LQR Optimization of Linear System Switching

Bo Lincoln and Bo Bernhardsson

***Abstract*—This note considers offline optimization of a switching sequence for a given finite set of linear control systems, together with joint optimization of control laws. A linear quadratic full information criterion is optimized and dynamic programming is used to find an optimal switching sequence and control law. The main result is a method for efficient pruning of the search tree to avoid combinatoric explosion. A method to prove optimality of a found candidate switch sequence and corresponding control laws is presented.**

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

I. Introduction and Motivation

Optimal hybrid control problems arise in many applications, see, e.g., [1] and [2]. An interesting subclass of hybrid systems consists of piecewise linear systems where either controlled or uncontrolled switches between linear systems are used; see [3]. The main question in most optimal hybrid control problem formulations is how to avoid the combinatorical explosion associated with exploring all possible switching alternatives. The problem area combines the traditionally separate research areas of search over graphs and control theory. In most papers on optimal hybrid control, this issue is generally not dealt with. The problem is sometimes solved by exhaustive search. In this note, we propose and evaluate a promising pruning method for efficient tree search, obtained by using information about the search objective.

The motivation for our work has mainly come from real-time control systems, where there often are restrictions on common resources such as communication bandwidth or CPU power. The different control loops have to share some media. This is often done by time-division-multiplexing, i.e., using some time slots for one loop and some other for another loop. One example where this problem is found is control over a wireless network environment such as Bluetooth [4]. The data packets are long and the maximum sample rate is restricted. In Bluetooth, only one network device can be accessed every 1.25 ms, so the controller has to choose which device to control (or sample); see Fig. 1.

The scheduling, i.e., the choice of control and measurement sequences, is normally optimized offline. The possibility to use online information in the scheduling algorithms, such as local information about signal values, has also been suggested recently, see [5]. Such online scheduling will not be studied here.

Offline scheduling of linear control systems under quadratic criteria has been treated recently in [6]–[8], where a separation property between control and estimation is presented. These references, however, do not present any efficient solving methods and lead to search problems over large trees. When the control horizon increases the size of the trees grows exponentially. The purpose of the present note is to present a pruning method which often decreases this complexity drastically.

Manuscript received April 14, 2000; revised July 2, 2001 and October 31, 2001. Recommended by Associate Editor R. S. Sreenivas.

The authors are with the Department of Automatic Control, Lund Institute of Technology, SE-22100 Lund, Sweden (e-mail: lincoln@control.lth.se; bob@control.lth.se).

Digital Object Identifier 10.1109/TAC.2002.803539.

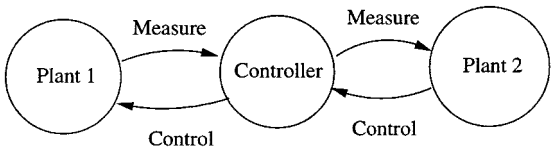

Fig. 1. A simple problem. The controller can only access one plant each time slot. Which sequence gives the best expected cost?

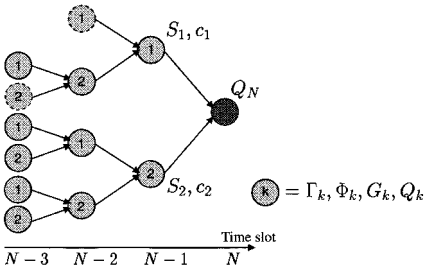

Fig. 2. The control sequence tree for $M = 2$ when expanding all possibilities from $N - 3$ to N (except for two nodes, dashed, where the tree was pruned).

II. Problem Formulation

The problems we are interested in can be formulated as finding the best switching sequence of system matrices for a discrete-time linear system

$$z(n+1) = \Phi(n)z(n) + \Gamma(n)u(n) + G(n)v(n). \tag{1}$$

Here, z is the (extended) state space vector, u the control signals, and v standard stochastic independent disturbances with zero mean and unit covariance. The system matrices $\Phi(n)$, $\Gamma(n)$, and $G(n)$ are chosen by the controller in each step from a small set of M alternative systems $\{(\Phi_k, \Gamma_k, G_k, Q_k)\}$, $y \in \{1, \ldots, M\}$ (where Q_k has to do with the cost of the system). Note that the system is time varying *only* since the controller can choose system matrices from a set at every control instant. The set of possible matrices does *not* change over time, so the *problem* is time invariant.

The problem is to find a linear feedback law $u(n) = -L(n)z(n)$ and a sequence $K(0, N) = [k(0), k(1), \ldots, k(N-1)]$ corresponding to choosing $\Phi(n) = \Phi_{k(n)}$, $\Gamma(n) = \Gamma_{k(n)}$, $G(n) = G_{k(n)}$, and $Q(n) = Q_{k(n)}$ that minimize the cost

$$V(P_z(0), 0, N, L(\cdot), K(0, N)) = \mathop{\mathbf{E}}_{v}\left\{\sum_{n=0}^{N-1}\begin{bmatrix} z(n) \\ u(n)\end{bmatrix}^T Q(n)\begin{bmatrix} z(n) \\ u(n)\end{bmatrix} + z(N)^T Q_N z(N)\right\} \tag{2}$$

where $\mathbf{E}\{z(0)\} = 0$, $\mathbf{E}\{z(0)z(0)^T\} = P_z(0)$, $Q(n) \geq 0$ and $Q_N \geq 0$.

III. Finding an Optimal Sequence

We will find an optimal scheduling sequence and control law by doing backward recursion of the cost (dynamic programming), evaluating all possible choices of $K(n, N)$. See Fig. 2 for an illustration. If this is done without care, the tree will of course grow exponentially. Therefore, we present a pruning algorithm which aims at keeping the tree size down to a reasonable level. The whole optimization of the sequence is done offline, so no feedback information is used in the scheduling.

We will use the notation "optimal sequence" for a sequence which achieves the optimal cost. There may be more than one sequence which does this, and we will aim at finding at least one.

Notation: Throughout the rest of this note, the cost function V will be written as

$$V(\text{start, end, sequence})$$

where *start* and *end* denote the first and last time-step of the cost. *Sequence* is either a sequence of choices [such as e.g., $K(0,\ N)$] from the start to the end time-step, a set of such sequences, or omitted. If a set, V denotes the minimum over all sequences in the set, and if omitted V denotes the minimum over all possible sequences. The initial variance $P_z(0)$ is omitted for notational convenience.

A. Cost Representation and Feedback Gain

For a fixed choice of $K(0,\ N)$, the problem is a standard time-varying linear-quadratic control problem. Under standard assumptions, it is well known that the best achievable cost can be written as

$$V(0,\ N,\ K(0,\ N)) = z(0)^T S_{K(0,\ N)} z(0) + c_{K(0,\ N)} \tag{3}$$

where $S_{K(0,\ N)}$ is a positive symmetric matrix and $c_{K(0,\ N)}$ is a constant term due to the noise. The optimal feedback law is

$$u(n) = -L_{K(n,\ N)} z(n) = {F^{uu}_{K(n,\ N)}}^{-1} {F^{zu}_{K(n,\ N)}}^T z(n) \tag{4}$$

where

$$\begin{aligned} F_{K(n,\ N)} &= \left[\begin{array}{c|c} F^{zz}_{K(n,\ N)} & F^{zu}_{K(n,\ N)} \\ \hline F^{uz}_{K(n,\ N)} & F^{uu}_{K(n,\ N)} \end{array}\right] \\ &= Q + [\,\Phi_k \quad \Gamma_k\,]^T S_{K(n+1,\ N)} [\,\Phi_k \quad \Gamma_k\,]. \end{aligned} \tag{5}$$

As the minimization of L is straightforward, it will be left out in the remaining sections.

B. Finding a Candidate Sequence

Finding a candidate sequence is, as mentioned before, done by backward iteration. One "step" in the iteration means expanding the set of candidate sequences one step in time. This is done by first expanding the search tree with new possible sequence choices (see Fig. 2), and then pruning (removing branches) using the algorithm described below. The set $\kappa_{\text{cand}}(n+1,\ N)$ of possible control sequences from time $n+1$ to N is expanded by

$$\kappa_{\text{expand}}(n,\ N) = \{1,\ 2,\ \ldots,\ M\} \times \kappa_{\text{cand}}(n+1,\ N). \tag{6}$$

Let $K_{\text{prune}} \in \kappa_{\text{expand}}(n,\ N)$ be a sequence in the set. The corresponding cost is parameterized by $S_{\text{cand}} = S_{K_{\text{prune}}}(n,\ N)$ and $c_{\text{cand}} = c_{K_{\text{prune}}}(n,\ N)$. The idea of the algorithm is to remove K_{prune} if another sequence $K_{\text{cand}} \in \kappa_{\text{expand}}(n,\ N)$ will perform better for all states. The algorithm calculates an $\alpha \in [0,\ 1]$ showing how close the quadratic cost S_{prune} is to be worse than S_{cand}, with $\alpha = 1$ meaning that K_{prune} is worse for all states. If α is close enough to one, and the noise cost c_{prune} is sufficiently much larger than c_{cand}, then K_{prune} is removed (tested by the inequality in step 3 of the algorithm in Table I).

"Sufficiently much larger" is here represented by one parameter, $R > 0$, which must be chosen by hand. The exact meaning of R is shown in Theorem 1.

The algorithm creates a set $M(n,\ N)$, called *motivation set*, which contains data on every pruned sequence and on the candidate sequences which was judged better. This data is used in Section III-C to prove optimality of the found sequence. The algorithm is described in Table I.

TABLE I
THE BRANCH-AND-BOUND ALGORITHM

1. Start with $\kappa_{\text{prune}}(n,N)$, $\kappa_{\text{cand}}(n,N)$, and $M(n,N)$ being empty sets and calculate $c_{\min} = \min_{K\in\kappa_{\text{expand}}(n,N)} c_K$.
2. If $\kappa_{\text{expand}}(n,N)$ is empty then quit else choose $K_{\text{prune}} \in \kappa_{\text{expand}}(n,N)$ for possible pruning.
3. If $\exists K_{\text{cand}} \in \kappa_{\text{expand}}(n,N) \cup \kappa_{\text{cand}}(n,N)$ such that for
$$\alpha = \max\{\lambda \le 1 \mid (S_{K_{\text{prune}}} - Q_N) \ge \lambda(S_{K_{\text{cand}}} - Q_N)\}$$
it holds that
$$c_{\text{prune}} - c_{\text{cand}} - (1-\alpha)(c_{\min} + R - c_{\text{cand}}) \ge 0$$
then K_{prune} is pruned, i.e.
   - Move K_{prune} from the set $\kappa_{\text{expand}}(n,N)$ to the set $\kappa_{\text{prune}}(n,N)$.
   - Move K_{cand} from the set $\kappa_{\text{expand}}(n,N)$ to the set $\kappa_{\text{cand}}(n,N)$.
   - Let $M(n,N)=M(n,N)\cup\{(c_{\text{prune}},\alpha,c_{\text{cand}})\}$.
   - Go to step 2.
4. Else move K_{prune} from the set $\kappa_{\text{expand}}(n,N)$ to $\kappa_{\text{cand}}(n,N)$. Go to step 2.

By using this tree pruning, the number of sequences in $\kappa_{\text{cand}}(n,\ N)$ can be kept reasonably low when recursing backward if R is chosen small enough. After N iteration stages, the final proposed sequence is

$$K_{\text{cand}}(0,\ N) = \operatorname*{argmin}_{K\in\kappa_{\text{cand}}(0,\ N)} (\mathbf{tr}(P_z(0)S_K) + c_K) \tag{7}$$

with cost $V_{\text{cand}} = V(0,\ N,\ K_{\text{cand}}(0,\ N))$.

C. Optimality of the Candidate Sequence

The candidate sequence found by the algorithm above may or may not be optimal, depending on the choice of R. A method will now be presented which can prove optimality of the proposed sequence if R is large enough. The idea of the proof is to show that a lower bound on the obtainable cost by using one of the pruned sequences is still higher than the cost of the found sequence. If this holds for every pruned sequence, the found candidate is optimal. Throughout the rest of the section, $M(n,\ N)$ is the motivation set from the tree pruning, and consists of

$$M(n,\ N) = \{(c_{\text{prune}},\ \alpha,\ c_{\text{cand}})_1,\ (c_{\text{prune}},\ \alpha,\ c_{\text{cand}})_2,\ \ldots\}.$$

We will also use $V_{\text{lowb}}(0,\ N,\ \kappa_{\text{cand}}(n,\ N))$, which fulfills

$$\begin{aligned} \min_{\substack{K(0,\ n)\\ K(n,\ N)\in\kappa_{\text{cand}}(n,\ N)}} & V(0,\ N,\ [K(0,\ n)\ K(n,\ N)]) \\ &\ge V_{\text{lowb}}(0,\ N,\ \kappa_{\text{cand}}(n,\ N)). \end{aligned} \tag{8}$$

Thus, $V_{\text{lowb}}(0,\ N,\ \kappa_{\text{cand}}(n,\ N))$ is a lower bound on the optimal cost using one of the sequences in $\kappa_{\text{cand}}(n,\ N)$ for steps from n to N [from now on *passing* $\kappa_{\text{cand}}(n,\ N)$].

Lemma 1—Best Cost Including Pruned Sequences: It holds that

$$\begin{aligned} &V(0,\ N,\ \kappa_{\text{cand}}(n,\ N)\cup\kappa_{\text{prune}}(n,\ N)) \\ &\quad\ge \min_{(c_{\text{prune}},\,\alpha,\,c_{\text{cand}})\in M'(n,\ N)} \{(1-\alpha)V_{\text{lowb}}(0,\ n) \\ &\qquad + \alpha(V_{\text{lowb}}(0,\ N,\ \kappa_{\text{cand}}(n,\ N)) - c_{\text{cand}}) + c_{\text{prune}}\} \\ &\quad =: V_{\text{lowb}}(0,\ N,\ \kappa_{\text{cand}}(n,\ N)\cup\kappa_{\text{prune}}(n,\ N)) \end{aligned} \tag{9}$$

where $V_{\text{lowb}}(0,\ n)$ is a lower bound on the optimal cost in the length-n -problem, and $M'(n,\ N) = M(n,\ N)\cup\{(0,\ 1,\ 0)\}$ to include the current lower bound.

Thus $V_{\text{lowb}}(0,\ N,\ \kappa_{\text{cand}}(n,\ N)\cup\kappa_{\text{prune}}(n,\ N))$ is a lower bound for the cost achieved if no sequences were pruned at step n. If $V_{\text{cand}} = V_{\text{lowb}}(0,\ N,\ \kappa_{\text{cand}}(n,\ N)\cup\kappa_{\text{prune}}(n,\ N))$ then V_{cand} is also the optimal cost passing $\kappa_{\text{cand}}(n,\ N)\cup\kappa_{\text{prune}}(n,\ N)$. ■

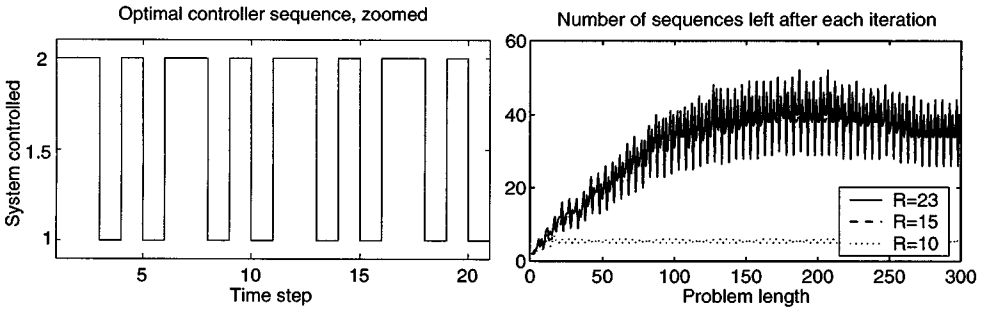

Fig. 3. The optimal controller sequence for Example 1 (left), and the number of sequences left after pruning in each iteration. The $R = 15$ case is almost the same as for $R = 23$ and, therefore, not visible. Generally, the algorithm often find sequences with short period, but not always.

For notational convenience we put

$$\sigma_0^{n-1} = \sum_{i=0}^{n-1} \begin{bmatrix} z(i) \\ u(i) \end{bmatrix}^T Q_{k(i)} \begin{bmatrix} z(i) \\ u(i) \end{bmatrix}$$

in the following.

Proof of Lemma 1: First, we notice that (8) gives

$$\begin{aligned} &\min_{\substack{K(0,n) \\ K(n,N)\in\kappa_{\text{cand}}(n,N)}} V(0,\, N,\, [K(0,\, n)\, K(n,\, N)]) \\ &\quad = \min_{\cdots} \mathbf{E}_v \left\{ \sigma_0^{n+1} + z(n)^T S_{K(n,N)} z(n) + c_{K(n,N)} \right\} \\ &\quad \geq V_{\text{lowb}}(0,\, N,\, \kappa_{\text{cand}}(n,\, N)) \Rightarrow \forall\, K(n,\, N) \in \kappa_{\text{cand}}(n,\, N), \\ &\min_{K(0,n)} \mathbf{E}_v \left\{ \sigma_0^{n-1} + z(n)^T S_{K(n,N)} z(n) \right\} \\ &\quad \geq V_{\text{lowb}}(0,\, N,\, \kappa_{\text{cand}}(n,\, N)) - c_{K(n,N)}. \end{aligned} \tag{10}$$

Second, for two sequences from n to N, $K_{\text{prune}}(n,\, N)$ and $K_{\text{cand}}(n,\, N)$, for which

$$S_{K_{\text{prune}}(n,\,N)} - Q_N \geq \alpha (S_{K_{\text{cand}}(n,\,N)} - Q_N) \tag{11}$$

and $K_{\text{cand}} \in \kappa_{\text{cand}}(n,\, N)$ it holds that

$$\begin{aligned} &\min_{K(0,n)} \mathbf{E}_v \left\{ \sigma_0^{n-1} + z(n)^T S_{K_{\text{prune}}(n,\,N)} z(n) \right\} \\ &\quad \geq \min_{K(0,n)} \mathbf{E}_v \left\{ \sigma_0^{n-1} + z(n)^T Q_N z(n) \right. \\ &\qquad \left. + \alpha z(n)^T (S_{K_{\text{cand}}(n,\,N)} - Q_N) z(n) \right\} \\ &\quad \geq \alpha \min_{K(0,n)} \mathbf{E}_v \left\{ \sigma_0^{n-1} + z(n)^T S_{K_{\text{cand}}(n,\,N)} z(n) \right\} \\ &\qquad + (1-\alpha) \min_{K(0,n)} \mathbf{E}_v \left\{ \sigma_0^{n-1} + z(n)^T Q_N z(n) \right\} \\ &\quad \geq \alpha (V_{\text{lowb}}(0,\, N,\, \kappa_{\text{cand}}(n,\, N)) - c_{K_{\text{cand}}(n,\,N)}) \\ &\qquad + (1-\alpha) V_{\text{lowb}}(0,\, n). \end{aligned} \tag{12}$$

Thus, we can put a lower bound on the optimal cost passing a pruned sequence $K_{\text{prune}}(n,\, N) \in \kappa_{\text{prune}}(n,\, N)$ using lower bounds for the cost of the length-n-problem and for the cost of sequences passing $\kappa_{\text{cand}}(n,\, N)$

$$\begin{aligned} &\min_{K(0,\,n),} V(0,\, N,\, [K(0,\, n) K_{\text{prune}}(n,\, N)]) \\ &\quad = \min_{K(0,n)} \mathbf{E}_v \left\{ \sigma_0^{n-1} + z(n)^T S_{K_{\text{prune}}(n,\,N)} z(n) \right\} \\ &\qquad + c_{K_{\text{prune}}(n,\,N)} \\ &\quad \geq \alpha (V_{\text{lowb}}(0,\, N,\, \kappa_{\text{cand}}(n,\, N)) - c_{K_{\text{cand}}(n,\,N)}) \\ &\qquad + (1-\alpha) V_{\text{lowb}}(0,\, n) + c_{K_{\text{prune}}(n,\,N)}. \end{aligned} \tag{13}$$

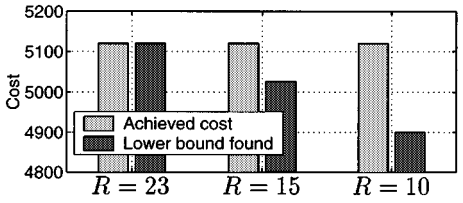

Fig. 4. Found sequence costs and lower bounds when running the tree pruning algorithm with different Rs.

A lower bound of all sequences passing $\kappa_{\text{cand}}(n,\, N) \cup \kappa_{\text{prune}}(n,\, N)$ is obtained by taking the minimum of lower bounds for all pruned sequences and the remaining sequences, yielding Equation (9). ■

Lemma 2—Lower Bound on Full Cost: Given the candidate sequence $K_{\text{cand}}(0,\, N)$ from (7), lower bounds $V_{\text{lowb}}(0,\, i)$ for the length-i-problems, $i \in \{0, \ldots, N-1\}$, and the pruned sequence motivations $M(i,\, N)$ $i \in \{0, \ldots, N\}$, a lower bound $V_{\text{lowb}}(0,\, N)$ on the optimal cost can be found by iterating (9) from $n = 1$ to $n = N$.

If $V_{\text{cand}} = V_{\text{lowb}}(0,\, N)$, then $K_{\text{cand}}(0,\, N)$ is a sequence that gives the optimal cost. ■

Proof of Lemma 2: Since all sequences $K(n,\, N) \in \kappa_{\text{cand}}(n,\, N) \cup \kappa_{\text{prune}}(n,\, N)$ pass $\kappa_{\text{cand}}(n+1,\, N)$, the iteration can use that

$$\begin{aligned} &V_{\text{lowb}}(0,\, N,\, \kappa_{\text{cand}}(n,\, N) \cup \kappa_{\text{prune}}(n,\, N)) \\ &\qquad = V_{\text{lowb}}(0,\, N,\, \kappa_{\text{cand}}(n+1,\, N)). \end{aligned} \tag{14}$$

Let $V_{\text{lowb}}(0,\, N,\, \kappa_{\text{cand}}(0,\, N)) = V_{\text{cand}}$. Equation (14) and Lemma 1 give $V_{\text{lowb}}(0,\, N,\, \kappa_{\text{cand}}(i,\, N))$, $i \in \{0, \ldots, N\}$. A lower bound on the cost for the length-N-problem is then $V_{\text{lowb}}(0,\, N) = V_{\text{lowb}}(0,\, N,\, \emptyset)$. ■

Using Lemma 2 iteratively, lower bounds (or optimal costs) for the length-N-problem can be found by starting with a length-1-problem and iterating. The lower bound on the solution for each problem is found and used in the calculation of lower bounds for larger problems. The sequence tree can be kept from the previous problem length and expanded by one step for each iteration, keeping complexity low.

Theorem 1—Optimal Sequence for Finite R: If

$$R \geq V(0,\, N) - (V(0,\, n) + c_{\min}(n,\, N)) \quad \forall n \in \{0 \cdots N\} \tag{15}$$

then an optimal sequence will be found. Also, if the optimal costs for problem lengths $1, \ldots, N-1$ have been found before s.t. $V_{\text{lowb}}(0,\, n) = V_{\text{cand}}(0,\, n) = V(0,\, n)$, $\forall\, n \in \{0, \ldots, N-1\}$, then the found cost for problem length N will also be proven optimal by Lemma 2. ■

Note: This theorem cannot be used to choose R directly, as $V(0,\, N)$ is not known before the optimziation. It only states that for large enough

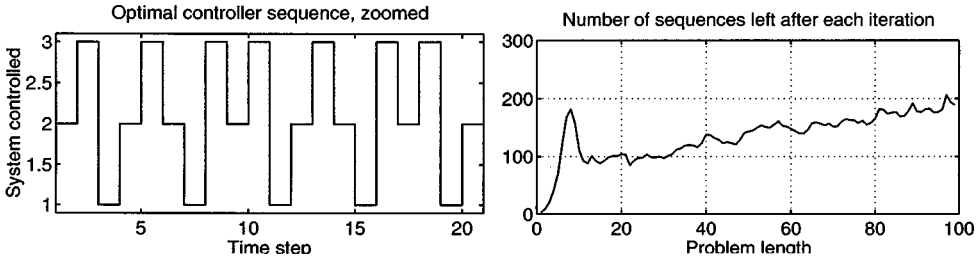

Fig. 5. The optimal controller sequence for Example 2, with three choices in each time step (left), and the number of sequences left after pruning in each iteration. The control signal is held when the system is not controlled, so the augmented system is of order eight. The sequence is periodic with period 8. Note that although the tree becomes rather large, it is still solvable, and a length-100-problem should give us a good insight in the steady-state behavior.

R, we will find and prove the optimal cost. The second assumption in the theorem, that optimal costs must be found for all shorter problems, is automatically satisfied if the tree is expanded iteratively and (15) holds in each step.

Proof: To show that all optimal sequences cannot be pruned, we note that the lower bound on costs of pruned sequences in (9) holds also when all lower bounds on candidate sequence costs are replaced by optimal costs. We want to show that these lower bounds are all worse than the optimal cost

$$\forall\,(c_{\text{prune}},\,\alpha,\,c_{\text{cand}}) \in M,\quad ((1-\alpha)V(0,\,n) + \alpha(V(0,\,N) - c_{\text{cand}}) + c_{\text{prune}}) - V(0,\,N) \geq 0 \quad (16)$$

and expanding the left-hand side

$$\begin{aligned}
&(1-\alpha)(V(0,\,n) - V(0,\,N)) - \alpha c_{\text{cand}} + c_{\text{prune}} \\
&\quad \geq (1-\alpha)(V(0,\,n) - V(0,\,N)) - \alpha c_{\text{cand}} + c_{\text{cand}} \\
&\qquad + \underbrace{(1-\alpha)(c_{\min}(n,\,N) + R - c_{\text{cand}})}_{\text{From pruning rule}} \\
&\quad = (1-\alpha)(\{V(0,\,n) + c_{\min}(n,\,N) + R\} - V(0,\,N)) \\
&\quad \geq 0 \;\Leftrightarrow\; R \\
&\quad \geq V(0,\,N) - (V(0,\,n) + c_{\min}(n,\,N)). \qquad (17)
\end{aligned}$$

This shows the role of R: $V(0,\,n)$ is the cost for the problem from 0 to n and $c_{\min}(n,\,N)$ is the cost for the problem from n to N without initial variance. If we add the costs of these two shorter problems together it will be lower than the cost from 0 to N ($V(0,\,N)$), and R has to be larger than the difference for the inequality to hold.

Thus, if (15) holds, the optimal sequence is found, i.e., $V_{\text{cand}}(0,\,N) = V(0,\,N)$. If also $V_{\text{lowb}}(0,\,n) = V(0,\,n)$, then the lower bound on cost from pruned sequences in (16) equals the one in (9) and from the latter we obtain

$$V_{\text{lowb}}(0,\,N,\,\kappa_{\text{cand}}(n,\,N) \cup \kappa_{\text{prune}}(n,\,N)) = V_{\text{lowb}}(0,\,N,\,\kappa_{\text{cand}}(n,\,N)).$$

Thus, Lemma 2 will prove that our candidate sequence is optimal. ■

By keeping R small, the number of branches in the tree can be kept down to a reasonable level. If R is chosen too small, the optimal solution might however not be found, or at least not be proven optimal using Lemma 2. A lower bound on the optimal case is always found, though.

IV. Time Complexity of the Algorithm

The described algorithm does not run in polynomial time (unless P = NP). In fact, one special version of the problem (which was not considered in the design of the algorithm) is easily shown to be NP-hard. The proof is based on solving an *N-mortality problem* which is NP-complete (see [9]).

V. Examples

To show the feasibility of the method, two examples have been constructed. They are based on the select-which-system-to-control problem, with different properties. The optimization times range from seconds to minutes in Matlab code on a standard computer.

A. Example 1

Consider the following two simple linear systems:

$$\begin{aligned}
x_1(n+1) &= \begin{bmatrix} 1 & 0.4 \\ 0.3 & 0.8 \end{bmatrix} x_1(n) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(n) + \begin{bmatrix} 0 \\ 0.8 \end{bmatrix} v_1(n) \\
x_2(n+1) &= 0.9x_2(n) + 1u(n) + 1v_2(n) \\
Q &= \text{diag}([1 \quad 1 \quad 10 \quad 1]).
\end{aligned}$$

The problem is to find the control access sequence which yields the lowest cost. In this problem, the control signal will only be held at the actuator for one sample period. If the system is not being controlled during the next sample period, the control signal is zero. Running the described tree optimization algorithm with $R = 23$ and $N = 300$ produce candidate sequences for all length-i-problems, $i \in \{0,\,\ldots,\,300\}$, which all prove to be optimal. For illustration, the algorithm has been run with $R = 15$, and $R = 10$ as well. Optimality could not be shown for these choices of R, but for $R = 15$ the same cost as for $R = 23$ was achieved. See Fig. 3 for the optimal sequence, and number of sequences left in each iteration of the algorithm. Fig. 4 compares found costs and guaranteed lower bounds for the different choices of R.

B. Example 2

We now show a larger example consisting of the two systems in Example 1 plus another unstable second order system. The control signal is held at the actuator if the system is not controlled, so extra "control-signal" states have been added, making the original order five system grow to order eight. The optimal controller sequence for the length-100-problem can be seen in Fig. 5.

VI. Conclusion

A method to find the optimal switching sequence in a linear-quadratic full-information problem has been presented, together with a method to prove optimality in each case. Empirically, the method works well in that it finds the solution in reasonable time. Future work could include formulating other problems in the same framework, such as for example choosing among distributed sensors. The infinite horizon problem and the problem of joint actuator scheduling and sensor scheduling are open.

REFERENCES

[1] M. S. Branicky and S. M. Mitter, "Algorithms for optimal hybrid control," in *Proc. 34th IEEE Conf. Decision and Control*, 1995, pp. 2661–2666.

[2] M. S. Branicky, V. S. Borkar, and S. M. Mitter, "A unified framework for hybrid control: Model and optimal control theory," *IEEE Trans. Automat. Contr.*, vol. 43, pp. 31–45, Jan. 1998.

[3] R. DeCarlo, M. Branicky, S. Petterson, and B. Lennartsson, "Perspectives and results on the stability and stabilizability of hybrid systems," *Proc. IEEE*, vol. 88, pp. 1069–1082, July 2000.

[4] J. Haartsen, "Bluetooth—The universal radio interface for *ad hoc*, wireless connectivity," *Ericsson Rev.*, vol. 3, pp. 110–117, 1198.

[5] K.-E. Årzén, A. Cervin, J. Eker, and L. Sha, "An introduction to control and real-time scheduling co-design," in *Proc. 39th Conf. Decision and Control*, Dec. 2000, pp. 4865–4870.

[6] E. Skafidas and A. Nerode, "Optimal measurement scheduling in linear quadratic Gaussian control problems," in *Proc. 1998 IEEE Int. Conf. Control Applications*, 1998, pp. 1225–1229.

[7] E. Skafidas, R. J. Evans, and I. M. Marcels, "Optimal controller switching for stsachastic systems," in *Proc. 36th Conf. Decision and Control*, 1997, pp. 3950–3955.

[8] H. Rehbinder and M. Sanfridson, "Scheduling of a limited communication channel for optical control," in *Proc. 39th Conf. Decision and Control*, 2000, pp. 1011–1016.

[9] V. D. Blondel and J. N Tsitsklis, "When is a pair of matrices mortal?," *Inform. Processing Lett.*, no. 63, pp. 283–286, 1997.

[10] J. N. Tsitsiklis and V. D. Blondel, "The Lyapunov exponent and joint spectral radius of pairs of matrices are hard—When not impossible—To compute and to approximate," *Math. Control, Signals, Syst.*, no. 10, pp. 31–40, 1997.

The Impact of Finite Buffers on the Optimal Scheduling of a Single-Machine Two-Part-Type Manufacturing System

Francesco Martinelli and Paolo Valigi

***Abstract*—In this note, we give a complete solution to a scheduling problem for a two-part-type, single-machine, flexible manufacturing system, with *finite-capacity* buffers. Backlogged and rejected requests incur a cost which must be minimized over an infinite time interval. If buffer capacities were infinite, the well-known $c\mu$ rule would have solved the problem. In this note, we find the optimal policy for the finite-capacity case and give a computation procedure and some illustrative examples.**

***Index Terms*—Finite buffers, fluid model, manufacturing systems, scheduling.**

I. INTRODUCTION

In this note, we consider a make-to-order, two-part-type manufacturing system comprising a single reliable and flexible machine. A fluid model is considered, with demands arriving with a deterministic constant rate and a holding cost associated with waiting demands. A maximum number of waiting demands is allowed and extra demands are dropped, with a cost for lost demands. The objective of the note is to find the dynamic scheduling policy which minimizes the total cost over an infinite time horizon.

The optimal solution turns out to be the combination of two classes of policies, which will be defined next and will be referred to as the "finite $c\mu$ rule" and the "two-phase motion."

Dynamic scheduling for manufacturing systems has been studied for decades, with a variety of approaches (see [1]–[5]). Most problems have been studied for systems with infinite-capacity buffers. In this case, for a broad class of single-machine systems, under fairly general assumptions on the arrival process (comprising the system considered in this note if buffer capacities were infinite), the optimal solution of the scheduling problem is the well-known "$c\mu$ rule" [6]–[8].

Finite-capacity buffers make the problem much more difficult: the $c\mu$ rule is no longer optimal, analytical solutions become complex, and the problem is often solved through a numerical approach or for a restricted set of parameter values.

A scheduling problem for a system with finite-capacity buffers, stochastic demand, and service has been considered in [9], with the objective of minimizing a discounted cost including a linear cost for waiting customers, and a blocking cost incurred whenever arrivals encounter a full queue and are lost. The optimal policy is found under the assumption that the holding cost is not larger than the blocking cost times the arrival rate, for each queue. It turns out that the optimal policy is characterized by a switching curve dividing the state space into two decision regions. The optimal policy maintains the same structure also in the undiscounted case [9]. A brief sketch of extension to $n > 2$ part types is given in [10]. Finite buffers and explicit rejection cost in a stochastic scenario have been assumed also in [11]. Key properties of the optimal solution have been investigated under some assumptions on system parameters, while numerical results are proposed to indicate that, in the general case, the optimal policy has not a threshold structure. Both [9] and [11] use dynamic programming to investigate the problem.

In [12], motivated by [13], we derived a numerical solution, based on discrete dynamic programming, for a single-machine, two-part-type, make-to-stock, finite-buffer system, with Poisson arrivals and service. Assuming part types are not distinguishable (symmetric case), we found in [12] an interesting result: if the cost for lost demand is larger than a bound, the optimal policy is a longest queue first policy, while, if such a cost is lower than another bound, we got a shortest queue first policy. To verify the conjecture that such a result was due to finite buffers, we studied in [14] a symmetric deterministic fluid system, with finite buffers, proving the necessity of a full-capacity policy, and stating the need for a shortest queue first policy, if cost for demand loss is less than a threshold. The system considered in [14] has been further studied in [15], where the symmetry assumption has been relaxed. The solution proposed in [15] is optimal only within a subset of all the admissible policies, not comprising the two-phase motion considered in this note.

Here, we present a policy which is optimal in the set of all the admissible policies, and whose optimality is analytically proved. As in [15], we have considered a very general fluid model without any assumption on cost parameters, unlike previous papers. Such an optimal policy clarifies the structure obtained in [12]. In addition, if the rejection cost is large enough, we get a policy (Algorithm 1) similar to the one presented in the example reported in [9]. The main difference is that our policy will travel exactly on the border of the state space, being in a deterministic scenario.

A further relevant motivation for this study is that the production model considered here can be regarded as a real model for a make-to-stock system where the failure rate of the machine and the cost of positive surplus are such that the hedging point is at the origin. This kind of problem has been mentioned in [5].

Manuscript received June 5, 2000; revised June 6, 2001 and April 12, 2002. Recommended by Associate Editor X. Zhou. This work was supported in part by MIUR and ENEA.

F. Martinelli is with the Dipartimento di Informatica, Sistemi e Produzione, Università degli Studi di Roma "Tor Vergata," 00133 Rome, Italy (e-mail: martinelli@disp.uniroma2.it).

P. Valigi is with the Dipartimento di Ingegneria Elettronica e dell'Informazione, Università di Perugia, 06125 Perugia, Italy (e-mail: valigi@diei.unipg.it).

Digital Object Identifier 10.1109/TAC.2002.803540.