approach, has been used to consider the problem of equivalence of a locally observable nonlinear system to a linear observer form by means of an output dependent time-scale transformation and a state-space diffeomorphism. The procedure is simple and requires only algebraic manipulations and differentiation to check for the existence of a suitable state-space diffeomorphism. The use of time-scaling transformations has been shown to be successful in dealing with certain nonlinearities encountered in biochemical and chemical reactor systems (see [16]). Although it was not shown explicitly in the note, the effect of an output-dependent time-scale transformation is very closely related to the effect of introducing an output diffeomorphism as exploited in [11]. Both transformations allow one to alleviate some nonlinearities that would prevent the solution of the observer linearization problem. However, the type of nonlinearity that can be cancelled is not the same in both cases. From this point of view, the introduction of a time-scale transformation does extend the class of observable nonlinear systems that are equivalent to the linear observer form.

ACKNOWLEDGMENT

The author would like to acknowledge the very insightful contributions of M. E. Valcher and the anonymous referees.

REFERENCES

[1] A. Banaszuk and W. M. Sluis, "On nonlinear observers with approximately linear error dynamics," in *Proc. Amer. Control Conf.*, Albuquerque, NM, 1997.
[2] D. Bestle and M. Zeitz, "Canonical from observer design for nonlinear time-variable systems," *Int. J. Control*, vol. 38, pp. 419–431, 1983.
[3] R. L. Bryant, S. S. Chern, R. B. Gardner, H. L. Goldschmidt, and P. A. Griffiths, *Exterior Differential Systems*. New York: Springer-Verlag, 1991.
[4] M. Guay, "An algorithm for orbital feedback linearization of single-input control-affine nonlinear systems," *Syst. Control Lett.*, vol. 38, pp. 271–281, 1999.
[5] M. Hou and A. C. Pugh, "Observer with linear error dynamics for nonlinear multi-output systems," *Syst. Control Lett.*, vol. 37, pp. 1–9, 1999.
[6] A. J. Krener and A. Isidori, "Linearization by output injection and nonlinear observers," *Syst. Control Lett.*, vol. 83, pp. 47–52, 1983.
[7] A. J. Krener and W. Respondek, "Nonlinear observers with linearizable error dynamics," *SIAM J. Control Optim.*, vol. 23, pp. 197–216, 1985.
[8] P. J. Olver, *Equivalence, Invariants, and Symmetry*. Cambridge, U.K.: Cambridge Univ. Press, 1995.
[9] W. Respondek, "Orbital feedback linearization of single-input nonlinear control systems," in *Proc. IFAC NOLCOS'98*, Enschede, The Netherlands, 1998, pp. 499–504.
[10] S. Sastry, *Nonlinear Systems: Analysis, Stability and Control*. New York: Springer-Verlag, 1999.
[11] K. Tchon and H. Nijmeijer, "On output linearization of observable dynamics," *Control Theory Adv. Technol.*, vol. 9, no. 4, pp. 819–857, 1993.
[12] D. Tilbury and S. Sastry, "The multi-steering N-trailer system: A case study of Goursat normal forms and prolongations," *Int. J. Robust Nonlinear Control*, vol. 5, no. 4, pp. 343–364, 1995.
[13] X.-H Xia and W.-B. Gao, "Non-linear observer design by observer canonical forms," *Int. J. Control*, vol. 47, pp. 1081–1100, 1988.
[14] ——, "Nonlinear observer design by observer error linearization," *SIAM J. Control Optim.*, vol. 27, pp. 199–216, 1985.
[15] F. Plestan and A. Glumineau, "Linearization by generalized input–output injection," *Syst. Control Lett.*, vol. 31, pp. 115–128, 1997.
[16] P. Moya, R. Ortega, M. S. Netto, L. Praly, and J. Pic, "Application of nonlinear time-scaling for robust controller design of reaction systems," *Int. J. Robust Nonlinear Control*, vol. 12, pp. 57–69, 2002.

An Improved Forward Algorithm for Optimal Control of a Class of Hybrid Systems

Ping Zhang and Christos G. Cassandras

***Abstract*—This note 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 a new property of the optimal state trajectory structure which holds under a modified condition on the cost function. This allows us to develop a low-complexity, scalable algorithm for explicitly determining the optimal controls, which can be more efficient than the best algorithm to date, known as the Forward Algorithm. A numerical example is included to illustrate the efficacy of the proposed algorithm, and to compare it with the Forward Algorithm.**

***Index Terms*—Hybrid systems, optimal control.**

I. 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 (DESs), such as timed automata, max-plus equations, or Petri nets (see [2]). 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 [3]–[5].

The hybrid system modeling framework considered in this note falls into the first category above. It is motivated by the structure of many manufacturing systems. In these systems, discrete entities (referred to as *jobs*) move through a network of work centers 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 [6] and [7]. Similar optimal control problems for hybrid systems have been studied using dynamic programming techniques in [8] and [9].

By the nature of the event-driven dynamics, the problem is inherently nonconvex and nondifferentiable. 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 [7], the task of solving these problems was simplified by exploiting structural properties of the optimal sample path. In particular, an optimal sample path

Manuscript received April 27, 2001; revised October 8, 2001. Recommended by Associate Editor R. S. Sreenivas. This work was supported in part by the National Science Foundation under Grants ACI-98-73339 and EEC-00-88073, by the Air Force Office of Scientific Research under Contract F49620-01-0056, by the Air Force Research Laboratory under Contract F30602-99-C-0057, and by EPRI/ARO under Contract WO8333-03.

The authors are with the Department of Manufacturing Engineering, Boston University, Brookline, MA 02446 USA (e-mail: pzhang@bu.edu; cgc@bu.edu).

Digital Object Identifier 10.1109/TAC.2002.803549.

0018-9286/02$17.00 © 2002 IEEE

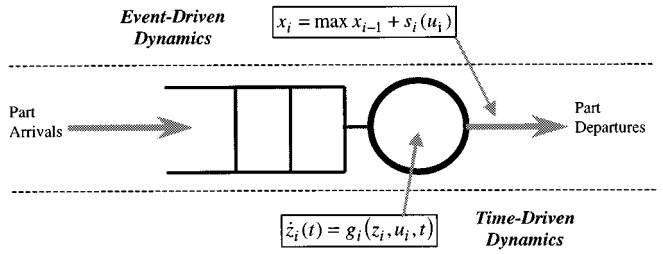

Fig. 1. A single-stage hybrid system.

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 has been realized using nonsmooth optimization methods [10], [7].

Recently, two algorithms were developed for solving such problems. They decompose the entire optimal control problem into a set of smaller convex optimization subproblems with linear constraints. The first is a "backward" recursive algorithm [11] proceeding backward in time from the last job to the first. 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 $2N-1$. The second is a "forward" algorithm whose complexity is simply N (again, in the number of convex constrained optimization problems to be solved) as shown in [1].

In this note, under a modified condition on the nature of the cost function, we further exploit the special structure of the optimal sample path in the singe-stage hybrid system framework. Based on the Forward Algorithm, an Improved Forward Algorithm is presented. Instead of increasing the number of jobs by one at every step, this algorithm may increase the number of jobs by more than one. This property can make it more efficient than the Forward Algorithm, as explained and illustrated in the remainder of this note.

II. 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 $0 \le a_1 \le \cdots \le a_N$. We denote these jobs by C_i, $i = 1, \ldots, N$. The jobs are processed first-come–first-served (FCFS) by a work-conserving and nonpreemptive 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, as in [1], varying only with each new job, and chosen to ensure that processing times are nonnegative, i.e., $s(u_i) \ge 0$.

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

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

Event-Driven Dynamics: The completion time of each job is denoted by x_i and is given by the standard Lindley equation for an FCFS nonpreemptive queue [12]

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

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

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

In a general setup, u_i is a control variable affecting the processing time through $s_i = s(u_i)$; see [6] for various examples and [13] for extensions to cases with time-varying controls $u_i(t)$ over a service time. 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 analysis in terms of the notation u_i. Hence, the optimal control problem, denoted by $\mathbf{P}$, has the following form:

$\mathbf{P}$:

$$\min_{u_1, \ldots, u_N} \left\{ J = \sum_{i=1}^{N} \{\theta_i(u_i) + \phi_i(x_i)\} : u_i \ge 0,\ i = 1, \ldots, N \right\} \tag{4}$$

subject to

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

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

Assumption 1: $\theta_i(u)$ 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: $\phi_i(\cdot)$ is continuously differentiable, strictly convex and its minimum is obtained at a finite point.

III. Properties of Optimal Solutions

We begin with the following definitions (see also [7]).

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: Considering a contiguous job subset $\{C_k, \ldots, C_n\}$, $1 \le k \le n \le N$ on the optimal sample path, the subset is said to be a *block* if

1) $x_{k-1} \le a_k$ and $x_n \le a_{n+1}$;
2) the subset contains no critical jobs.

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

1) $x_{k-1} < a_k$;
2) $x_n < a_{n+1}$;
3) $x_i \ge a_{i+1}$, for every $i = k, \ldots, n-1$.

Definition 4: A busy-period structure is a partition of the jobs $C_1, \ldots, C_n$ into busy periods.

The lth busy period consists of jobs $C_{k(l)}, \ldots, C_{n(l)}$ where $k(1) = 1$, $k(l) = n(l-1) + 1$ and $n(M) = N$, where M is the number of busy periods.

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

$Q(k, n)$:

$$\min_{u_k, \ldots, u_n} \left\{ \sum_{i=k}^{n} \left\{ \theta_i(u_i) + \phi_i\left(a_k + \sum_{j=k}^{i} u_j \right) \right\} : u_i \ge 0 \right\} \tag{6}$$

subject to

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

TABLE I
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
$u_j^* := u_j^*(k, n)$ for $j = k, \cdots, n$;
$k := n + 1$;
endif
Step 4 : (increment index n.)
$n := n + 1$;
end while

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 [11]). The solution of $Q(k, n)$ and the corresponding departure times are denoted by $u_j^*(k, n)$ and $x_j^*(k, n)$ for $j = k, \ldots, n$. Note that the equality $a_k + \sum_{j=k}^{i} u_j = a_{i+1}$, for some $i = k, \ldots, n-1$ is satisfied by the solution of $Q(k, n)$ 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 $C_k, \ldots, C_n$ constrained through (7) to all belong to the same busy period. There is one important property of $Q(k, n)$ that is captured in the following theorem (whose proof is given in [1]).

Theorem 3.1: Jobs $C_k, \ldots, C_n$ 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 $i = k, \ldots, n-1$;
3) $x_n^*(k, n) < a_{n+1}$.

Theorem 3.1 provides the basis for the Forward Algorithm presented in [1]. Given the arrival times of jobs $C_1, \ldots, C_N$, the optimal problem $\mathbf{P}$ can be solved by the Forward Algorithm shown in Table I. 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. Letting $k = 1$, $n = 1$, we first solve the linearly constrained convex optimization problem $Q(k, n)$ and obtain the control $u_j^*(k, n)$, $j = k, \ldots, n$ and departure times $x_j^*(k, n)$, $j = k, \ldots, n$. Then the structure of busy periods is identified by checking if $x_n^*(k, n) < a_{n+1}$. If $C_k, \ldots, C_n$ are identified as a single busy period, the control on the optimal path for $C_k, \ldots, C_n$ is given by $u_j^*(k, n)$, $j = k, \ldots, n$. Then, we set the index of the first job of a new busy period as $n + 1$, i.e., $k = n + 1$. We repeat the procedure over all jobs. This algorithm requires N steps, because n is increased by 1 at every step. Therefore, the Forward Algorithm must solve N subproblems to obtain the optimal solution.

IV. IMPROVED FORWARD ALGORITHM

In this section, we further exploit a property of the optimal state trajectory for the optimal control problem $\mathbf{P}$, which is based on modifying Assumption 2 as follows.

Assumption 3: $\phi_i(\cdot)$ is continuously differentiable and strictly convex. Moreover, $\phi_i(\cdot)$ for $i = 1, 2, \ldots, N-1$ is nondecreasing, and $\phi_N(\cdot)$ is monotone increasing.

This is in fact a common condition for a large class of problems, when one wishes, for example, to penalize the departure time of jobs (e.g., through x_i^2) or their system time [e.g., through $(x_i - a_i)^2$]. Thus, under Assumption 3, cost functions of the form $(x_i - \delta_i)^2$ are not appropriate (where δ_i represents a desired deadline for the ith job); instead, one focuses on a "tardiness" measure, such as $\max(x_i - \delta_i, 0)$, which is commonly used in order-driven manufacturing systems. To avoid the issue of nondifferentiability at the point δ_i in functions such as $\max(x_i - \delta_i, 0)$, one may use a variety of other functions that capture tardiness. A convenient class is that of Bezier functions and an example of a cost function that satisfies Assumption 3 is

$$\phi_i(x_i) = \begin{cases} 0, & x_i < \delta_i \\ \epsilon[a + b(x_i - \delta_i - \epsilon)]^2, & x_i \geq \delta_i \end{cases}$$

where ϵ is a positive number and a, b are appropriately selected real constants.

Theorem 4.1: If for the problem $Q(1, N)$, job C_i is not a critical job, i.e., $x_i^*(1, N) > a_{i+1}$, then for problem $\mathbf{P}$, job C_i cannot be the last job of any busy period, i.e., for any $k = 1, \ldots, N$, $n(k) \neq i$.

Proof: The statement of the theorem is equivalent to the following: If job C_i is the last job of some busy period on the optimal sample path, i.e., there exists some k such that $n(k) = i$, then $x_i^*(1, N) = a_{i+1}$, i.e., C_i is a critical job for the problem $Q(1, N)$.

Assume there are N jobs and M busy periods in the optimal sample path, and $n(0) = n_0 = 0$, $n(1) = n_1$, $n(n_1 + 1) = n_2, \ldots, n(n_{M-1} + 1) = n_M = N$. We have the following decoupling property of the optimal problem (see [7, Lemma 4.1]): for a busy period defined by $\{k(l), \ldots, n(l)\}$, and letting $i \in \{k(l), \ldots, n(l)\}$, the optimal control u_i^* depends only on $a_{k(l)}, \ldots, a_{n(l)}$ (it does not depend on the arrival times of jobs in any other busy period). By this decoupling property, the optimal controls for job $C_{n_{l-1}+1}, C_{n_{l-1}+2}, \ldots, C_{n_l}$, $l = 1, \ldots, M$, can be obtained by solving problem $Q(k, n)$ where $k = n_{l-1} + 1$ and $n = n_l$. The solution of this problem satisfies

$$\frac{\partial J}{\partial u_i} + \frac{\partial}{\partial u_i}\left[\sum_{i=n_{l-1}+1}^{n(l)-1} \lambda_i \left(a_{i+1} - a_{n_{l-1}+1} - \sum_{j=n_{l-1}+1}^{i} u_j\right)\right] = 0 \quad \text{for } i = n_{l-1} + 1, \ldots, n(l)$$

where $\lambda_i \geq 0$, $i = n_{l-1} + 1, \ldots, n(l) - 1$, are Lagrange multipliers adjoining the constraints (6) to (7). This gives

$$\frac{d\theta_i}{du_i} + \sum_{j=i}^{n_l} \frac{d\phi_j}{du_i} - \sum_{j=i}^{n_l} \lambda_j = 0, \qquad \text{for } i = n_{l-1} + 1, \ldots, n(l)$$

which implies that

$$\frac{d\theta_i}{du_i} + \sum_{j=i}^{n_l} \frac{d\phi_j}{du_i} \geq 0, \qquad \text{for } i = n_{l-1} + 1, \ldots, n(l). \tag{8}$$

Moreover, since C_{n_l} is the last job in a busy period, the solution u_i^*, $i = n_{l-1} + 1, \ldots, n(l)$, satisfies

$$a_{n_{l-1}+1} + \sum_{i=n_{l-1}+1}^{n_l} u_i^* < a_{n_l+1}. \tag{9}$$

Then, consider the relaxed optimization problem

$Q_r(k, n)$:

$$\min_{u_k, \ldots, u_n} \left\{\sum_{i=k}^{n} \left\{\theta_i(u_i) + \phi_i\left(a_k + \sum_{j=k}^{i} u_j\right)\right\} : u_i \geq 0\right\}. \tag{10}$$

The necessary condition satisfied by the solution, denoted by $u_{i_r}^*(1, N)$, of problem $Q_r(1, N)$ is

$$\frac{\partial J}{\partial u_i} = \frac{d\theta_i}{du_i} + \sum_{j=i}^{N} \frac{d\phi_j}{du_i} = 0, \qquad \text{for } i = 1, \ldots, N. \tag{11}$$

Using (8) and the monotonicity of $\phi_i(\cdot)$ for $i = n_{l-1}+1, \ldots, n_l$, i.e., i belonging to the lth busy period, where $l = 1, \ldots, M$

$$\begin{aligned}\frac{\partial J}{\partial u_i^*} &= \frac{d\theta_i}{du_i^*} + \sum_{j=i}^{N} \frac{d\phi_j}{du_i^*} \\ &= \frac{d\theta_i}{du_i^*} + \sum_{j=i}^{n_l} \frac{d\phi_j}{du_i^*} + \sum_{j=n_l+1}^{N} \frac{d\phi_j}{du_i^*} \\ &\geq \sum_{j=n_l+1}^{N} \frac{d\phi_j}{du_i^*} > 0 \end{aligned} \tag{12}$$

so, by (11) and (12), we have

$$u_{i_r}^*(1, N) < u_i^*, \qquad \text{for } i = n_{l-1}+1, \ldots, n(l). \tag{13}$$

Therefore, using (9)

$$\sum_{i=n_{l-1}+1}^{n_l} u_{i_r}^*(1, N) < \sum_{i=n_{l-1}+1}^{n_l} u_i^* < a_{n_l+1} - a_{n_{l-1}+1}. \tag{14}$$

By the convexity of the problem $Q(1, N)$, the optimal solution must be on the constraint boundary, otherwise, (14) implies that we can decrease the cost by decreasing the controls. Therefore, we have

$$\sum_{i=n_{l-1}+1}^{n_l} u_i^*(1, N) = a_{n_l+1} - a_{n_{l-1}+1} \tag{15}$$

i.e., $x_{n_l}^*(1, N) = a_{n_l+1}$, or C_{n_l} is a critical job in $Q(1, N)$. ■

This theorem suggests that we can get some information about the busy period structure on an optimal sample path from the solution $x_i^*(1, N)$ of $Q(1, N)$. In particular, only critical jobs of $Q(1, N)$ may end a busy period on an optimal path. Therefore, instead of adding only one job at every step of the Forward Algorithm involving the solution of a problem of the form $Q(k, n)$, we can add multiple jobs and make the algorithm more efficient by solving fewer problems of the form $Q(k, n)$. The price to pay is that we need to solve the larger problem $Q(1, N)$ first; however, some additional flexibility can be added to the algorithm as discussed in Remark 3. Based on Theorem 4.1, the Improved Forward Algorithm is shown in Table II, where Crit denotes the set of critical job indices and $\mathrm{Crit}(i)$ denotes the ith element of this set.

Remark 1: The differences between the Improved Forward Algorithm and the Forward Algorithm presented in [1] are

1) in the initialization step we need to solve problem $Q(1, N)$ and get information on all critical jobs;
2) in Step 4, we increase index n to the next critical job, instead of $n + 1$.

Remark 2: This algorithm requires at most $N + 1$ steps (usually, it requires less than N steps), while the Forward Algorithm always requires N steps. The worst case of the Forward Algorithm becomes the best case for this algorithm. If all N jobs are in a single busy period and there is no critical job, then the optimal control is immediately obtained when we solve the problem $Q(1, N)$.

Remark 3: When N is large, the solution of $Q(1, N)$ becomes computationally intensive. However, a simple variation of the algorithm, shown in Table III, provides some added flexibility. Specifically, it is not necessary to start with all N jobs in one busy period; instead, we may solve a problem $Q(1, N')$, where $1 \leq N' \leq N$, which still provides information on critical jobs leading to the optimal busy period structure. In this case, we decompose $\{1, \ldots, N\}$ into two or more subsequences of job indices and solve the associated problems. Thus, this approach provides the flexibility to control the number of convex problems to be solved. Since the actual computational time depends on both the number of problems $Q(k, n)$ solved and their individual sizes [i.e., the number of jobs $n - k + 1$ in $Q(k, n)$], we can take advantage of using N' to tradeoff the complexity of a small number of larger problems against a larger number of problems of lower dimensionality.

TABLE II
IMPROVED FORWARD ALGORITHM

Step 1 : (initialization)
  solve sub-optimal problem $Q(1, N)$;
  for $j = 1, \cdots, N$
    if $x_j^*(1, N) = a_{j+1}$, add j to set $Crit$;
  endfor
  add $N + 1$ to set $Crit$ in order to end the loop;
  $i := 1, k := 1, n := Crit(i), a_{N+1} := \infty$;
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
      $u_j^* := u_j^*(k, n)$ for $j = k, \cdots, n$;
      $k := n + 1$;
    endif
  Step 4 : (increment index n.)
    $i := i + 1$;
    $n := Crit(i)$;
end while

TABLE III
IMPROVED FORWARD ALGORITHM II

Step 1 : (initialization)
  $m := 1, a_{N+1} := \infty, d := 0$;
while $m < N$ do
  Step 2 : (initialization)
    $Crit := \emptyset$;
    solve sub-optimal problem $Q(m, m + N' + d - 1)$;
    for $j = m, \cdots, m + N' + d - 1$
      if $x_j^*(m, m + N' + d - 1) = a_{j+1}$, add j to set $Crit$;
    endfor
    add $m + N' + d$ to set $Crit$ in oder to end the loop;
    $i := 1, k := m, n := Crit(i)$;
  while $n \leq m + N' + d - 1$ do
    Step 3 : solve sub-optimal problem $Q(k, n)$;
    Step 4 : (identify single busy periods.)
      if $x_n^*(k, n) < a_{n+1}$ then
        $u_j^* := u_j^*(k, n)$ for $j = k, \cdots, n$;
        $k := n + 1$;
      endif
    Step 5 : (increment index n.)
      $i := i + 1$;
      $n := Crit(i)$;
  end while
  Step 6 : (increment index m or number d.)
    if $m < k$ then
      $m := k$;
      $d := 0$;
    else, this busy period has more than $N' + d$ jobs, increase d
      $d := d + 1$;
    endif
end while

V. EXAMPLES

Let us consider the following problem with $N = 10$:

$$\min_{u_1, \ldots, u_{10}} J = \sum_{i=1}^{10} \left\{ u_i^{-1} + x_i^2 \right\}$$

$$\text{subject to } x_i = \max(a_i, x_{i-1}) + u_i \tag{16}$$

for the arrival sequence $\{1, 1.4, 1.5, 1.55, 1.6, 3.0, 3.1, 3.2, 3.3, 3.5\}$.

First, let us set the maximum number of jobs considered in problem $Q(k, k+N'-1)$ equal to the total number of jobs N, i.e., we shall use the algorithm shown in Table II. At the first step, $Q(1, 10)$ is solved and we have a critical job set $\mathrm{Crit} = \{1, 5, 10\}$ (see also Fig. 2). Then, we start to verify the busy period structure. First, $Q(1, 1)$ for $k = 1$, $n = \mathrm{Crit}(1) = 1$ is solved. Since $x_1^*(1, 1) = 1.56 > a_2$, we set $n = \mathrm{Crit}(2) = 5$. Here, $Q(1, 2)$, $Q(1, 3)$ and $Q(1, 4)$ do not need to be considered. Then, $Q(1, 5)$ is solved. Since $x_5^*(1, 5) = 2.67 < a_6$, jobs $C_1, \ldots, C_5$ constitute a single busy period. The optimal controls for this busy period are obtained by solving $Q(1, 5)$. Then, we set $k = 6$ and $n = \mathrm{Crit}(3) = 10$ where k is the index of the first job of the next busy period. Finally, we obtain the optimal solution. Let $|Q(k, n)| = n-k+1$ denote the dimensionality of a convex problem. Then, for this example, as seen in Fig. 2, the cumulative dimensionality over all subproblems solved is $|Q(1, 10)| + |Q(1, 1)| + |Q(1, 5)| + |Q(6, 10)| = 10 + 1 + 5 + 5 = 21$, requiring a total of four steps, while the corresponding numbers using the Forward Algorithm are 30 with ten total steps.

Next, let us set the maximum number of jobs considered in problem $Q(k, k + N' - 1)$ equal to $5 < N$ and use the algorithm in Table III. This reduces the dimensionality of the initial problem, in case the computational complexity of $Q(1, 10)$ is substantial. Thus, at the first step $Q(1, 5)$ is solved and we have a critical job set $\mathrm{Crit} = \{1, 5\}$. Then, we need to solve $Q(1, 1)$ and get the optimal controls for this busy period by solving $Q(1, 5)$. Subsequently, we set $k = 6$ and $n = k + 5 = 10$. The cumulative dimensionality over all subproblems is now $|Q(1, 5)| + |Q(1, 1)| + |Q(6, 10)| = 5 + 1 + 5 = 11$, requiring three total steps, an improvement over the case where $Q(1, 10)$ was initially solved. In terms of actual CPU time, our implementation required 3.02 s for the original Improved Forward Algorithm [of which 1.86 s are spent in solving problem $Q(1, 10)$], while using the second version of the algorithm involving the initial solution of problem $Q(1, 5)$, the resulting CPU time was reduced to 0.92 s.

As previously illustrated, the proposed algorithm reduces the number of convex problems solved under the original Forward Algorithm. Its ultimate efficiency depends on the tradeoff between solving a single large problem of the form $Q(1, N)$ (required in the algorithm of Table II) and a collection of smaller problems (as in Table III). The ability to select some $N' \leq N$ in Table III provides a degree of flexibility for maximizing the benefit of the Improved Forward Algorithm.

VI. Conclusion

We have 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 and of time delivery requirements of the completed jobs. These optimal control problems are inherently neither convex nor differentiable because of the nature of the event-driven dynamics involved. However, the structure of optimal sample paths allows an efficient decomposition into busy periods which can be separately analyzed, leading to the efficient forward decomposition algorithm in [1]. In this note, we have identified a new property of the optimal sample paths, under a modified condition on the cost functions used, based on which we derived an efficient, low-complexity, scalable algorithm for computing optimal controls, which improves the algorithm presented in our earlier work.

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

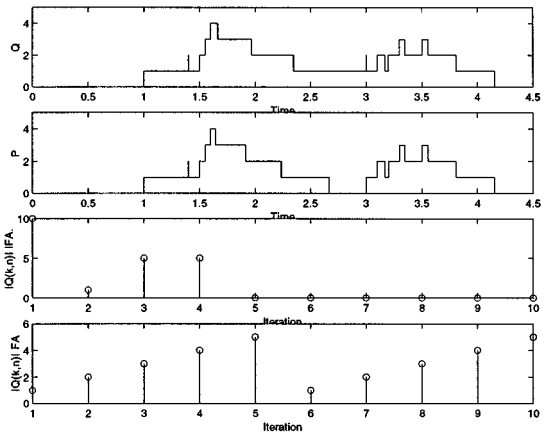

Fig. 2. The first two plots show the queue length of $Q(1, N)$ and $\mathbf{P}$. The second two present the complexity of $Q(k, n)$ using the Improved Forward Algorithm and the Forward Algorithm, respectively.

Acknowledgment

The authors would like to thank an anonymous referee for pointing out an omission in the proof of Theorem 4.1.

References

[1] Y. C. Cho, C. G. Cassandras, and D. L. Pepyne, "Forward decomposition algorithms for optimal control of a class of hybrid systems," *Int. J. Robust Nonlinear Control*, vol. 11, no. 5, pp. 497–513, 2001.

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

[3] R. Alur, T. A. Henzinger, and E. D. Sontag, Eds., *Hybrid Systems*. New York: Springer-Verlag, 1996.

[4] P. Antsaklis, W. Kohn, M. Lemmon, A. Nerode, and S. Sastry, Eds., *Hybrid Systems V*. New York: Springer-Verlag, 1998.

[5] 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, pp. 31–45, Jan. 1998.

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

[7] C. G. Cassandras, D. L. Pepyne, and Y. Wardi, "Optimal control of a class of hybrid systems," *IEEE Trans. Automat. Contr.*, vol. 46, pp. 398–415, Mar. 2001.

[8] S. Hedlund and A. Rantzer, "Optimal control of hybrid systems," in *Proc. 38th IEEE Conf. Decision and Control*, 1999, pp. 3972–3977.

[9] X. Xu and P. J. Antsaklis, "Optimal control of switched systems: New results and open problems," in *Proc. Amer. Control Conf.*, 2000, pp. 2683–2687.

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

[11] Y. Wardi, C. G. Cassandras, and D. L. Pepyne, "A backward algorithm for computing optimal controls for single-stage hybrid manufacturing systems," *Intl. J. Prod. Res.*, vol. 39, no. 2, pp. 369–394, 2001.

[12] L. Kleinrock, Ed., *Queuing Systems*. New York: Wiley-Interscience, 1975, vol. I.

[13] K. Gokbayrak and C. G. Cassandras, "A hierarchical decomposition method for optimal control of hybrid systems," in *Proc. 39th IEEE Conf. Decision and Control*, Dec. 2000, pp. 1816–1821.