Forward Algorithms for Optimal Control of a Class of Hybrid Systems

Young C. Cho[1], Christos G. Cassandras[2], and David L. Pepyne[3]
[1] Sch. of Electrical Engr., Seoul Nat'l Univ., Korea, jyc@cisl.snu.ac.kr
[2] Dept. of Manufacturing Engr., Boston Univ., Boston, MA, cgc@bu.edu
[3] Div. of Engr. and Applied Sciences, Harvard Univ., Cambridge, MA, pepyne@hrl.harvard.edu

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 an efficient, low-complexity, scalable algorithm for explicitly determining the optimal controls. Several numerical examples are included to illustrate the efficacy of the proposed algorithm.

Key words : Hybrid system, optimal control, nonconvex optimization

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]. 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], 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 [8, 6].

Recently, an efficient "backward" recursive algorithm was developed for computing the optimal controls *without* the explicit identification of critical jobs [9]. 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

[1] This work is supported in part by Brain Korea and KOSEF.
[2] 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.

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 [9]. 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 [9] 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 is possible.

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. 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 ξ_i at time x_0 and subsequently evolves according to the *time-driven* dynamics

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

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 [10]:

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

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. For the above single-stage framework defined by equations (1)-(2), the optimal control objective is to choose a control sequence

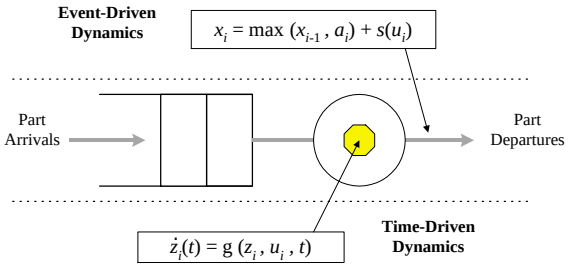

Figure 1: A single stage hybrid system.

$\{u_1, \cdots, u_N\}$ to minimize an objective function of the form

$$J = \sum_{i=1}^{N} \{\theta(u_i) + \phi(x_i)\}. \tag{3}$$

In a general setup, u_i is a control variable affecting the processing time through $s_i = s(u_i)$; see [5] for various examples. 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 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 which satisfies $u_i \geq 0$ for all $i = 1, \cdots, N$. Hence, the optimal control problem, denoted by $\mathbf{P}$, has the following form:

$$\mathbf{P}: \min_{u_1, \cdots, u_N} \{ J = \sum_{i=1}^{N} \theta(u_i) + \phi(x_i) \quad : \ u_i \geq 0, \ \ \forall i \} \tag{4}$$

subject to

$$x_i = \max(x_{i-1}, a_i) + u_i, \quad i = 1, \cdots, N. \tag{5}$$

The optimal solution of $\mathbf{P}$ is denoted by u_i^* for $i = 1, \cdots, N$ and the corresponding departure times in (5) are denoted by x_i^* for $i = 1, \cdots, N$.

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

Assumption 2 $\phi(\cdot)$ is continuously differentiable, strictly convex and its minimum is obtained at a finite point.

These assumptions are necessary to be consistent with the previous discussion regarding tradeoffs between the temporal and physical requirements in hybrid systems. We can interpret the processing time u_i as a measure of "quality" of the finished ith job, since, in general, the quality of a job decreases as the processing time devoted to it is shorter. On the other hand,

$\phi(\cdot)$ typically penalizes the departure time or tardiness of jobs relative to target "due dates". As an example, let $\theta(u_i) = 1/u_i$ and $\phi(x_i) = (x_i - \delta_i)^2$, where the two functions satisfy the above assumptions. Here, δ_i is the due date of the ith job.

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 jth 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 $Q(k,n)$:

$$Q(k,n): \min_{u_k, \cdots, u_n} J(k,n) = \sum_{i=k}^{n} \theta(u_i) + \phi(a_k + \sum_{j=k}^{i} u_j) \tag{6}$$

subject to

$$x_i = a_k + \sum_{j=k}^{i} u_j \geq a_{i+1}, \ i = k, \cdots, n-1. \tag{7}$$

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 [9]). 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 $a_k + \sum_{j=k}^{i} u_i = a_{i+1}$, 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 **P**, but it is limited to jobs $C_k, \cdots, C_n$ constrained through (7) 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; proofs are omitted, but may be found in [11].

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.

An implication of Lemma 1 is that if the busy period structure obtained with the optimal controls in problem **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:

$$x_n^*(k,n) < a_{n+1}. \tag{8}$$

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, whose proof is given in [11].

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

The uniqueness of the optimal solution of **P** under Assumption 1-2 was established in [6] using arguments relying on the optimality conditions formulated through generalized gradients. Theorem 1 above, however, provides simpler means for proving uniqueness, based on the following result whose proof is given in [11].

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.

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 **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 [6]). ■

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 **P** can be solved by Forward Algorithm I shown in Table 1. In **Step 2**, we solve the linearly constrained 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. $k = n+1$.

Table 1: Forward Algorithm I

```
Step 1 : (initialization) k = 1, n = 1, a_{N+1} = ∞;
while n ≤ 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 ← u*_j(k, n) for j = k, ···, n.
               k ← n + 1;
            endif
   Step 4 : (increment index n.)
            n ← n + 1;
end while
```

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 $N = 5$:

$$\min_{u_1, \cdots, u_5} J = \sum_{i=1}^{5} \{u_i^{-1} + x_i^2\} \tag{9}$$

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

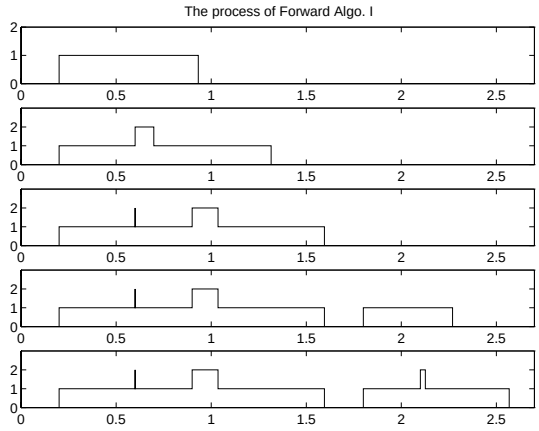

Figure 2: An illustration of the process of Forward Algorithm I.

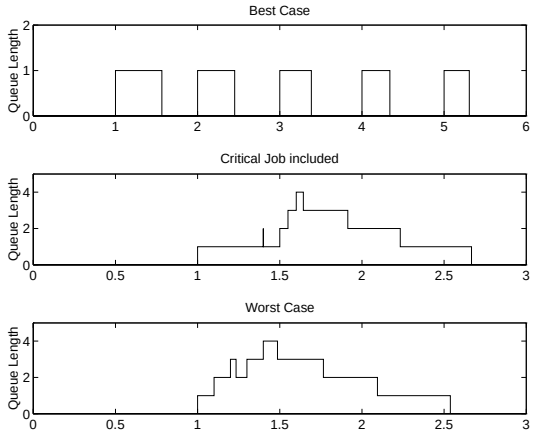

Figure 3: Some examples of Forward Algorithm I .

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. 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 $\theta(u_i) = \frac{1}{u_i}$ and $\phi(x_i) = x_i^2$. 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.

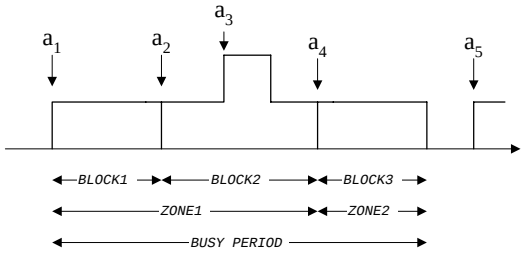

Figure 4: Three blocks and two zones in a busy period.

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 $k = 1$). The proof of the following result can be found in [11].

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. $x_k^*(1,n) = a_{k+1}$.

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

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.,

$$u_j^* = \begin{cases} u_j^*(1,k) & \text{for } j = 1, \cdots, k \\ u_j^*(k+1,n), & \text{for } j = k+1, \cdots, n \end{cases} \qquad (10)$$

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

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 '${x^*}_n(k, n) \geq a_{n+1}$' 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.

Some numerical examples for the Forward Algorithm II can be found in [11].

6 Conclusions

This paper has considered optimal control problems defined on a single-stage hybrid system motivated from manufacturing environments. 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.

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*, 43:31–45, Jan., 1998.

[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: Theorey and Applications*, 8:175–201, 1998.

[6] 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*, 2000.

[7] 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*, pages 450–455, Dec. 1999.

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

[9] 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*, 2000.

[10] L. Kleinrock, Editor. *Queuing Systems*, volume I. Wiley-Interscience, 1975.

[11] Young C. Cho, C. G. Cassandras, and D. L. Pepyne. Forward decomposition algorithms for optimal control of a class of hybrid systems. *submitted to Intl. J. of Robust and Nonlinear Control*, April, 2000.