = \begin{cases} 0, & \text{if } x_i < a_{i+1} - \epsilon \\ \dfrac{1}{2\epsilon}(x_i - a_{i+1} + \epsilon), & \text{if } a_{i+1} - \epsilon \leq x_i \leq a_{i+1} + \epsilon \\ 1, & \text{if } x_i > a_{i+1} + \epsilon. \end{cases}$$

Using this approach with a standard gradient-based TPBVP solver, we can periodically reduce ϵ and recompute the Bezier function to make the fit increasingly tighter, to achieve any desired accuracy. The numerical results shown next illustrate the effectiveness of this simple approach.

As an illustration, we consider a problem with $N = 6$ jobs. These six jobs arrive at known times 0.2, 0.7, 1.3, 1.4, 1.8, and 2.5, respectively. Each job is processed in the first server until it reaches a physical state $q_1 = 0.5$, and in the second server until it reaches a physical state $q_2 = 0.6$. The cost function we consider is

$$J = \sum_{i=1}^{N} \{\alpha_1 u_{i,1}^2 + \alpha_2 u_{i,2}^2 + \beta x_{i,2}^2\} \tag{34}$$

with parameters $\alpha_1 = 2$, $\alpha_2 = 3$, and $\beta = 2$. Fig. 8 shows the optimal sample paths for each server. Note that the presence of critical jobs at stage 2 is captured by the Bezier approximation. It is also worthwhile observing that the optimal sample path at server 2 consists of either single-job busy periods or busy periods containing only critical jobs. In other words, the optimal control is to always keep the queue between the two stages empty. This is not a coincidence but a reflection of a property of the optimal solutions for this class of problems. We are currently investigating this property in order to assess its range of applicability (preliminary results are included in [23]).

Of course, the Bezier approach can also be used to solve single-stage problems. Its simplicity would seem to give it a clear advantage over Algorithm 1 in Section III-B. This, however, may not be the case. Recall that Class 1 problems are not convex, meaning that the Bezier approximation technique, which uses gradients to guide the search, may get trapped in a local minimum. Moreover, the parameter ϵ must be reduced to smaller and smaller values for the Bezier approach to get the exact solution, making convergence sometimes slow. Algorithm 1, on the other hand, may require the solution of many subproblems, but each of these subproblems may be simpler (because it usually involves only few jobs) and can be solved very quickly. An advantage of the Bezier approximation may be its generality, in that it is applicable to any network configuration whose dynamics can be described in the "max-plus" algebra; in fact, it has been used successfully for N-stage processes, $N > 2$. As mentioned, an analysis of the several competing algorithms that we have developed is still in progress.

V. Conclusions and Ongoing Research Directions

In this paper, we introduced a hybrid system modeling framework where time-driven dynamics are switched by events characterized by their own dynamics. Although this framework was motivated by problems in metalmaking, it is representative of many manufacturing processes, where the main objective is to trade off the quality of the completed jobs against the satisfaction of job deadlines, each of the two factors carrying its own costs. This leads to problems, which, in general, involve a scheduling component (i.e., deciding the order in which jobs should be processed), a server assignment component (i.e., deciding which servers should perform the processing and the order in which servers must be visited by the jobs), and an optimal control component to determine various process control settings that directly affect server parameters such as speed, service time, etc. As described in this paper, our research so far has focused on this last component of the control problem. We have assumed that a job schedule is given and examined the issues involved in determining the server parameters. Dealing with single-stage and two-stage processes, we encountered most of the difficulties characteristic of hybrid system optimal control problems, including nondifferentiability and nonconvexity of objective functions.

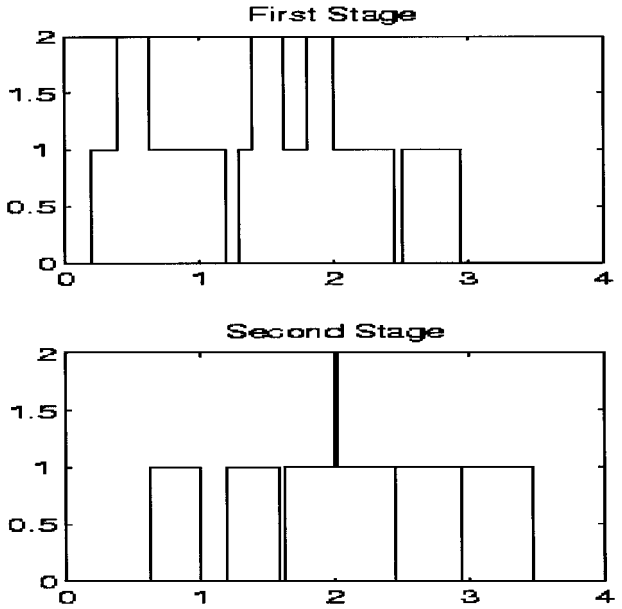


Fig. 8. Optimal sample path for a two-stage hybrid system.

Generally speaking, dynamic programming (DP) can be used to solve the optimal control problems described in this paper. In practice, however, the "curse of dimensionality" makes it numerically impractical to obtain a control policy by using DP. For deterministic problems, the problem can be phrased as a nonsmooth mathematical programming problem that can be treated with a combination of variational techniques and nonsmooth optimization theory. In general, however, these techniques still lead to solution methods no simpler than solving the DP equations. Our approach has been to seek and exploit any type of special structure to be found in our framework. The result is a "divide-and-conquer" scheme that decomposes the solution of a difficult large-scale nonsmooth optimization problem into a collection of simpler smaller scale and smooth optimization problems. For these problems, fast numerical algorithms (e.g., conjugate gradient methods) can then be used. Alternatively, we showed how Bezier functions can be used as surrogates for the nonsmooth max functions present in the event-driven dynamics to approximate the overall nonsmooth optimization problem with a smooth one. This approach can be applied to any configuration of servers that can be described using the "max-plus" algebra. In contrast, it is not yet clear whether the divide-and-conquer decomposition algorithms, which include Algorithm 1 in Section III-B, the backward algorithm in [11], and the forward algorithm in [13], can be extended beyond single-stage processes. Our ongoing research is looking into ways to extend the decomposition algorithms. We are also

conducting a systematic comparison of all of the algorithms we have developed to date to catalog their respective advantages and disadvantages.

The work presented in this paper has taken only the first steps toward analyzing a large class of hybrid systems and seeking explicit solutions. We have considered in great detail single-stage processes and identified several properties based on which we have developed several efficient solution algorithms. Extensions to multistage processes, beyond the approach presented in Section IV, still remain to be thoroughly analyzed. We have also limited ourselves to controls that are fixed over the duration of a job processing cycle, as opposed to time-varying control throughout this cycle. Some early work along these lines presented in [8] suggests a hierarchical decomposition for hybrid systems that deserves further investigation.

An obvious question also relates to the fact that the modeling framework we have studied in this paper is limited to a deterministic setting. This defines another important research direction involving stochastic hybrid system models and associated optimal control problems. Along these lines, recent work in [24] has treated the case where the job arrival time sequence or the processing times (or both) are modeled through random processes. Using DP equations, it is possible to extract some structural properties of the optimal control policy. In particular, it can be shown that simple threshold-based policies are optimal, i.e., a specific control action is taken when a certain state variable exceeds a threshold. One advantage of a threshold policy is that it converts the optimal control problem from a search over a space of functions to a simpler parametric optimization problem for determining the optimal values of the threshold parameters.

Returning to the issue of determining explicit optimal control solutions for specific problems, we believe that in the case of hybrid systems it is particularly important to take advantage of structural properties. This observation is related to the so-called *No Free Lunch Theorem* [25], which identifies the trade-off between generality and computational efficiency of optimization algorithms. A consequence of the theorem is that it is possible to outperform a general-purpose optimization algorithm (like DP) by incorporating specialized knowledge about the specific problem being solved. The complexity of hybrid system optimal control problems is such that it is very unlikely for any "general" solution procedure to be computationally feasible; a more promising direction is that of seeking structural properties in broad classes of interesting problems and developing solution methodologies tailored to these problems.

REFERENCES

[1] A. Alur, T. A. Henzinger, and E. D. Sontag, Eds., *Hybrid Systems*. Berlin, Germany: Springer-Verlag, 1996.

[2] P. Antsaklis, W. Kohn, M. Lemmon, A. Nerode, and S. Sastry, Eds., *Hybrid Systems*. Berlin, Germany: Springer-Verlag, 1998.

[3] R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, Eds., *Hybrid Systems*. Berlin, Germany: Springer-Verlag, 1993, vol. 736, Lecture Notes in Computer Science.

[4] M. Lemmon, K. He, and I. Markovsky, "Supervisory hybrid systems," *IEEE Contr. Syst. Mag.*, vol. 19, no. 4, pp. 42–55, 1999.

[5] D. Liberzon and A. Morse, "Basic problems in stability and design of switched systems," *IEEE Contr. Syst. Mag.*, vol. 19, no. 5, pp. 59–70, 1999.

[6] C. G. Cassandras, *Discrete Event Systems: Modeling and Performance Analysis*. Homewood, IL: Irwin, 1993.

[7] L. Kleinrock, *Queueing Systems*. New York: Wiley-Interscience, 1975, vol. I, Theory.

[8] K. Gokbayrak and C. G. Cassandras, "Hybrid controllers for hierarchically decomposed systems," in *Proc. 3rd Int. Workshop Hybrid Systems: Computation and Control*, March 2000, pp. 117–129.

[9] C. G. Cassandras, D. L. Pepyne, and Y. Wardi, "Optimal control of systems with time-driven and event-driven dynamics," in *Proc. 37th IEEE Conf. Decision and Control*, Dec. 1998, pp. 7–12.

[10] C. G. Cassandras, D. L. Pepyne, and Y. Wardi, Optimal control of a class of hybrid systems, submitted for publication.

[11] Y. Wardi, C. G. Cassandras, and D. L. Pepyne, Algorithm for computing optimal controls for single-stage hybrid manufacturing systems, submitted for publication.

[12] D. L. Pepyne and C. G. Cassandras, "Modeling, analysis, and optimal control of a class of hybrid systems," *J. Discrete Event Dynamic Syst.*, vol. 8, no. 2, pp. 175–201, 1998.

[13] Y. Cho, C. G. Cassandras, and D. Pepyne, Forward algorithms for optimal control of a class of hybrid systems, submitted for publication.

[14] M. S. Branicky, V. S. Borkar, and S. K. Mitter, "A unified framework for hybrid control: Model and optimal control theory," *IEEE Trans. Automat. Contr.*, vol. 43, no. 1, pp. 31–45, 1998.

[15] S. Galan and P. Barton, "Dynamic optimization of hybrid systems," *Chem. Eng.*, vol. 22, pp. S183–S190, 1998.

[16] A. E. Bryson and Y. C. Ho, *Applied Optimal Control*: Hemisphere, 1975.

[17] D. E. Kirk, *Optimal Control Theory*. Englewood Cliffs, NJ: Prentice-Hall, 1970.

[18] D. L. Pepyne, "Performance optimization strategies for discrete event and hybrid systems," Ph.D. dissetation, Dept. of Electrical and Computer Engineering, Univ. of Massachusetts, Amherst, Feb. 1999.

[19] F. H. Clarke, *Optimization and Nonsmooth Analysis*. New York: Wiley-Interscience, 1983.

[20] M. M. Makela and P. Neittaanmaki, *Nonsmooth Optimization*. Cleveland, OH: World Scientific, 1992.

[21] R. Rockafellar, *Convex Analysis*. Princeton, NJ: Princeton Univ. Press, 1970, vol. 28, Princeton Mathematics Series.

[22] N. Singh, *Systems Approach to Computer Integrated Design and Manufacturing*. New York: Wiley, 1996.

[23] C. G. Cassandras, Q. Liu, K. Gokbayrak, and D. L. Pepyne, "Optimal control of a two-stage hybrid manufacturing system model," in *Proc. 38th IEEE Conf. Decision and Control*, Dec. 1999, pp. 450–455.

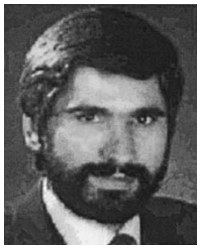
[24] K. Gokbayrak and C. G. Cassandras, "Stochastic optimal control of a hybrid manufacturing system model," in *Proc. 38th IEEE Conf. Decision and Control*, Dec. 1999, pp. 919–924.

[25] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Computat.*, vol. 1, no. 1, pp. 67–82, 1997.

David L. Pepyne (Member, IEEE) received the B.S. degree from the University of Hartford, West Hartford, CT, in 1986, and the M.S. and Ph.D. degrees from the University of Massachusetts at Amherst, Amherst, MA, in 1995 and 1999, respectively, all in electrical engineering.

From 1986 to 1990, he served as an Officer in the United States Air Force, during which time he was stationed at Edwards Air Force Base, CA, and worked as a Flight Test Engineer in a simulation laboratory and on a cruise missile test program. From 1995 to 1997, he was a Project Engineer with Alphatech, Inc., Burlington, MA. Since 1999, he has been a Research Fellow in the Division of Engineering and Applied Sciences at Harvard University, Cambridge, MA, where his research focuses on complexity theory, intrusion and fault detection, optimization theory, and optimal control of discrete-event and hybrid systems.

Dr. Pepyne is currently an Associate Editor for the IEEE Control Systems Society Conference Editorial Board.



Christos G. Cassandras (Fellow, IEEE) received the B.S. degree from Yale University, New Haven, CT, in 1977, the M.S.EE degree from Stanford University, Stanford, CA, in 1978, and the S.M. and Ph.D. degrees from Harvard University, Cambridge, MA, in 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 at the Department of Electrical and Computer Engineering, University of Massachusetts at Amherst, Amherst, MA. He is currently Professor of manufacturing engineering and Professor of electrical and computer engineering at Boston University, Boston, MA. He specializes in the areas of discrete event systems, stochastic optimization, and computer simulation, with applications to computer networks, manufacturing systems, and transportation systems. He has published more than 150 papers in these areas, and two textbooks. He has guest-edited several technical journal issues and serves on several editorial boards.

Dr. Cassandras is currently Editor-in-Chief of the IEEE TRANSACTIONS ON AUTOMATIC CONTROL and has served as Editor for Technical Notes and Correspondence and Associate Editor. He is a member of the CSS Board of Governors, chaired the CSS Technical Committee on Control Theory, and served as Program Chair of various conferences. He is the recipient of the 1999 Harold Chestnut Prize (IFAC Best Control Engineering Textbook) for *Discrete Event Systems: Modeling and Performance Analysis* and a 1991 Lilly Fellowship, and is a member of Phi Beta Kappa and Tau Beta Pi.