# Forward Decomposition Algorithms for Optimal Control of a Class of Hybrid Systems

**Young C. Cho**[†]
School of Electrical Engineering
Seoul National University, Seoul, 151-742, Korea
jyc@cisl.snu.ac.kr, Tel:(02)883-9921, Fax:(02)874-8256

**Christos G. Cassandras**[‡]
Department of Manufacturing Engineering
Boston University, Boston, MA 02215
cgc@bu.edu, Tel:(617)353-7154, Fax:(617)353-4830

**David L. Pepyne**[‡]
Division of Engineering and Applied Sciences
Harvard University, Cambridge, MA 02138
pepyne@hrl.harvard.edu, Tel:(617)495-8911, Fax:(617)495-9837

## Abstract

This paper considers optimal control problems for a class of hybrid systems motivated by the structure of manufacturing environments that integrate process and operations control. We derive new necessary and sufficient conditions that allow us to determine the structure of the optimal sample path and hence decompose a large non-convex, non-differentiable problem into a set of smaller convex, constrained optimization problems. Using these conditions, we develop two efficient, low-complexity, scalable algorithms for explicitly determining the optimal controls. Several numerical examples are included to illustrate the efficacy of the proposed algorithms.

**Key words** : Hybrid system, optimal control, nonconvex optimization

† This work is supported in part by KOSEF Fellowship.

‡ This work is supported in part by NSF under grant ACI-9873339, AFOSR under grant F49620-98-1-0387, AFRL under contract F30603-99-C-0057 and EPRI/DOD under contract WO8333-03.

## 1 Introduction

Hybrid systems are characterized by the combination of *time-driven* and *event-driven* dynamics. The former are represented by differential (or difference) equations, while the latter may be described through various frameworks used for discrete event systems (DES), such as timed automata, max-plus equations, or Petri nets (see [1]). Broadly speaking, two categories of modeling frameworks have been proposed to study hybrid systems: Those that extend event-driven models to include time-driven dynamics; and those that extend the traditional time models to include event-driven dynamics; for an overview, see [2, 3, 4].

The hybrid system modeling framework considered in this paper falls into the first category above. It is largely motivated by the structure of many manufacturing systems. In these systems, discrete entities (referred to as *jobs*) move through a network of workcenters which process the jobs so as to change their physical characteristics according to certain specifications. Each job is associated with a *temporal* state and a *physical* state. The temporal state of a job evolves according to event-driven dynamics and includes information such as the service time or departure time of the job. The physical state evolves according to time-driven dynamics and describes some measures of the "quality" of the job such as temperature, weight and chemical composition. The interaction of time-driven with event-driven dynamics leads to a natural tradeoff between temporal requirements on job completion times and physical requirements on the quality of the completed jobs.

Such modeling frameworks and optimal control problems have been considered in [5, 6, 7, 8]. By the nature of event-driven dynamics, the problem is inherently non-convex and non-differentiable. Moreover, its dimension (number of independent variables) is identical to the number of considered jobs. If this number is in the hundreds or thousands, the problem is highly complex and defies general-purpose algorithms (like dynamic programming) for its solution. In earlier work [6, 7], the task of solving these problems was simplified by exploiting structural properties of the optimal sample path. In particular, an optimal sample path is decomposed into decoupled segments, termed "busy periods". Moreover, each busy period is further decomposed into "blocks" defined by certain jobs termed *critical*; identifying such jobs and their properties was a crucial part of the analysis and the key to developing effective algorithms for solving the optimal control problems. The identification of critical jobs and busy periods have been realized using nonsmooth optimization methods [7, 9].

Recently, an efficient "backward" recursive algorithm was developed for computing the optimal controls *without* the explicit identification of critical jobs [10]. The backward algorithm also decomposes the entire optimal control problem into a set of smaller optimal control subproblems by proceeding backward in time from the last job to the first. Each of these subproblems turns out to be a convex optimization problem with linear constraints, a much simpler problem to solve than the original non-convex and non-differentiable optimal control problem. The complexity of the problem (measured in the number of convex constrained optimization problems required to solve) was thus reduced from exponential in $N$ (the number of jobs processed) to linear bounded by $2N-1$.

In this paper, we further exploit the special structure of the optimal sample path in the single-stage hybrid system framework. The resulting algorithm is also based on solving convex optimization problems with linear constraints, as in [10]. However, there are a number of differences and advantages: (i) The algorithm iterates *forward* in time and turns out to be simpler to implement and of lower computational complexity. In fact, we will show that it is always of complexity $N$ (in contrast to the *backward* algorithm in [10] which is bounded by $2N-1$). (ii) The algorithm is based on a necessary and sufficient condition which allows us to simultaneously identify the exact busy period structure of the optimal sample path, and (iii) Using this condition, a simpler proof of uniqueness of the optimal solution (compared to the one presented in [7]) is possible.

The paper is organized as follows. Section 2 reviews the framework we use for the class of hybrid systems and related optimal control problems considered. In Section 3, we derive a necessary and sufficient condition for identifying the structure of optimal sample paths and show that a solution is unique even though the problem is non-convex and non-differentiable. Section 4 presents our new forward algorithm, whose complexity and features are discussed. Section 5 presents a more efficient forward algorithm based on decomposing each busy period into smaller decoupled segments. Section 6 summarizes the paper.

## 2 A Single-Stage Hybrid System Framework

The single-stage hybrid system framework we consider is illustrated in Fig. 1. A sequence of $N$ jobs is assigned by an external source to arrive for processing at known times $0 \leq a_1 \leq \cdots \leq a_N$. We denote these jobs by $C_i$, $i = 1, \cdots, N$. The jobs are processed first-come, first-served (FCFS) by a work-conserving and non-preemptive server. The processing time is $s(u_i)$, which is a function of a control variable $u_i$. In general, the control is time varying over the course of the processing time $s_i$ (see [11]). 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 non-negative, i.e., $s(u_i) \geq 0$.

**Time-driven dynamics.** A job $C_i$ is initially at some physical state $\xi_i$ at time $x_0$ and subsequently evolves according to the *time-driven* dynamics

$$\dot{z}_i(t) = g_i(z_i, u_i, t), \quad z_i(x_0) = \xi_i. \tag{1}$$

**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 non-preemptive queue [12]:

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

where we assume $x_0 = -\infty$.

Note that the choice of control $u_i$ affects both the physical state $z_i$ and the next temporal state $x_i$, justifying the hybrid nature of the system. In this framework, each job must be processed until it reaches a certain quality level denoted by $\Gamma_i$. That is, the processing time for each job is chosen such that

$$s_i(u_i) = min\left[t \geq 0 : z_i(t_0 + t) = \int_{t_0}^{t_0+t} g_i(\tau, u_i, t)d\tau + z(t_0) \in \Gamma_i\right] \tag{3}$$

For the above single-stage framework defined by equations (1)-(2), the optimal control objective is to choose a control sequence $\{u_1, \cdots, u_N\}$ to minimize an objective function of the form

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

Although, in general, the state variables $z_1, \cdots, z_N$ evolve continuously with time, minimizing (4) is an optimization problem in which we are concerned with the values of the state variables only at the job completion times $x_1, \cdots, x_N$. Note that a cost on the physical state $z_i(x_i)$ is not included. 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 the quality objectives, i.e., $z_i(x_i) \in \Gamma_i$. More generally, 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).

In particular, in this paper, a class of hybrid problems is considered where 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; see [5] for various examples and [11] for extension to cases with time varying controls $u_i(t)$. By assuming that $s_i(\cdot)$ is either monotone increasing or monotone decreasing, given a control $u_i$, the service time $s_i$ can be uniquely determined from $s_i = s(u_i)$ and vice versa. Therefore, to simplify the exposition, we can identify the control variables with the service times, i.e., we set $s_i = u_i$ and carry out the rest of the discussion in terms of the notation $u_i$. Hence, the optimal control problem, denoted by $\mathbf{P}$, has the following form:

$$\mathbf{P} : \min_{u_1,\cdots,u_N} \left\{ J = \sum_{i=1}^{N} \{\theta_i(u_i) + \phi_i(x_i)\} \quad : \; u_i \geq 0, \quad i = 1, \cdots, N \right\} \tag{5}$$

subject to

$$x_i = \max(x_{i-1}, a_i) + u_i, \quad i = 1, \cdots, N. \tag{6}$$

The optimal solution of $\mathbf{P}$ is denoted by $u_i^*$ for $i = 1, \cdots, N$ and the corresponding departure times in (6) are denoted by $x_i^*$ for $i = 1, \cdots, N$.

In this setup, following assumptions are necessary to make the problem somewhat more tractable, yet it still captures the essence of the original problem.

**Assumption 1** [7, 10] $\theta_i(\cdot)$ is continuously differentiable, strictly convex and monotone decreasing for $u > 0$ and the following limit holds: $\lim_{u \to 0^+} \theta_i(u) = \infty$.

**Assumption 2** [7, 10] $\phi_i(\cdot)$ is continuously differentiable, strictly convex and its minimum is obtained at a finite point.

While the above assumptions seem to be somewhat restrictive, their interpretation is consistent with the previous discussion regarding tradeoffs between the temporal and physical requirements in manufacturing problems modeled through hybrid systems. In this case, the processing time $u_i$ can be interpreted as a measure of the quality of the finished $i$th 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 due dates. As an example, let

$$\theta(u_i) = \frac{1}{u_i} \quad \text{and} \quad \phi_i(x_i) = (x_i - \delta_i)^2$$

where the two functions satisfy the above assumptions. The cost on the control penalizes short service times, since this generally results in poor quality. Letting $\delta_i$ be the due date of job $i$, the cost on the departure time penalizes job earliness and tardiness. For a specific manufacturing problem application of this framework, the reader is referred to [13] where a steel heating/annealing process is modeled, analyzed, and optimized using a setting similar to (5)-(6).

## 3 Properties of Optimal Solutions

In this section, we exploit properties of an optimal sample path in order to derive necessary and sufficient conditions for identifying its structure. We begin with the following definitions:

**Definition 1** A job $C_i$ is *critical* if it departs at the arrival time of the next job, i.e. $x_i = a_{i+1}$.

**Definition 2** Consider a contiguous job subset $\{C_k, \cdots, C_n\}$, $1 \leq k \leq n \leq N$ on the optimal sample path. The subset is said to be a *block* if

1) $x_{k-1} \leq a_k$ and $x_n \leq a_{n+1}$

2) The subset contains no critical jobs.

**Definition 3** *A busy period* is a contiguous set of jobs, $C_k, \cdots, C_n$ for $1 \leq k \leq n \leq N$ such that the following three conditions are satisfied:

i) $x_{k-1} < a_k$,

ii) $x_n < a_{n+1}$,

iii) $x_i \geq a_{i+1}$, for every $i = k, \cdots, n-1$.

**Definition 4** *A busy-period structure* is a partition of the jobs $C_1, \cdots, C_N$ into busy periods.

The busy-period structure is represented by $\{\bar{B}_1, \bar{B}_2, \cdots, \bar{B}_M\}$ for some $M \in \{1, \cdots, N\}$. The $j$th busy period consists of jobs $C_{k(j)}, \cdots, C_{n(j)}$ where $k(1) = 1$, $k(j) = n(j-1) + 1$ and $n(M) = N$.

Consider the following optimization problem for $C_k, \cdots, C_n$, which is denoted by $Q(k,n)$:

$$Q(k,n) : \min_{u_k, \cdots, u_n} \left\{ J(k,n) = \sum_{i=k}^{n} \{\theta_i(u_i) + \phi_i(a_k + \sum_{j=k}^{i} u_j)\} \ : \ u_i \geq 0 \right\} \tag{7}$$

subject to

$$x_i = a_k + \sum_{j=k}^{i} u_j \geq a_{i+1}, \quad i = k, \cdots, n-1. \tag{8}$$

Since the cost functional is continuously differentiable and strictly convex, the problem $Q(k,n)$ is also a convex optimization problem with linear constraints and has a unique solution at a finite point (see [10]). The minimal cost of $Q(k,n)$ is denoted by $\bar{J}(k,n)$, the solution of $Q(k,n)$ is denoted by $u_j^*(k,n)$ for $j=k,\cdots,n$, and the corresponding departure times are denoted by $x_j^*(k,n)$ for $j=k,\cdots,n$. Note that the equality $a_k+\sum_{j=k}^{i}u_j=a_{i+1}$, for some $i=k,\cdots,n-1$ is satisfied at the solution to $Q(k,n)$ if and only if job $C_i$ is *critical.*

Problem $Q(k,n)$ is of the same form as $\mathbf{P}$, but it is limited to jobs $C_k,\cdots,C_n$ constrained through (8) to all belong to the same busy period. There are two important properties of $Q(k,n)$ that are captured in the two lemmas that follow.

**Lemma 1** If jobs, $C_k,\cdots,C_n$ constitute a single busy period on the optimal sample path, then the solution, $u_i^*$, $i=k,\cdots,n$ is identical to $u_i^*(k,n)$, $i=k,\cdots,n$ respectively.

Proof : From (6) and Definition 3, if jobs $C_k,\cdots,C_n$ constitute a single busy period on the optimal sample path, then $a_k+\sum_{j=k}^{i}u_j\geq a_{i+1}$, for all $i=k,\cdots,n-1$. This constraint is identical to the constraint (8) which $Q(k,n)$ is subject to. In addition, due to the idle period decoupling property (Lemma 4.1 in [7]), the solution, $u_i^*$ for $i=k,\cdots,n$, on the optimal sample path is identical to $u_i^*(k,n)$ for $i=k,\cdots,n$ of problem $Q(k,n)$. ■

An implication of Lemma 1 is that if the busy period structure obtained with the optimal controls in problem $\mathbf{P}$ were known in advance, then the optimal controls could be obtained by solving a differentiable, convex problem of the form $Q(k,n)$ for each busy period. The following result provides a new necessary and sufficient condition for identifying busy periods on an optimal sample path based on the solution of $Q(k,n)$.

**Lemma 2** Let $C_k$ initiate a busy period on an optimal sample path and suppose $C_k,\cdots,C_n$ all belong to this busy period. Then, $C_k,\cdots,C_n$ constitute a single busy period on the optimal sample path if and only if:

$$x_n^*(k,n)<a_{n+1}. \tag{9}$$

Proof : (*Necessity*) Since jobs $C_k,\cdots,C_n$ constitute a single busy period on the optimal sample path, the optimal control $u_i^*$, for each $i=k,\cdots,n$ is identical to the solution of $Q(k,n)$, $u_i^*(k,n)$ for $i=k,\cdots,n$, by Lemma 1. It follows that the departure time of $C_n$, $x_n^*$, on the optimal sample path is equal to $x_n^*(k,n)$. Then, $x_n^*(k,n)=x_n^*<a_{n+1}$ because $C_n$ is the last job in the busy period defined by $C_k,\cdots,C_n$. This establishes (9).

(*Sufficiency*) By assumption, a busy period consists of $C_k,\cdots,C_{n'}$ jobs for some $n'\geq n$ on the optimal sample path. Assume that $n'>n$. We will then show that

$$\bar{J}(k,n')>\bar{J}(k,n)+\bar{J}(n+1,n') \tag{10}$$

under the constraint (9), which means that the combined cost of two busy periods formed by $C_k, \cdots, C_n$ and $C_{n+1}, \cdots, C_{n'}$ is lower than the cost of one busy period consisting of $C_k, \cdots, C_{n'}$. This is a contradiction, since $C_k, \cdots, C_{n'}$ are assumed to constitute a single busy period on the *optimal* sample path. This would imply that $n' = n$. Therefore, it remains to prove that (10) indeed holds under (9) and the assumption $n' > n$.

The minimal cost, $\bar{J}(k, n')$, of problem $Q(k, n')$ may be written as

$$\begin{aligned}\bar{J}(k,n') &= \sum_{i=k}^{n'}[\theta_i(u_i^*(k,n')) + \phi_i(x_i^*(k,n'))] \\ &= \sum_{i=k}^{n}[\theta_i(u_i^*(k,n')) + \phi_i(x_i^*(k,n'))] + \sum_{i=n+1}^{n'}[\theta_i(u_i^*(k,n')) + \phi_i(x_i^*(k,n'))] \quad (11)\end{aligned}$$

subject to

$$x_i^*(k,n') = a_k + \sum_{j=k}^{i} u_j^*(k,n') \geq a_{i+1}, \text{ for } i = k, \cdots, n'-1. \quad (12)$$

Let the first term on the right-hand-side of (11) be denoted by $J_A(k, n')$, and the second term be denoted by $J_B(k, n')$, i.e.,

$$\bar{J}(k,n') = J_A(k,n') + J_B(k,n').$$

Since $x_n^*(k, n) < a_{n+1}$ from (9) and $x_n^*(k, n') \geq a_{n+1}$ from (12), it is obvious that $u_i^*(k, n') \neq u_i^*(k, n)$ for some $i \in \{k, \cdots, n\}$. Then,

$$\begin{aligned} J_A(k,n') \;>\; \bar{J}(k,n) = \min_{u_k,\cdots,u_n}\left\{\sum_{i=k}^{n}[\theta_i(u_i)+\phi_i(x_i)] \;:\; u_i \geq 0, \forall i\right\} \quad (13) \\ \text{subject to } x_i = a_k + \sum_{j=k}^{i} u_j \geq a_{i+1}, i = k, \cdots, n-1 \end{aligned}$$

because the optimal control, $u_i^*(k, n)$ for $i = k, \cdots, n$ in $Q(k, n)$ is unique. In addition,

$$\begin{aligned} J_B(k,n') &= \sum_{i=n+1}^{n'}\left[\theta_i(u_i^*(k,n')) + \phi\left(x_n^*(k,n') + \sum_{j=n+1}^{i} u_j^*(k,n')\right)\right] \\ &\quad \text{subject to } x_n^*(k,n') + \sum_{j=n+1}^{i} u_j^*(k,n') \geq a_{i+1}, \; i = n+1, \cdots, n'-1 \\ &\quad \text{and } u_i^*(k,n') \geq 0, \quad i = n+1, \cdots, n' \end{aligned}$$

Now, set $u_{n+1}^*(k, n') + x_n^*(k, n') - a_{n+1} \overset{\triangle}{=} u_{n+1}$ and $u_i^*(k, n') \overset{\triangle}{=} u_i$ for $i = n+2, \cdots, n'$. In addition, for simplicity, let $\alpha \overset{\triangle}{=} x_n^*(k, n') - a_{n+1}$ and note that $\alpha \geq 0$ by (12). Then, we can rewrite $\bar{J}_B(k, n')$ as

$$\bar{J}_B(k,n') = \theta_i(u_{n+1} - \alpha) + \sum_{i=n+2}^{n'} \theta_i(u_i) + \sum_{i=n+1}^{n'} \phi\left(a_{n+1} + \sum_{j=n+1}^{i} u_j\right)$$

$$
\begin{aligned}
&\qquad \text{subject to} \quad a_{n+1} + \sum_{j=n+1}^{i} u_j \geq a_{i+1},\ i = n+1, \cdots, n'-1, \\
&\qquad u_{n+1} \geq \alpha \text{ and } u_j \geq 0,\ i = n+2, \cdots, n' \\
\geq\ & \sum_{i=n+1}^{n'} \left[ \theta_i(u_i) + \phi\left( a_{n+1} + \sum_{j=n+1}^{i} u_j \right) \right] && (14) \\
&\qquad \text{subject to} \quad a_{n+1} + \sum_{j=n+1}^{i} u_j \geq a_{i+1},\ i = n+1, \cdots, n'-1, \\
&\qquad u_{n+1} \geq \alpha \text{ and } u_j \geq 0,\ i = n+2, \cdots, n' \\
\geq\ & \min_{u_{n+1},\cdots,u_{n'}} \left\{ \sum_{i=n+1}^{n'} \left[ \theta_i(u_i) + \phi\left( a_{n+1} + \sum_{j=n+1}^{i} u_j \right) \right] : u_i \geq 0, \forall i \right\} && (15) \\
&\qquad \text{subject to} \quad a_{n+1} + \sum_{j=n+1}^{i} u_j \geq a_{i+1},\ i = n+1, \cdots, n'-1 \\
=\ & \bar{J}(n+1, n') && (16)
\end{aligned}
$$

The inequality in (14) is obtained because $\alpha \geq 0$ and $\theta_i(\cdot)$ is monotone decreasing by Assumption 1. The inequality in (15) is obtained because we relax the constraint on $u_{n+1}$ from $u_{n+1} \geq \alpha$ to $u_{n+1} \geq 0$. The last equality in (16) is obtained by the definition of problem $Q(n+1, n')$. Finally, from (13) and (16),

$$\bar{J}(k, n') = J_A(k, n') + J_B(k, n') > \bar{J}(k, n) + \bar{J}(n+1, n') \qquad (17)$$

which proves (10) and therefore completes the proof. ■

Lemma 2 allows us to identify busy periods on the optimal sample path. For $C_k, \cdots, C_n$, if $C_k$ is the first job of a busy period on the optimal sample path, we can identify the busy period by sequentially solving $Q(k, i)$ and checking if $x_i^*(k, i) \geq a_{i+1}$, for $i = k+1, \cdots, n$. This idea is formalized in the following theorem.

**Theorem 1** Jobs $C_k, \cdots, C_n$ jobs constitute a single busy period on the optimal sample path if and only if the following conditions are satisfied:

1) $a_k > x_{k-1}^*$

2) $x_i^*(k, i) \geq a_{i+1}$, for all $i = k, \cdots, n-1$,

3) $x_n^*(k, n) < a_{n+1}$.

Proof : (*Necessity*) Since $C_k, \cdots, C_n$ constitute a single busy period on the optimal sample path, Condition 1) is met by Definition 3, and Condition 3) is met directly by Lemma 2. Condition 2) can be proved by a contradiction argument. Suppose that $x_{\bar{k}}^*(k, \bar{k}) < a_{\bar{k}+1}$, for some $\bar{k} \in \{k, \cdots, n-1\}$; this means $C_k, \cdots, C_{\bar{k}}$ constitute a single busy period on the optimal sample path by Lemma 2, which contradicts the fact that $C_k, \cdots, C_n$ constitute a single busy period. Thus, $x_i^*(k,i) \geq a_{i+1}$, for all $i = k, \cdots, n-1$ should be satisfied.

(*Sufficiency*) If $k = n$, then $a_{n+1} > x_n^*(n,n)$ which means that $C_k$ alone forms a busy period on the optimal sample path by Lemma 2. Thus, $x_n^* = x_n^*(n,n)$ by Lemma 1. For $k < n$, we can proceed by contradiction. First, suppose that $C_k, \cdots, C_n$ form two busy periods on the optimal sample path: $\bar{B}_1 = \{C_k, \cdots, C_{\bar{k}}\}$ and $\bar{B}_2 = \{C_{\bar{k}+1}, \cdots, C_n\}$, for some $\bar{k} \in \{k+1, \cdots, n-1\}$. Then, $x_{\bar{k}}^* < a_{\bar{k}+1}$ holds by the definition of a busy period. Since $\bar{B}_1$ is a single busy period on the optimal sample path, the optimal departure time of the $\bar{k}$-th job is $x_{\bar{k}}^* = x_{\bar{k}}^*(k,\bar{k})$ by Lemma 1, where $x_{\bar{k}}^*(k,\bar{k})$ is obtained by solving $Q(k,\bar{k})$. Then, $x_{\bar{k}}^*(k,\bar{k}) = x_{\bar{k}}^* < a_{\bar{k}+1}$. This is a contradiction because $x_{\bar{k}}^*(k,\bar{k}) \geq a_{\bar{k}+1}$ by Condition 2). Thus, at least jobs $C_k, \cdots, C_n$ lie within a single busy period on the optimal sample path. The same argument holds for more than two busy periods, up to $n-k+1$. In addition, since $x_n^*(k,n) < a_{n+1}$ by Condition 3), it follows from Lemma 2 that $C_k, \cdots, C_n$ constitute a single busy period on the optimal sample path. ■

The uniqueness of the optimal solution of **P** under Assumption 1-2 was first established in [7] using arguments relying on the optimality conditions formulated through generalized gradients. Theorem 1 above, however, provides simpler means for proving uniqueness, as follows.

**Lemma 3** Under Assumptions 1-2, the busy period structure of an optimal sample path is unique in the sense that for any $C_i$, $i = 1, \cdots, N$, the last job of the busy period containing $C_i$ is unique on the optimal sample path.

Proof : The proof is by contradiction. In particular, suppose that, for a given arrival sequence, there exist two different sample paths that both satisfy optimality. Due to the idle period decoupling property (Lemma 4.1 in [7]), we can assume, without loss of generality, that the difference between the two sample paths is in their respective first busy periods. Let us denote the two sample paths by $A$ and $B$. Let $n_A$ be the last job in the first busy period on sample path $A$, and $n_B$ be the last job in the first busy period on sample path $B$. Assume (without loss of generality) that $n_A < n_B$. Applying condition 3) in Theorem 1 to sample path $A$, we have

$$x_{n_A}^*(1, n_A) < a_{n_A+1}. \tag{18}$$

On the other hand, applying condition 2) in Theorem 1 to sample path $B$ for $i = n_A$, we get

$$x_{n_A}^*(1, n_A) \geq a_{n_A+1} \tag{19}$$

since $n_A \in \{1, \cdots, n_B - 1\}$. The two inequalities (18) and (19) clearly contradict each another, implying that $n_A = n_B$, i.e., the busy periods must coincide, and the proof is complete. ■

Given the uniqueness of the busy period structure, the controls within each busy period are unique, and hence the uniqueness of the entire optimal control sequence is proved as follows.

**Theorem 2** Under Assumptions 1-2, the optimal control sequence of $\mathbf{P}$ is unique.

Proof : From Lemma 3, the order of the busy period and its composition $\{C_k, \cdots, C_n\}$ on the optimal sample path are unique. The optimal control sequence $\{u_k^*, \cdots, u_n^*\}$ is obtained by solving problems $Q(k,n)$ and it is unique because $Q(k,n)$ is a convex program (a detailed proof of this fact is in Theorem 5.1 in [7]). ■

## 4 Forward Algorithm I

Theorem 1 provides the basis for a forward algorithm presented in this section. By "forward" we mean that the optimal sample path is constructed starting with job 1 and proceeding forward in time without the need for multiple forward-backward sweeps involved in a solution based on the framework of a two-point-boundary-value problem. Before presenting the algorithm, we explain the basic idea. Let $k = 1$ and $n = 1$. We already know that $C_1$ is the first job of a busy period. We then solve $Q(1,1)$ and check whether $x_1^*(1,1) < a_2$. If $x_1^*(1,1) < a_2$, then $C_1$ forms a single busy period on the optimal sample path by Theorem 1. Consequently, the first job of the next busy period is $C_2$. If, on the other hand, $x_1^*(1,1) \geq a_2$, then $C_1$ alone cannot form a busy period; in this case, we increment $n$ by 1, i.e., set $n = 2$. We then solve $Q(1,2)$ and check whether $x_2^*(1,2) < a_3$. If $x_2^*(1,2) < a_3$, then $C_1$ and $C_2$ form a single busy period on the optimal sample path by Theorem 1 and the optimal controls are obtained by solving $Q(1,2)$. If, on the other hand, $x_2^*(1,2) \geq a_3$, then $C_1$ and $C_2$ cannot form a busy period on the optimal sample path and we increment $n$ by 1, i.e., set $n = 3$. We then repeat the procedure over all jobs.

Given the arrival times of jobs $C_1, \cdots, C_N$, the optimal problem $\mathbf{P}$ can be solved by Forward Algorithm I shown in Table 1. In **Step 2**, we solve the linearly constrainted convex optimization problem $Q(k,n)$ and obtain the control $u_j^*(k,n), j = k, \cdots, n$ and departure times $x_j^*(k,n), j = k, \cdots, n$. In **Step 3**, the structure of busy periods is identified by checking if $x_n^*(k,n) < a_{n+1}$. If $C_k, \cdots, C_n$ are identified as a single busy period, the control on the optimal path for $C_k, \cdots, C_n$ is given by $u_j^*(k,n), j = k, \cdots, n$. Then, we set the index of the first job of a new busy period as $n+1$, i.e. $k=n+1$.

**Remark 1** This algorithm requires only $N$ iterations, because in **Step 4**, $n$ is increased by 1 at every iteration. Therefore, Forward Algorithm I must solve $N$ subproblems to obtain the optimal solution. The dimensionality of each of these $N$ problems depends on the given arrival sequence.

**Remark 2** In Forward Algorithm I, the index $k$ always *indicates the first job* of all busy periods on the optimal sample path. This is because $k$ is updated as $k=n+1$ in **Step 3** only when the last job of a busy period is identified to be $C_n$.

**Remark 3** (Best case): If each job constitutes a single busy period on the optimal sample path, the optimal control is obtained by solving a convex problem $Q(i,i)$, for $i=1,\cdots,N$. This is the best case in the sense that Forward Algorithm I requires the smallest computation because the dimensionality of each $Q(i,i)$ is just 1 job.

**Remark 4** (Worst case): If all $N$ jobs are in a single busy period on the optimal sample path, the algorithm needs to solve problem $Q(1,k)$, for $k=1,\cdots,N$ at each iteration and the optimal control is ultimately obtained as the solution of $Q(1,N)$. This is the worst case in the sense that the algorithm requires the largest possible computation, i.e., solving $Q(1,k)$, for all $k=1,\cdots,N$.

## 4.1 Numerical Examples

In this subsection, a few examples are presented to illustrate some features of Forward Algorithm I. Let us consider the following problem with $N=5$ :

$$\min_{u_1,\cdots,u_5} J=\sum_{i=1}^{5}\{u_i^{-1}+x_i^2\} \quad \text{subject to } x_i=\max(a_i,x_{i-1})+u_i \tag{20}$$

for the arrival sequence $\{0.2,0.6,0.9,1.8,2.1\}$. Figure 2 shows the progress of the algorithm as it proceeds job by job toward the final solution. At the first step, $Q(1,1)$ for $k=n=1$ is solved. Since $x^*{}_1(1,1)=0.932>a_2$, we set $n=2$. Then, $Q(1,2)$ is solved. Since $x^*{}_2(1,2)=1.315>a_3$, we set $n=3$. Then, $Q(1,3)$ is solved. Since $x^*{}_3(1,3)=1.595<a_4$, jobs $C_1,C_2$ and $C_3$ constitute a single busy period. The optimal controls for this busy period are obtained by solving $Q(1,3)$. Then, we set $k=4$ and $n=4$ where $k$ is the index

of the first job of the next busy period. Next, $Q(4,4)$ is solved and $x^{*}{}_{4}(4,4) = 2.269$ is obtained. Since $x^{*}{}_{4}(4,4) = 2.269 > a_5$, we set $n = 5$. Finally, by solving Q(4,5), we obtain the optimal solution. Observe that $x_1^* = a_2 = 0.6$, i.e., $C_1$ is a critical job on the optimal sample path automatically detected as part of the solution to $Q(1,3)$.

Additional examples are presented in Fig. 3 to show the best case, a case that includes a critical job, and the worst case. Each example uses $\theta_i(u_i) = \frac{1}{u_i}$ and $\phi_i(x_i) = x_i^2$. The arrival sequences are $\{1,2,3,4,5\}$, $\{1,1.4,1.5,1.55,1.6\}$ and $\{1,1.1,1.2,1.3,1.4\}$ respectively. In the best case of Fig. 3, the optimal processing time assigned to each job is such that it departs before the next job arrives, i.e., each job constitutes a single busy period by itself. In this case, the optimal control is obtained by solving $Q(i,i)$, for $i = 1,\cdots,5$, which results in the minimal computation load. In the worst case of Fig. 3, each job cannot be completed until the next job arrives, i.e., all jobs constitute a single busy period. In this case, the optimal control is obtained by solving for $Q(1,i)$, for $i = 1,\cdots,5$, which results the maximal computation load.

## 5 Forward Algorithm II

In this section, we present a variant of Forward Algorithm I which is more efficient by exploiting a feature of critical jobs in a single busy period. Without loss of generality, we assume that jobs $C_1,\cdots,C_n$ constitute a single busy period (i.e., we set $k = 1$).

**Lemma 4** Consider a single busy period consisting of jobs $C_1,\cdots,C_n$ on the optimal sample path. If $x_k^*(1,k) = a_{k+1}$ for some $k \in \{1,\cdots,n\}$, then the optimal departure time of $C_k$ on this busy period is identical to $a_{k+1}$, i.e. $x_k^*(1,n) = a_{k+1}$.

Proof : Since $x_k^*(1,n)$ is the optimal departure time on the busy period, $a_{k+1} \le x_k^*(1,n)$ is satisfied. Suppose that $a_{k+1} < x_k^*(1,n)$. As in the proof of Lemma 2, we can show that $\bar{J}(1,n) > \bar{J}(1,k) + \bar{J}(k+1,n)$ (detail omitted), which means that under the assumption that $a_{k+1} < x_k^*(1,n)$, there exists another control sequence $\{u_j, j = 1,\cdots,n\}$ for which the cost is less than $\bar{J}(1,n)$. This is a contradiction, because $\bar{J}(1,n)$ is the minimal cost. Thus, $x_k^*(1,n) = a_{k+1}$. ■

**Definition 5** Consider a contiguous job subset $\{C_k,\cdots,C_n\}$, $1 \le k \le n \le N$ on the optimal sample path. The subset is said to be a *zone* if

1) $x_{k-1} \le a_k$

2) $x_i^*(k,i) > a_{i+1}$, for all $i = k, \cdots, n-1$,

3) $x_n^*(k,n) \leq a_{n+1}$.

By the definition of block and zone and Lemma 4, a zone consists of a number of blocks and a single busy period consists of a number of zones. An illustration of the relation between blocks, zones, and busy periods is shown in Fig. 4 where $x_1^*(1,1) > a_2$, $x_2^*(1,2) > a_3$ and $x_3^*(1,3) = a_4$.

Lemma 4 states that even in a single busy period on the optimal sample path, the convex optimization problem $Q(1,n)$ can be decomposed into a number of smaller convex problems $Q(1,k)$ and $Q(k+1,n)$ for some $k$ if the condition $x_k^*(1,k) = a_{k+1}$ is satisfied. This is stated formally in the following theorem.

**Theorem 3** Consider a single busy period consisting of jobs $C_1, \cdots, C_n$. If $x_k^*(1,k) = a_{k+1}$, then the solution of $Q(1,n)$ can be obtained by solving $Q(1,k)$ and $Q(k+1,n)$, i.e.,

$$u_j^* = \begin{cases} u_j^*(1,k) & \text{for } j = 1, \cdots, k \\ u_j^*(k+1,n), & \text{for } j = k+1, \cdots, n \end{cases} \tag{21}$$

Proof : This is obvious from Lemma 4. ■

Based on Theorem 3, we propose a more efficient algorithm (called Forward Algorithm II) than Forward Algorithm I, as shown in Table 2. The only difference from Forward Algorithm I is that '$x^*{}_n(k,n) < a_{n+1}$' is replaced by '$x^*{}_n(k,n) \leq a_{n+1}$' in **Step 3**. It should be clear that Forward Algorithm II also requires $N$ iterations and that the index $k$ always indicates the first job of all the zones on the optimal sample path.

**Remark 5** (Best case compared with Forward Algorithm I): If each job, $C_k, \cdots, C_n$, in a single busy period constitutes a single zone on the optimal sample path, then the optimal controls are obtained by solving each convex problem $Q(i,i)$, for $i = k, \cdots, n$. This is the best case, compared with Forward Algorithm I, in the sense that Forward Algorithm I has to solve the larger problem $Q(k,n)$.

**Remark 6** (Worst case compared with Forward Algorithm I): If a single busy period, $C_k, \cdots, C_n$, is identical to a single zone on the optimal sample path, the optimal controls are obtained by solving $Q(k,n)$. This is the worst case, compared with Forward Algorithm I, in the sense that the algorithm has to solve the same problem as Forward Algorithm I.

### 5.1 Numerical Examples

In this subsection, best and worst case examples are presented to illustrate Forward Algorithm II. Let us consider the following problem with $N = 5$:

$$\min_{u_1,\cdots,u_5} J = \sum_{i=1}^{5}\{u_i^{-1} + x_i^2\} \quad \text{subject to } x_i = \max(a_i, x_{i-1}) + u_i^2 \tag{22}$$

With the arrival sequence $\{0.4, 0.844, 1.196, 1.499, 1.771\}$, Figure 5 shows the optimal sample path for the best case satisfying $x_i^*(1,i) = a_{i+1}$, for all $i = 1,\cdots,4$. At each iteration, the size of the subproblem is just 1 job, because $k = n$ at each iteration. Finally, the optimal control is obtained by solving $Q(i,i)$ for $i = 1,\cdots,5$. Note that each job is a critical job on the optimal sample path.

Another example illustrates a worst case with the arrival sequence $\{0.4, 0.8, 1.1, 1.3, 1.5\}$. In this optimal sample path we have $x_i^*(1,i) > a_{i+1}$, for all $i = 1,\cdots,4$, as shown in Figure 6. We can observe that the size of the problem is increasing at each iteration, because $k$ is fixed at 1 while $n$ is increasing. Finally, the optimal control is obtained by solving $Q(1,5)$. Note that each job is also critical on the optimal sample path, i.e. $x_i^*(1,5) = a_{i+1}$ $i = 1,\cdots,4$.

## 6 Conclusions

This paper has considered optimal control problems defined on a single-stage hybrid system motivated from manufacturing environments. The control variables are comprised of the service times of the various jobs and the performance metrics involve measures of quality requirements and time delivery requirements of the completed jobs. This optimal control problem is inherently neither convex nor differentiable because of the nature of event-driven dynamics. We presented necessary and sufficient conditions to identify the busy period structure of the optimal sample path and derived an efficient and low-complexity, scalable algorithm for computing optimal controls. This algorithm iterates forward in time, decomposing the optimal sample path into a number of decoupled segments, i.e., busy periods and zones, and solving small-scale convex optimization problems with linear constraints. Its complexity is just the number of considered jobs $N$.

Ongoing work is aimed at extending our approach to systems with uncertainties in arrival times of jobs and to more complex dynamics encountered in multi-stage processes. In addition, we believe that the forward approach will enable us to make use of a receding horizon control scheme for a system with an infinite number of sequential jobs.

Finally, it is obviously of interest to apply an optimal control framework in order to determine time-varying controls $u_i(t)$, a problem not addressed in this paper. A hierarchical decomposition approach for this purpose was introduced in [11]. This approach may be used for hybrid system models beyond the manufacturing context considered here, as recently described in [14].

## References

[1] C. G. Cassandras. *Discrete Event Systems: Modeling and Performance Analysis.* Irwin Publ., 1993.

[2] R. Alur, T.A. Henzinger and E.D. Sontag, editors. *Hybrid Systems.* Springer-Verlag, 1996.

[3] P. Antsaklis, W. Kohn, M. Lemmon, A. Nerode and S. Sastry, editors. *Hybrid Systems V.* Springer-Verlag, 1998.

[4] M.S. Branicky, V.S. Borkar and S.K. Mitter. A unified framework for hybrid control: Model and optimal control theory. *IEEE Trans. on Automatic Control* 1998; **43**:31-45.

[5] D. L. Pepyne and C. G. Cassandras. Modeling, analysis, and optimal control of a class of hybrid systems. *Journal of Discrete Event Dynamic Systems: Theory and Applications* 1998;**8**:175-201.

[6] C. G. Cassandras, D. L. Pepyne and Y. Wardi. Optimal control of systems with time-driven and event-driven dynamics. *In Proc. of 37th IEEE Conf. on Decision and Control* 1998; 7-12.

[7] C. G. Cassandras, D. L. Pepyne and Y. Wardi. Optimal control of a class of hybrid systems. *to appear in IEEE Trans. on Automatic Control*, 2001.

[8] C. G. Cassandras, Q. Liu, K. Gokbayrak, and D. L. Pepyne. Optimal control of a two-stage hybrid manufacturing system model. In *Proceedings of 38th IEEE Conf. On Decision and Control* 1999;450-455.

[9] F.H. Clarke. *Optimization and Nonsmooth Analysis.* Wiley-Interscience, 1983.

[10] Y. Wardi, C. G. Cassandras and D. L. Pepyne. Algorithm for computing optimal controls for single-stage hybrid manufacturing systems. *to appear in Intl. J. of Production Research*, 2001.

[11] K. Gokbayrak and C. G. Cassandras. Hybrid controllers for hierarchically decomposed systems. *In Proc. of 3rd Intl. Workshop on Hybrid Systems: Computation and Control* 2000; 117–129.

[12] L. Kleinrock, editor. *Queuing Systems*, volume I. Wiley-Interscience, 1975.

[13] Young C. Cho and C. G. Cassandras. Optimal control for steel annealing processes as hybrid systems. *to appear in Proc. of 39th IEEE Conf. on Decision and Control*, 2000.

[14] K. Gokbayrak and C. G. Cassandras. A hierarchical decomposition method for optimal control of hybrid systems. *to appear in Proc. of 39th IEEE Conf. on Decision and Control*, 2000.

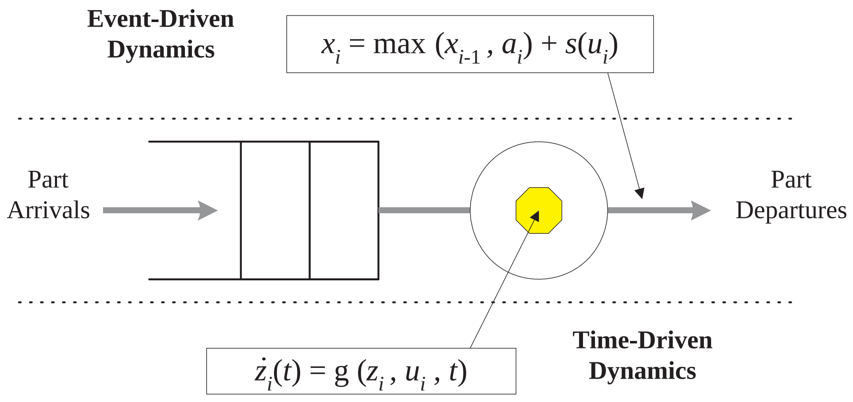

Figure 1: A single stage hybrid system.

Table 1: Forward Algorithm I

```
Step 1 : (initialization) $k = 1, n = 1, a_{N+1} = \infty$;
while $n \leq N$ do
    Step 2 : solve sub-optimal problem $Q(k, n)$;
    Step 3 : (identify single busy periods.)
            if ${x^*}_n(k, n) < a_{n+1}$ then
                ${u^*}_j \leftarrow {u^*}_j(k, n)$ for $j = k, \cdots, n$.
                $k \leftarrow n + 1$;
            endif
    Step 4 : (increment index $n$.)
            $n \leftarrow n + 1$;
 end while
```

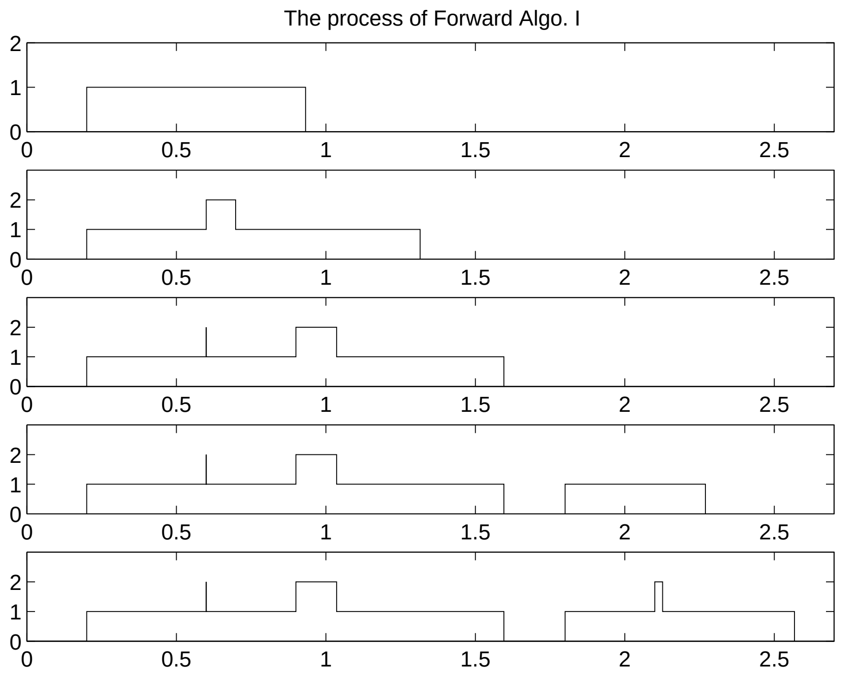

Figure 2: An illustration of the process of Forward Algorithm I.

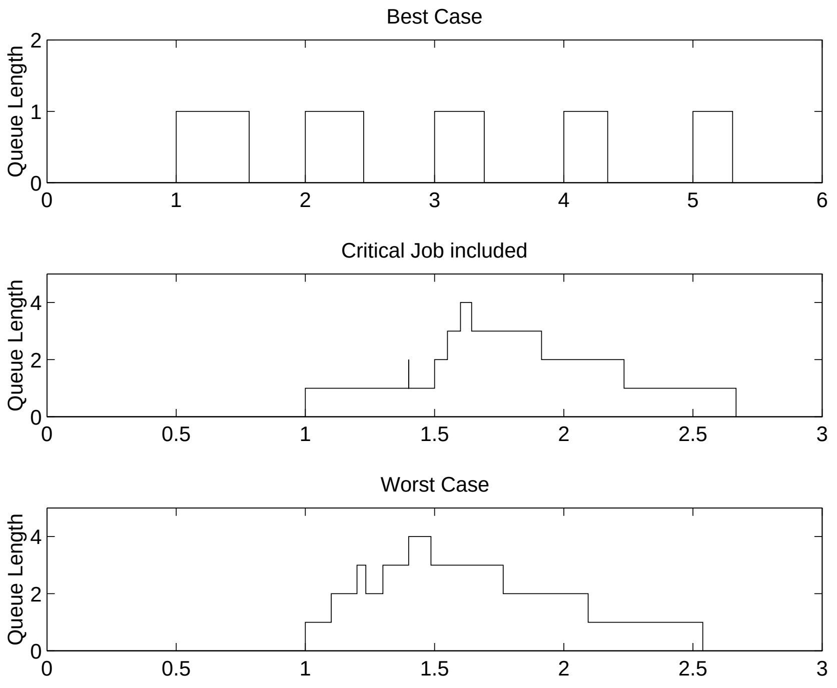

Figure 3: Some examples of Forward Algorithm I .

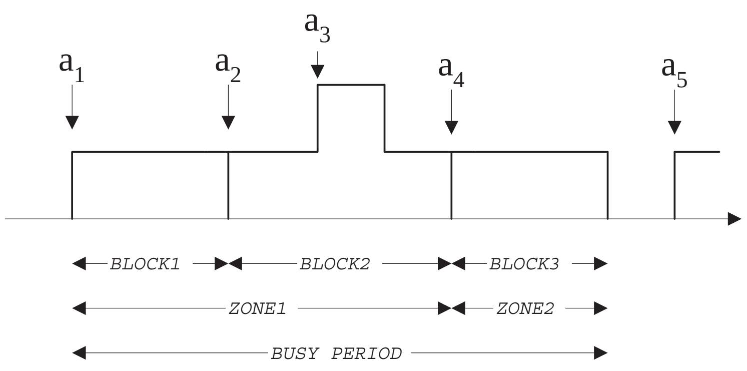

Figure 4: Three blocks and two zones in a busy period.

Table 2: Forward Algorithm II

```
Step 1 : (initialization) $k = 1, n = 1, a_{N+1} = \infty$;
while $n \leq N$ do
    Step 2 : solve sub-optimal problem $Q(k, n)$;
    Step 3 : (check single zones.)
            if ${x^*}_n(k, n) \leq a_{n+1}$ then
                ${u^*}_j \leftarrow {u^*}_j(k, n)$ for $j = k, \cdots, n$.
                $k \leftarrow n + 1$;
            endif
    Step 4 : (increment index $n$.)
            $n \leftarrow n + 1$;
end while
```

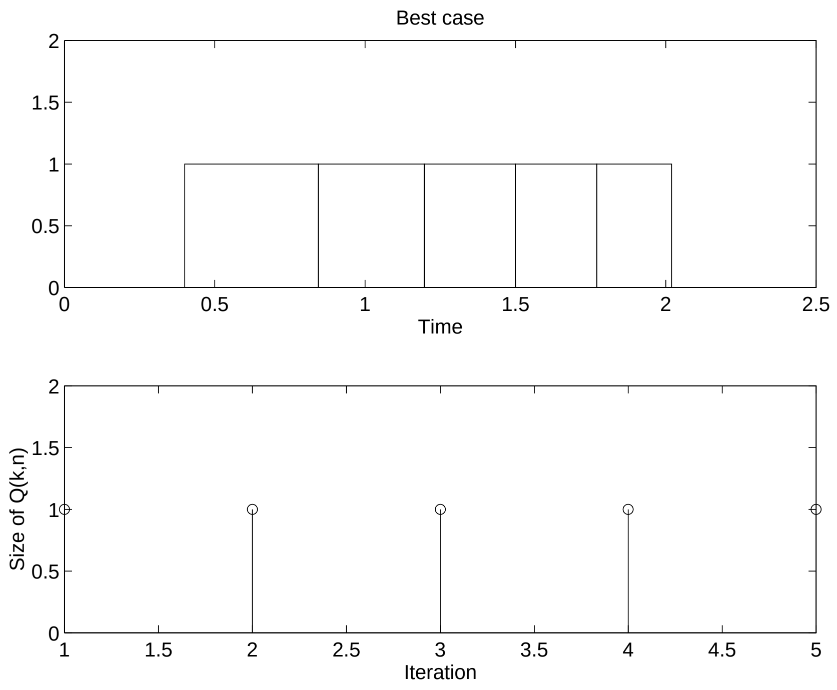

Figure 5: Best case of Forward Algorithm II.

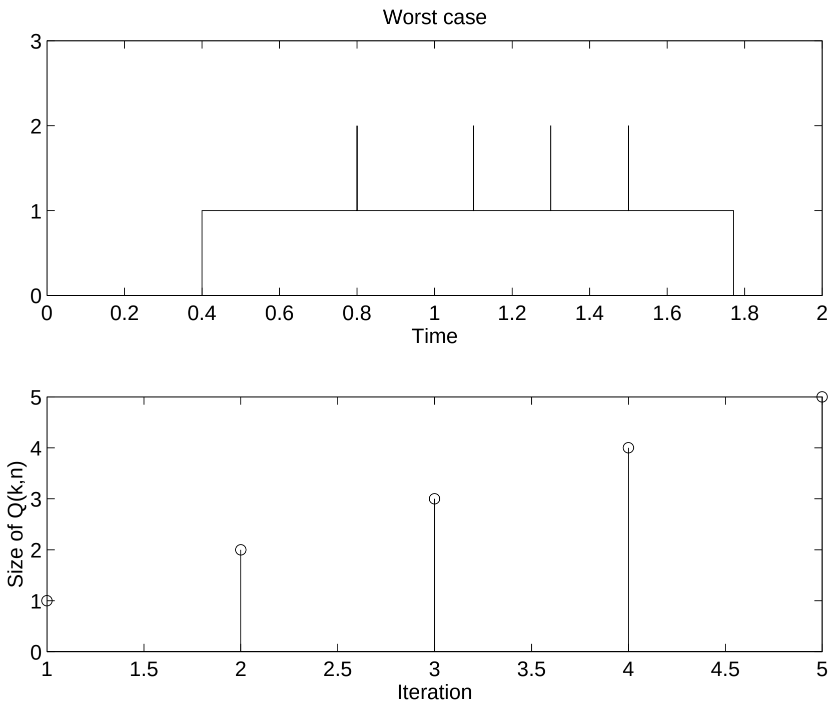

Figure 6: Worst case of Forward Algorithm II.