Proceedings of the 13th
Mediterranean Conference on Control and Automation
Limassol, Cyprus, June 27-29, 2005

Optimal Control of Single-Stage Hybrid Systems with Poisson Arrivals and Deterministic Service Times

Kagan Gokbayrak and Muzaffer Misirci

***Abstract*— We tackle an optimal control problem for a single-stage hybrid system with Poisson arrivals and deterministic service times. In our setting, not only that the optimization problem is non-convex and non-differentiable, but also future arrival times are unknown at the times of decision. We propose a state-dependent service times policy where the state is defined as the system size. These service times are determined iteratively by a steepest descent algorithm whose derivative information is supplied by an infinitesimal perturbation analysis derivative estimator. We also propose an improved receding horizon controller with zero-length time window that utilizes the interarrival time distribution information available from the observed arrivals. Performances of these methods are compared to the optimal performance obtained from the Forward Decompositon Algorithm for which all future arrival times are known. It is also shown that the utilization of the observed interarrival time distribution information improves the performance of the receding horizon controller with zero-length time window.**

***Index Terms*— Hybrid Systems, Stochastic, Optimal Control, Receding Horizon Controller, Perturbation Analysis, Steepest Descent**

I. INTRODUCTION

The term "hybrid" is used to characterize systems that include time-driven and event-driven dynamics. The former are represented by differential (difference) equations, while the latter may be described through various frameworks used for Discrete Event Systems (DES), such as timed automata, max-plus equations, queueing networks, 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-driven models to include event-driven dynamics (for an overview, see [2], [3], [4], [5])

In this paper, we consider a single stage hybrid system framework falling into the first category above. It is motivated by the structure of many manufacturing systems. In this system, discrete entities referred to as jobs are moving through a workcenter which processes the jobs in order to change their physical characteristics. Each of these jobs has temporal and physical states. The physical state of the ith job, z_i evolves according to the time-driven dynamics

$$\dot{z}_i = u_i, \quad z_i(\tau_i) = \zeta_0 \text{ for } i = 1, 2, ... \tag{1}$$

and tracks the job quality measures such as temperature, weight, and etc... The temporal state x_i evolves according to the event-driven dynamics

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

where $x_0 = -\infty$, and tracks the departure time information. In (2), a_i is the arrival time of the ith job and s_i is the service time of the ith job dependent on the control input u_i applied on job i to bring it from the initial state $z_i(\tau_i) = \zeta_0$ to a desired final state $z_i(x_i) = \zeta_d$.

For the single stage framework above, the optimal control objective is to choose a control sequence $\{u_i : i = 1, ..., N\}$ for N jobs to minimize an objective function of the form

$$J = \sum_{i=1}^{N} \phi_i\left(z_i, u_i, s_i\right) + \psi_i(x_i) \tag{3}$$

subject to the time-driven dynamics in (1) and the event-driven dynamics in (2).

The term $\psi_i(x_i)$ is the cost associated with the system time of the ith job

$$\psi_i(x_i) = \alpha\left(x_i - a_i\right)^2 \tag{4}$$

and the term $\phi_i\left(z_i, u_i, s_i\right)$ is the quadratic operation cost for the ith job to bring its physical state z_i from ζ_0 to ζ_d

$$\phi_i\left(z_i, u_i, s_i\right) = \int_{\tau_i}^{\tau_i + s_i} \frac{1}{2} r u_i^2(t) dt \tag{5}$$

In order to tackle the optimization problem in (3), the system can be decomposed into two hierarchical levels; a lower level that represents physical processes characterized by time-driven dynamics and a higher level that represents events related to these physical processes (see in [6]).

It was shown in [6] that the lower level problem solution yields the operation cost

$$\theta_i\left(s_i\right) = \min_{u_i(t)} \phi_i(z_i, u_i, s_i) = \frac{1}{s_i} \frac{r(\zeta_d - \zeta_0)^2}{2} = \frac{\beta}{s_i} \tag{6}$$

where $\beta = \frac{r(\zeta_d - \zeta_0)^2}{2}$ and that the optimal control input $u_i^*(t)$ is

$$u_i^*(t) = \frac{\zeta_d - \zeta_0}{s_i} \tag{7}$$

K. Gokbayrak and M. Misirci are with the Department of Industrial Engineering, Bilkent University, Ankara, Turkey kgokbayr@bilkent.edu.tr, misirci@ug.bilkent.edu.tr

The higher level problem can then be written as

$$P: \min_{s_1,\ldots,s_N} J = \sum_{i=1}^{N} \theta_i(s_i) + \psi_i(x_i) = \sum_{i=1}^{N} \frac{\beta}{s_i} + \alpha(x_i - a_i)^2 \tag{8}$$

subject to event driven dynamics in (2). Note that longer processing times s_i result with lower operation costs and higher system time costs. This introduces a trade-off between the system time costs and the operation costs, and results with the challenging optimization problem in (8). Uniqueness of the optimal solution for this non-convex and non-differentiable cost function was established in [7]. In [8] forward decomposition algorithms are proposed for solving the optimization problem in (8) for which all arrival times are known. If only partial information on future arrivals, e.g. within a time window of length T, is available, receding horizon controllers in [9], [10], and [11] can be employed. These receding horizon controllers implement a forward decomposition algorithm assuming that the first arrival after the time window occurs at time $t = \infty$.

In this paper, we consider the case for which the future arrival times are unknown, i.e., the time window is of zero length. We propose two solution methods for the optimization problem in (8). The first method considers a state-dependent service times (SDST) policy, where the state is defined as the system size. Using *Perturbation Analysis* techniques (see in [1]), we obtain sensitivity information from the observed sample path, and utilize this sensitivity information in a steepest descent method to update our SDST policy. The second method considers a receding horizon controller with a time window of zero length and estimated arrivals (RHT0EA), and employs the receding horizon controller idea with a modified assumption: the arrivals after the time window are assumed to occur deterministically with a rate estimated from the observed sample path. The performance of these methods are compared against the optimal performance obtained from applying the forward decomposition algorithm (FWD) with all future arrival times are known (same as applying a receding horizon controller with $T = \infty$). This comparison gives us an upper bound on the value of future arrival information. Our methods are also compared against a receding horizon controller with a time window of zero length (RHT0) operating under the "no-arrival after time window" assumption.

II. SDST Method

Consider a single server queueing system with deterministic state-dependent service times and Poisson arrivals whose times are unknown. The system is FCFS (First Come First Served) with no preemption.

Let us denote the state-dependent service time for job i as $s_i^{k_i} \geq 0$ where k_i is the system size when job i enters service. In our policy, we want to have

$$s_i^{k_i} = s_j^{k_j} \text{ for } k_i = k_j \tag{9}$$

i.e., all jobs entering the service when the system size is k should be assigned the same service time s^k. Introducing this extra constraint, the problem becomes

$$\bar{P}: \min_{s_1^{k_1},\ldots,s_N^{k_N}} J = \sum_{i=1}^{N} \theta_i\left(s_i^{k_i}\right) + \psi_i(x_i) \tag{10}$$

subject to

$$x_i = \max(x_{i-1}, a_i) + s_i^{k_i} \quad x_0 = -\infty \tag{11}$$

$$s_i^{k_i} = s_j^{k_j} \quad \text{for } k_i = k_j \tag{12}$$

In order to solve for $\bar{P}$, we need to find s^k service times for $k = 1, 2, 3, ...$that depend on the values of α, β and the arrival rate λ. We employ a sensitivity based technique, the steepest descent algorithm for this task and the sensitivity estimation is presented next.

A. Sensitivity Estimation

An infinitesimal perturbation analysis (IPA) estimator (see in [1]) is employed to estimate the sensitivities of the cost to the service time perturbations. These perturbations are infinitesimal so that the busy period structures are not altered.

Let h_i^k be the number of times we have observed a system size of k up to (and including) job i in the current busy period. The observations are made at the service time decision points, when a new service starts. Idle periods reset the h vector to zero. Let $\delta^k > 0$ be the service time perturbations for s^k.

Let us analyze the effect of (positive) perturbations on the cost for a busy period. Since h vector is reset to zero at each idle period, we can decompose the sample path to busy periods and add the resulting busy period sensitivities to determine the overall sensitivity. Let us assume jobs $j+1$ to $j+m$ constitute a busy period. The cost perturbation for that busy period can be given as

$$\begin{aligned}
\delta J_{j+1,j+m} &= \sum_{i=j+1}^{j+m} [\frac{\beta}{s_i + \Delta s_i} - \frac{\beta}{s_i} \\
&\quad + \alpha(x_i + \Delta x_i - a_i)^2 - \alpha(x_i - a_i)^2] \\
&= \sum_{i=j+1}^{j+m} [-\frac{\beta \Delta s_i}{s_i^2 + s_i \Delta s_i} \\
&\quad + \alpha\left(2\Delta x_i(x_i - a_i) + (\Delta x_i)^2\right)]
\end{aligned}$$

Since the perturbations are infinitesimal we can approximate the cost perturbation as

$$\delta J_{j+1,j+m} \tilde{=} \sum_{i=j+1}^{j+m} \left[-\frac{\beta \Delta s_i}{s_i^2} + \alpha\left(2\Delta x_i(x_i - a_i)\right)\right]$$

Next, we need to write Δs_i and Δx_i in terms of δ^k

$$\Delta s_i = \delta^n$$

where n is the system size when the service for job i starts. The perturbation on the departure time, on the other hand, is

$$\Delta x_i = \sum_k h_i^k \delta^k$$

An algorithm to estimate the sample path sensitivity σ follows from the discussion above:

Step 1. Initialize $\sigma^k = 0$, $h_0^k = 0$ for all k

For each job $i = 1, ..., N$

Step 2. Check system size n at the beginning of the process

a. For all k, let $h_i^k = \begin{cases} h_{i-1}^k & k \neq n \\ h_{i-1}^k + 1 & k = n \end{cases}$

b. Update $\sigma^n = \sigma^n - \frac{\beta}{(s^n)^2}$

Step 3. Check system time T at the end of the process

a. For all k, update $\sigma^k = \sigma^k + 2\alpha T h_i^k$

b. If idleness is observed, set $h_i^k = 0$ for all k

Now that we can estimate the sensitivities, we can employ the steepest descent method to improve our decision variables, the state-dependent service times, which is presented next.

B. Steepest Descent Algorithm

In this algorithm, we initialize the state-dependent service times and update them at after every K jobs (or at the idle period following the Kth job).

In order to initialize and update the system with proper service times, let us determine the range of the optimal service times for the case where the constraint in (9) is removed. For this task, we first recall the "block" definition in [7]:

Definition 2.1: A block is a time interval $[a_k, x_m]$ defined by the subsequence of jobs $\{k, k+1, ..., m\}$ such that

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

2) $x_i > a_{i+1}$ for all $i = k, ..., m-1$

3) $x_m \leq a_{m+1}$

The following lemma presents the monotonicity property for the service times within a block.

Lemma 2.1: Consider the block consisting of jobs $\{k, ..., m\}$ with arrival times $\{a_i : i = k, ..., m\}$. The optimal service time for the ith job, s_i, in the block is increasing in i.

Proof: We need to determine the optimal service times s_i in

$$\min_{\substack{s_k,...,s_m \\ s_i>0}} J = \sum_{i=k}^{m} \frac{\beta}{s_i} + \alpha(x_i - a_i)^2$$

subject to

$$x_i = \max(x_{i-1}, a_i) + s_i \text{ for } i = k, ..., m$$

Since jobs $\{k, ..., m\}$ are in the same block we can write

$$x_i = a_k + \sum_{j=k}^{i} s_j$$

$$\begin{aligned} \min_{\substack{s_k,...,s_m \\ s_i>0}} J &= \sum_{i=k}^{m} \frac{\beta}{s_i} + \alpha(x_i - a_i)^2 \\ &= \sum_{i=k}^{m} \frac{\beta}{s_i} + \alpha(a_k + \sum_{j=k}^{i} s_j - a_i)^2 \end{aligned}$$

It is shown in [12] that $J(s)$ is continuously differentiable and strictly convex so that the unique optimal service times $s_i > 0$ should satisfy

$$\frac{\partial J}{\partial s_i} = 0 = -\frac{\beta}{s_i^2} + 2\alpha \sum_{j=i}^{m} \left(a_k - a_j + \sum_{n=k}^{j} s_n \right)$$

i.e., for $k \leq i < m$ we can write,

$$\begin{aligned} \frac{\beta}{s_i^2} &= 2\alpha \sum_{j=i}^{m} \left(a_k - a_j + \sum_{n=k}^{j} s_n \right) \\ &= 2\alpha \sum_{j=i+1}^{m} \left(a_k - a_j + \sum_{n=k}^{j} s_n \right) \\ &\quad +2\alpha \left(a_k - a_i + \sum_{n=k}^{i} s_n \right) \\ &= \frac{\beta}{s_{i+1}^2} + 2\alpha \left(a_k - a_i + \sum_{n=k}^{i} s_n \right) > \frac{\beta}{s_{i+1}^2} \end{aligned}$$

Hence for $k \leq i < m$, $s_{i+1} > s_i$.

Since we know that the optimal service times are increasing with the job index in a block, the following lemma should suffice to show that s^k should take values from the range $(0, \sqrt[3]{\frac{\beta}{2\alpha}})$.

Lemma 2.2: The optimal service time of the last job in a block $s_m \leq \sqrt[3]{\frac{\beta}{2\alpha}}$.

Proof: Let $t = \max(x_{m-1}, a_m)$ be the optimal time the service starts for the last job m in the block.

The cost associated with that job is:

$$J_m = \frac{\beta}{s_m} + \alpha(t + s_m - a_m)^2$$

In order to determine the minimizer we need to solve for

$$\begin{aligned} \frac{\partial J_m}{\partial s_m} &= -\frac{\beta}{s_m^2} + 2\alpha(t + s_m - a_m) = 0 \\ &\Rightarrow \quad \beta = 2\alpha (s_m)^3 + 2\alpha(t - a_m) (s_m)^2 \geq 2\alpha(s_m)^3 \\ &\Rightarrow \quad \sqrt[3]{\frac{\beta}{2\alpha}} \geq s_m \end{aligned}$$

Since the service time of last job of any block is upper bounded by $\sqrt[3]{\frac{\beta}{2\alpha}}$, the optimal service times in a sample path take values from the $(0, \sqrt[3]{\frac{\beta}{2\alpha}})$ interval.

Hence, the steepest descent method for updating the state-dependent service times is as follows:

Step 1. For all k, initialize s_0^k with values from the interval $(0, \sqrt[3]{\frac{\beta}{2\alpha}})$ decreasing with k.

For each iteration $n = 1, 2, ...$

Step 2. Observe the system for K job departures.

Step 3. Estimate sensitivities σ_n^k by running the sensitivity estimation algorithm above.

Step 4. Update $s_n^k = s_{n-1}^k - \eta_n \sigma_n^k$

Here η_n is the stepsize for the nth iteration satisfying the standard assumptions for convergence in, e. g., [13] and [14]:

(A1) $\sum_{n=1}^{\infty} \eta_n = \infty$

(A2) $\sum_{n=1}^{\infty} \eta_n^2 < \infty$

Before we proceed with our RHT0EA method, let us briefly describe two methods in literature for solving the higher level problem in (8); the forward decomposition (FWD) method, for the case where all future arrival times are known, and a receding horizon controller with zero-length time window (RHT0), for the case where no future arrival times are known. RHT0EA method is similar to the RHT0 method except for that it assumes an infinite sequence of deterministic future arrivals with the observed arrival rate.

III. FWD METHOD

We describe below the forward decomposition algorithm referred to as 'Forward II' in [8]. This algorithm requires us to possess all future arrival time information $\{a_i : i = 1, ..., N\}$ and yields the optimal solution off-line.

We begin with the following definitions:

Definition 3.1: A busy period is a time interval $[a_k, x_n]$ defined by the subsequence of jobs $\{k, k+1, ..., n\}$, such that the following conditions are satisfied:

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

2) $x_n < a_{n+1}$

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

Definition 3.2: A busy period structure is a partition of the jobs $\{1, ..., N\}$ into busy periods.

The busy period structure is represented by $\{B_1, ..., B_M\}$ for some positive integer $M \leq N$. The jth busy period consists of jobs $\{k(j), ..., n(j)\}$ where $k(1) = 1$, $k(j) = n(j-1) + 1$ and $n(M) = N$.

Consider the following optimization problem for the busy period consisting of jobs $\{k, ..., n\}$:

$$Q(k,n) = \min_{s_k,...,s_n} \sum_{i=k}^{n} \theta_i(s_i) + \psi_i(x_i) \qquad (13)$$

subject to

$$\begin{aligned} x_i &= a_k + \sum_{j=k}^{i} s_j, \quad i = k, ..., n \\ x_i &\geq a_{i+1}, \qquad i = k, ..., n-1 \\ s_i &\geq 0, \qquad i = k, ..., n \end{aligned}$$

Note that since the functional is continuously differentiable and strictly convex, the problem (13) is a convex optimization problem with linear constraints and has a unique solution at a finite point.

Now, consider that we know the busy period structure of the optimal sample path, which is shown to be unique in [8], and denote it by $\{B_1^*, ..., B_M^*\}$. Then, the solution to the optimization problem in (8) can be written as

$$\sum_{j=1}^{M} Q(k(j), n(j))$$

which is also unique.

The forward decomposition algorithm is developed to determine the *unique* optimal busy period structure for a given system. In the meantime, it also determines the optimal controls and the optimal cost. The algorithm consists of the following steps:

Step 1. (Initialization) Set $k = 1$, $n = 1$, $a_{N+1} = \infty$

while $n \leq N$ do

Step 2. Determine $Q(k, n)$

Step 3. If $x_n(k, n) \leq a_{n+1}$ then

a. Let $s_i = s_i(k, n)$ for $i = k, ..., n$

b. Update $k = n + 1$

Step 4. Increment $n = n + 1$

This algorithm decomposes (8) into smaller convex optimization problems. The size and the complexity of the subproblems depend on the given system.

IV. RHT0 METHOD

In this special case of the receding horizon controller method (presented in [9], [10], and [11]), we do not have any future arrival time information, however we have the arrival times of jobs that are currently in the system, and the time of the last departure from the system. We assume that the first future arrival will occur at time $t = \infty$, i.e., an idle period will follow once the jobs in the system are processed.

If jobs $\{k, ..., n\}$ are currently in the system at time $t = \max(a_k, x_{k-1})$, when the service starts for job k, in order to determine the service time s_k we need to solve the following optimization problem:

$$\bar{Q}(k,n) = \min_{s_k,...,s_n} \sum_{i=k}^{n} \theta_i(s_i) + \psi_i(x_i) \qquad (14)$$

subject to

$$\begin{aligned} x_i &= t + \sum_{j=k}^{i} s_j, \quad i = k, ..., n \\ s_i &\geq 0, \qquad i = k, ..., n \end{aligned}$$

Note that job k is not necessarily the first job in its busy period. After job k departs, the process is repeated to determine the service time s_{k+1}.

V. RHT0EA METHOD

Modifying the assumption on future arrivals for the RHT0 method, we assume that an infinite sequence of future arrivals will be observed with a constant interarrival time. The interarrival time can be estimated from the observed

sample path; if n jobs have arrived up to time t then future arrivals are expected to occur at

$$\hat{a}_i = t + \frac{t}{n}(i-n)$$

for $i = n+1, n+2, ...$ Note that $\hat{a}_i$ values for future arrivals may change at each decision time.

Let us assume that jobs $\{k, ..., n\}$ are currently in the system at time $t = \max(a_k, x_{k-1})$, when the service starts for job k. In order to determine the service time s_k for some $m \geq n$, we need to solve the following optimization problem:

$$\tilde{Q}(k,m) = \min_{s_k,...,s_m} \sum_{i=k}^{m} \theta_i(s_i) + \psi_i(x_i)$$

subject to

$$\begin{aligned} x_i &= t + \sum_{j=k}^{i} s_j, \quad i = k, ..., m \\ x_i &\geq \hat{a}_{i+1}, \quad i = n, ..., m-1 \\ s_i &\geq 0, \qquad\qquad i = k, ..., m \end{aligned}$$

Note that $x_i \geq \hat{a}_{i+1}$ constraints exist only for $m > n$. For $m = n$, the optimization problem reduces to the formulation in (14). Hence, the algorithm to determine the service time s_k at time t is as follows:

Step 1. Set $m = n$, $\hat{a}_{m+1} = t + \frac{t}{n}$, $done$ = False
Do
Step 2. Determine $\tilde{Q}(k,m)$
Step 3. If $x_i(k,m) \leq \hat{a}_{i+1}$ for some $i = n, ..., m$ then
a. Set $s_k = s_k(k,m)$
b. Set $done$ = True
Step 4. Increment $m = m + 1$
Step 5. Set $\hat{a}_{m+1} = \hat{a}_m + \frac{t}{n}$
While not $done$

VI. NUMERICAL EXAMPLES

In order to evaluate the performances of our SDST and RHT0EA methods, we compare them to the RHT0 method. FWD method results are given as lower bounds for the costs. The "optimal" state-dependent service times, s^i are determined by applying the steepest descent algorithm on the same arrival sequence repeatedly using the updated s vector from the current iteration as the initial vector for the next iteration until convergence is observed.

Example 1. Consider the problem P in (8) with $\alpha = 2$ and $\beta = 1$. The arrival process is Poisson with rate λ. As λ is varied, some of the optimal state-dependent service times are given in Table 1a and the resulting costs for 10000 jobs can be estimated as in Table 1b.

λ	s^1	s^2	s^3	s^4	s^5
2.0	0.374	0.266	0.220	0.195	0.188
3.0	0.310	0.231	0.199	0.175	0.162
5.0	0.233	0.178	0.155	0.144	0.132

Table 1a: Optimal state-dependent service times under various arrival rates

λ	SDST	RHT0EA	RHT0	FWD
2.0	37846	38770	42554	36679
3.0	45920	46852	54264	41988
5.0	63190	65449	79220	56460

Table 1b: Cost estimates under various arrival rates

Note that both the SDST and the RHT0EA methods outperformed the RHT0 method for all arrival rates, and approached within 16% of the optimal performance obtained from applying the FWD method. The cost advantages over the RHT0 method increase as the arrival rate increases. One explanation may be that the RHT0 method assumes an idle period after processing the jobs currently in the system, however; as the arrival rate increases this assumption is violated more often. The SDST and the RHT0EA methods, on the other hand, decreases the service times as the rate increases with the expectancy of more arrivals.

Example 2. Consider the same problem P above, and assume that the job arrival rate is $\lambda = 2.0$. Under various (α, β) pairs, some of the optimal state dependent service times are given in Table 2a, and the resulting costs for 10000 jobs can be estimated as in Table 2b. Both the SDST and the RHT0EA methods performed within 11% of the optimal and at least 9% better than the RHT0 method under different (α, β) pairs.

α	β	s^1	s^2	s^3	s^4	s^5
2	1	0.374	0.266	0.220	0.195	0.188
1	1	0.426	0.313	0.263	0.231	0.210
1	2	0.473	0.357	0.309	0.272	0.255

Table 2a: Optimal state-dependent service times under various (α, β) pairs.

α	β	SDST	RHT0EA	RHT0	FWD
2	1	37846	38770	42554	36679
1	1	33322	33863	38655	30476
1	2	59769	60817	71051	54628

Table 2b: Cost estimates under various (α, β) pairs.

Example 3. Consider the same problem P above with $\lambda = 2.0$, $\alpha = 2$, and $\beta = 1$ for a sample path of 10000 Poisson arrivals. The average service times calculated for the FWD, the RHT0EA and the RHT0 methods for each system size are compared to the SDST service times in Table 3.

Method	s^1	s^2	s^3	s^4	s^5
SDST	0.374	0.266	0.220	0.195	0.188
RHT0EA	0.416	0.292	0.237	0.208	0.189
FWD	0.433	0.310	0.258	0.223	0.190
RHT0	0.557	0.353	0.281	0.239	0.210

Table 3: Average service times for alternative methods

Since the RHT0 method assumes an idle period after the last job in the system, it overestimates the optimal service times, and results with longer busy periods. The RHT0EA and the SDST methods tend to give smaller service times then the optimal service times obtained from the FWD method.

Example 4. Consider the real-time system where 10000 Poisson arrivals with rate $\lambda = 2.0$ are observed and we update the service times after every 500 jobs. We assume $\alpha = 2$ and $\beta = 1$. The steepest descent algorithm updates the service times as in Figure 1 and a total cost of 39455 results from this sample path. Note that this total cost depends on the initial service time estimates and in this example it turns out to be lower than the RHT0 cost of 42554 and higher than the RHT0EA cost of 38770.

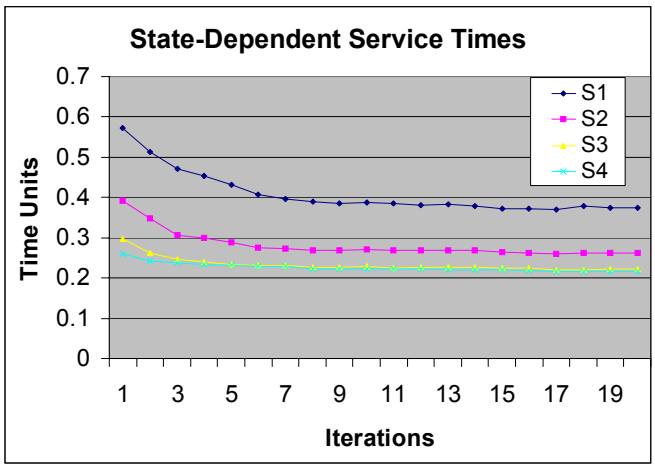

Figure 1. Service time trajectories resulting from the steepest descent method

VII. CONCLUSIONS AND FUTURE WORK

We considered a single stage hybrid system with a certain cost structure, Poisson arrivals and deterministic service times. For the case where future arrival times are unknown, we proposed a state-dependent service time policy and an IPA based sensitivity estimator to use with a steepest descent algorithm. Even though this policy is simpler to implement, as it does not have to solve (14), it outperformed the RHT0 method, a special case of the Receding Horizon method in [9], [10] and [11] with a time window of zero-length. This superiority was expected as the RHT0 method assumes the first future arrival to occur at time $t = \infty$, so that an idle period follows the departure of the last job currently in the system. The SDST method, on the other hand, learns some form of interarrival time distribution information from the observed arrivals. A similar idea is implemented in the RHT0EA method we proposed to improve the RHT0 method, where the controller learns the average arrival rate from the observed arrivals and assumes an infinite sequence of deterministic future arrivals with the same rate. The RHT0EA also outperformed the RHT0 and performed comparably to the SDST method.

Extending the "estimated arrivals" idea to receding horizon controllers with small window sizes is the topic of ongoing research. For large window sizes, the receding horizon controllers perform comparable to the FWD method, and the estimated arrivals idea can improve them for small window sizes.

REFERENCES

[1] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, 1999.

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

[3] P. Antsaklis, W. Kohn, M. Lemmon, A. Nerode, and S. Sastry, eds., *Hybrid Systems*. 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 Tr. on Automatic Control*, vol. 43, no. 1, pp. 31–45, 1998.

[5] R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, eds., *Hybrid Systems - Vol. 736 of Lecture Notes in Computer Science*. Springer-Verlag, 1993.

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

[7] C. G. Cassandras, D. L. Pepyne, and Y. Wardi, "Optimal control of a class of hybrid systems," *IEEE Trans. on Automatic Control*, vol. AC-46, 3, pp. 398–415, 2001.

[8] Y. C. Cho, C. G. Cassandras, and D. L. Pepyne, "Forward decomposition algorithms for optimal control of a class of hybrid systems," *Intl. J. of Robust and Nonlinear Control*, vol. 11, pp. 497–513, 2001.

[9] C. G. Cassandras and R. Mookherjee, "Receding horizon control for a class of hybrid systems with event uncertainties," *Proc. of 2003 American Control Conf*, pp. 5197–5202, 2003.

[10] C. G. Cassandras and R. Mookherjee, "Properties of receding horizon controllers for some hybrid systems with event uncertainties," *Proc. 2003 IFAC Conf. on Analysis and Design of Hybrid Systems*, pp. 413–418, 2003.

[11] C. G. Cassandras and R. Mookherjee, "Receding horizon optimal control for some stochastic hybrid systems," *Proc. of 42nd IEEE Conf. Decision and Control*, pp. 2162–2167, 2003.

[12] Y. Wardi, C. G. Cassandras, and D. L. Pepyne, "A backward algorithm for computing optimal controls for single-stage hybrid manufacturing systems," *Int. J. Prod. Res.*, vol. 39-2, pp. 369–393, 2001.

[13] H. Kushner and G. Yin, *Stochastic Approximation Algorithms and Applications*. Springer-Verlag New York, Inc., 1997.

[14] E. K. P. Chong and P. J. Ramadge, "Convergence of recursive optimization algorithms using ipa derivative estimates," *Journal of Discrete Event Dynamic Systems: Theory and Applications*, vol. 1, pp. 339–372, 1992.