Optimizing linear system switching

Lincoln, Bo; Rantzer, Anders

Published in:
Proceedings of the 40th IEEE Conference on Decision and Control, 2001.

DOI:
10.1109/.2001.980555

2001

Proceedings of the 40th IEEE
Conference on Decision and Control
Orlando, Florida USA, December 2001

Optimizing Linear System Switching

Bo Lincoln, Anders Rantzer

Department of Automatic Control, LTH
Box 118, 221 00 Lund, Sweden
{ lincoln | rantzer } @ control.lth.se

Abstract

In this paper an algorithm to optimize switching sequences for a class of switched linear problems is presented. The algorithm searches for solutions which are arbitrarily close to optimal – finding the optimal solution does often require a much larger search. Both deterministic and stochastic problems are considered.

1. Introduction

This paper considers a class of control problems where there is a need to find *switching-sequences* between different linear systems, as well as linear control laws, to minimize some quadratic cost function. Often, this kind of problems occur in implementation of digital controllers, where issues of sharing CPU power or network bandwidth have to be dealt with. These sharing or scheduling problems can be modeled as control problems where the controller not only has to control a plant, but also choose between discrete actions ("use network to send this or that right now?").

Similar problems has been studied by a number of groups before, in e.g. [1, 2, 4, 5, 6]. These papers, though, do not present any efficient optimization methods for finding switching sequences. In [3], we presented a optimization method which could find the optimal solution to some of these problems. In this paper, a new result will be presented, where we search for solutions which are not optimal, but α-optimal. An α-optimal solution, in this case, means that the resulting cost is at most α times the optimal cost. In problems where many sequences give similar results, an α slightly larger than 1 increases optimization speed significantly, and we can solve larger problems. It also enables us to optimize deterministic switching problems, without state noise.

2. Problem Formulation

Consider the following problem: Given a standard discrete-time linear system

$$x(n+1) = \Phi(n)x(n) + \Gamma(n)u(n) + G(n)v(n). \quad (1)$$

where x is the state space vector, u the control signals, and v standard Gaussian, independent, disturbances with zero mean and unit covariance. The system matrices $\Phi(n)$, $\Gamma(n)$ and $G(n)$ can be chosen by the controller in each step from a small set of M systems $\{(\Phi_i, \Gamma_i, G_i)\}, i = 1, \ldots, M$.

The problem is to find the linear feedback law $u(n) = -L(n)x(n)$ and the sequence $K(0,N) = \{k(0), k(1), k(2), \ldots, k(N)\}$ 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 minimizes the cost

$$V(P(0), L(\cdot), K(0,N)) = \mathop{\mathbf{E}}_{v}\left\{\sum_{n=0}^{N-1}\begin{pmatrix}x(n)\\u(n)\end{pmatrix}^T Q(n)\begin{pmatrix}x(n)\\u(n)\end{pmatrix}\right\}$$

where $\mathbf{E}\{x(0)\} = 0$ and $\mathbf{E}\{x(0)x(0)^T\} = P(0)$. Thus, the sequence $K(0,N)$ is to be optimized *off-line*, minimizing the expected cost.

2.1 Equivalent deterministic problem

Consider this equivalent problem: Let $P(n)$ (the x variance) be the new state. Its dynamics can be written

$$P(n+1) = \Big(\Phi(n) - \Gamma(n)L(n)\Big)^T P(n)\Big(\Phi(n) - \Gamma(n)L(n)\Big) + R(n)$$

where $R(n) = \mathbf{E}\Big\{G(n)v(n)v(n)^T G(n)^T\Big\}$. The cost is

now

$$V(P(0),L(\cdot),K(0,N)) = \sum_{n=0}^{N-1} \mathrm{tr}\left(\begin{pmatrix} I \\ -L(n) \end{pmatrix} P(n) \begin{pmatrix} I \\ -L(n) \end{pmatrix}^T Q_{k(n)} \right) = \sum_{n=0}^{N-1} \varphi(P(n),n,k(n)). \quad (2)$$

This problem is deterministic with linear dynamics and linear cost. The state-space is the cone of positive matrices.

2.2 Optimal Cost

Let $V^*(P(n),n)$ denote the optimal cost starting with state P at time n. Then V^* satisfies

$$V^*(P(n),n) = \min_{k(n),L(n)} \Big\{ \varphi(P(n),n,k(n)) + V^*(P(n+1),n+1) \Big\} \quad (3)$$

Thus the optimal sequence step k at time n can be written as the state feedback

$$k(n) = F(P(n),n) = \operatorname*{argmin}_k \Big\{ \varphi(P(n),n,k(n)) + V^*(P(n+1),n+1) \Big\}. \quad (4)$$

In fact, as the problem is deterministic, the full sequence of $K(n,N)$ can be determined from $P(n)$.

Now consider a non-optimal cost-function V which satisfies

$$V(P(n),n) \le \alpha \cdot \varphi(P(n),n,k(n)) + V(P(n+1),n+1) \ \forall P(n),\ n \quad (5)$$

where $\alpha \in \mathbf{R}, \alpha \ge 1$. Then

$$V(P(0),0) - V(P(N),N) = V(P(0),0) = \sum_{n=0}^{N-1} V(P(n),n) - V(P(n+1),n+1) \le \sum_{n=0}^{N-1} \alpha\varphi(P(n),n), \quad (6)$$

so V is a lower bound on the optimal cost scaled by α.

In this paper, we use α to be able to find a solution ($K(0,N)$ and $L(\cdot)$) with a cost which is **at most α times the optimal cost**. The idea is, for some α find a V which satisfies (5), and an actual solution

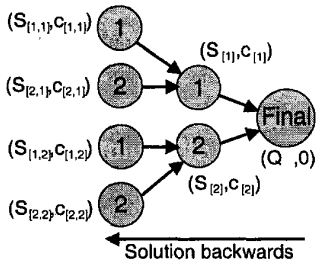

Figure 1: The control sequence tree for $M = 2$ when expanding all possibilities from $N-3$ to N.

($K(0,N)$ and $L(\cdot)$) which gives exactly $V(P(n),n)$. Then the solution is an upper bound on V^*, and also a lower bound on $\alpha \cdot V^*$ so

$$V^*(P(0),0) \le V(P(0),L(\cdot),K(0,N)) \le \alpha V^*(P(0),0) \quad (7)$$

Let us denote this solution α-optimal.

3. Finding an α-optimal solution

We want to find a sequence $K(0,N)$ and corresponding feedback gains $L(n)$ which satisfies (7). This problem is hard (probably NP-hard), and therefore we do not expect an algorithm that works for *all* problems.

In order to simplify this section, we will here only consider problems where $\Gamma = 0$, i.e. there is no control signal u. Thus, we are only interested in the switching sequence $K(0,N)$.

The problem will be solved by expanding a sequence tree, starting at time N (see Figure 1). The idea is that we can remove branches from (prune) the tree without violating (5).

For some time n, let

$$V(P(n+1),n+1) = \min_{K(n+1,N)\in\kappa(n+1,N)} \mathrm{tr}\Big(P(n+1)S_{K(n+1,N)}\Big) + c_{K(n+1,N)}, \quad (8)$$

where S_K is a positive symmetric matrix (state cost), and c_K is a constant (noise cost). Each pair (S_K, c_K) corresponds to choosing sequence K from $n+1$ to N. $\kappa(n+1,N)$ is a set of possible sequences (not necessarily all).

Expanding the tree backwards to time n by forming $\kappa'(n,N) = \{1,2,\ldots,M\} \times \kappa(n+1,N)$ (making all possible sequence choices for time n), the cost for each sequence is calculated as

$$S_{K(n,N)} = \Phi_{k(n)}^T S_{K(n+1,N)} \Phi_{k(n)} + Q_{k(n)} \quad (9)$$

and

$$c_{K(n,N)} = c_{K(n+1,N)} + \mathrm{tr}\Big(G_{k(n)}^T S_{K(n+1,N)} G_{k(n)}\Big). \quad (10)$$

These form a cost function $V'(P(n),n)$ of the same form as (8). Note that V' satisfies (5) for any $\alpha \geq 1$. We want to remove some $K(n,N)$ from $\kappa'(n,N)$ to reduce the size of κ, while still satisfying (5).

Now form

$$S^{\alpha}_{K(n,N)} = \Phi^{T}_{k(n)} S_{K(n+1,N)} \Phi_{k(n)} + \alpha Q_{k(n)}, \qquad (11)$$

and $c^{\alpha}_{K(n,N)} = c_{K(n,N)}$. The corresponding cost function V^{α} also satisfies (5), but now with a *tight bound*. Any cost function V with

$$V(P(n),n) \leq V^{\alpha}(P(n),n) \qquad (12)$$

would still satisfy (5). $V'(P(n),n)$ is one such function, but it consists of M times more sequences than $V(P(n+1),n+1)$. We now simply remove sequences from $\kappa'(n,N)$ as long as (12) holds.

PROCEDURE 1—TREE PRUNING
Let $\kappa(n,N) := \kappa'(n,N)$.
For all $K_{\text{keep}} \in \kappa$:

For all $K_{\text{prune}} \in \kappa$ with $c_{K_{\text{prune}}} \geq c_{K_{\text{keep}}}$

If $S^{\alpha}_{K_{\text{prune}}} \geq S_{K_{\text{keep}}}$ then remove K_{prune} from $\kappa(n,N)$

The remaining $\kappa(n,N)$ with corresponding set of (S_K, c_K) forms the new $V(P(n),n)$. □

THEOREM 1—α-OPTIMALITY OF V
The $V(P(n),n)$ obtained from Procedure 1 satisfies (5), and the solutions in $\kappa(n,N)$ are thus α-optimal. □

Proof: By construction of Procedure 1 it holds that

$$\begin{aligned} V(P(n),n) &\leq V^{\alpha}(P(n),n) \\ &\leq \alpha \cdot \varphi(P(n),n,k(n)) + V(P(n+1),n+1) \end{aligned} \qquad (13)$$

for any $P(n)$.

3.1 The Solution

To find the full α-optimal solution from time 0 to N, repeat the procedure from $n = N$ with $\kappa(N,N) = end$, $S_{end} = 0$ and $c_{end} = 0$ and expand backwards until $n = 0$. Note that this solution method does not at all rely on noise, but can equally well solve the problem in the noise free case (with only initial variance).
The algorithm works well for many problems, in that it can return an α-optimal solution with a reasonable search tree size. For problems with many sequences giving approximately the same cost, α can often be set to e.g 1.001 to find a solution which is very close to the optimal. The algorithm in [3] would have a very hard time trying to find *the* optimal solution to such a problem.

3.2 Optimizing with Control Signal

The above method was described with $\Gamma = 0$ to make the idea clearer. With $\Gamma \neq 0$, the same ideas hold, but the calculations of S_K (9) and (11) change to the standard LQ-iteration.

$$\begin{aligned} S_{K(n,N)} = \\ \left(\Phi_{k(n)} - \Gamma_{k(n)} L_{K(n,N)}(n)\right)^{T} S_{K(n+1,N)} \Phi_{k(n)} \times \\ \left(\Phi_{k(n)} - \Gamma_{k(n)} L_{K(n,N)}(n)\right) + Q_{k(n)} \end{aligned} \qquad (14)$$

where

$$L_{K(n,N)}(n) = {F^{uu}_{K(n,N)}}^{-1} {F^{xu}_{K(n,N)}}^{T} \qquad (15)$$

and

$$\begin{aligned} F_{K(n,N)} = \left[\begin{array}{c|c} F^{xx}_{K(n,N)} & F^{xu}_{K(n,N)} \\ \hline F^{ux}_{K(n,N)} & F^{uu}_{K(n,N)} \end{array}\right] = \\ Q_{k(n)} + \begin{pmatrix} \Phi_k & \Gamma_k \end{pmatrix}^{T} S_{K(n+1,N)} \begin{pmatrix} \Phi_k & \Gamma_k \end{pmatrix} \end{aligned} \qquad (16)$$

4. Improving Search

An α-optimal cost-function found by using the algorithm in Section 3 contains a solution for *all* $P(0)$, including very degenerate ones. A $P(0)$ such as diag($[10^{100}\ 0]$) would e.g. force the optimizer to ignore state 2, and "normal-sized" state noise would make no difference at all.
In this section, we will present an addition to Procedure 1, which excludes sequences which would require very large or skewed $P(n)$ to ever be chosen in a minimization. This addition will often decrease the size of κ a lot, but it requires the input of a problem-dependent parameter R to the optimizer. The idea behind this additional pruning is to remove sequences which have a very bad noise cost compared to other sequences, but where the S (cost) matrix is a bit better in some direction. This means that it may take a very large P for this sequence to come into question. Consider two sequences at time n: $K_{\text{keep}}(n,N)$ and $K_{\text{prune}}(n,N)$ (where the names reflect possible actions). Their corresponding costs are parameterized by $(S_{K_{\text{keep}}}, c_{K_{\text{keep}}})$ and $(S_{K_{\text{prune}}}, c_{K_{\text{prune}}})$, and we assume that $c_{K_{\text{prune}}} \geq c_{K_{\text{keep}}}$.

4.1 Minimum Cost Calculation

We want to calculate the smallest possible cost $C_{\min}$ of $K_{\text{prune}}(n,N)$ so that it is lower than $K_{\text{keep}}(n,N)$, i.e.

$$\text{tr}\left(P S_{K_{\text{prune}}}\right) + c_{K_{\text{prune}}} \leq \text{tr}\left(P S_{K_{\text{keep}}}\right) + c_{K_{\text{keep}}} \qquad (17)$$

or written as

$$\mathrm{tr}\Big(P(\underbrace{S_{K_{\text{keep}}} - S_{K_{\text{prune}}}}_{\Delta S})\Big) \quad \geq \quad \underbrace{c_{K_{\text{prune}}} - c_{K_{\text{keep}}}}_{\Delta c}. \quad (18)$$

$\Delta c \geq 0$ is given. Thus, if $\Delta S \leq 0$ then (17) cannot hold for any P, and K_{keep} is better than K_{prune} for all P. This case is already handled by Procedure 1.
We can formulate the problem as

$$C_{\min} = \min_{P\geq 0} \mathrm{tr}(PS_{K_{\text{prune}}}) + c_{K_{\text{prune}}} \text{ s.t. } \mathrm{tr}(P\Delta S) \geq \Delta c =$$
$$\min_{P\geq 0} \max_{\lambda \geq 0} \Big\{ \mathrm{tr}(PS_{K_{\text{prune}}}) + c_{K_{\text{prune}}} + \lambda(\Delta c - \mathrm{tr}(P\Delta S)) \Big\}. \quad (19)$$

From the general inequality

$$\max_x \min_y F(x,y) \leq \min_y \max_x F(x,y) \quad (20)$$

we obtain

$$C_{\min} \geq \max_{\lambda\geq 0} \min_{P\geq 0} \Big\{ \mathrm{tr}(PS_{K_{\text{prune}}}) + c_{K_{\text{prune}}} + \lambda(\Delta c - \mathrm{tr}(P\Delta S)) \Big\} \quad (21)$$

which is actually an equality, but there is no need to prove it here. Rewriting it as

$$C_{\min} \geq \max_{\lambda\geq 0} \min_{P\geq 0} \Big\{ \mathrm{tr}(P(S_{K_{\text{prune}}} - \lambda\Delta S)) + \lambda\Delta c \Big\} + c_{K_{\text{prune}}} \quad (22)$$

it can be seen that the minimization will return $-\infty$ if $S_{K_{\text{prune}}} - \lambda\Delta S < 0$. Since $S_{K_{\text{prune}}} \geq 0$, $\lambda = 0$ gives a positive matrix. Thus the maximizing $\lambda_{\max}$ will be the smallest non-negative λ which solves

$$\det\Big(S_{K_{\text{prune}}} - \lambda\Delta S\Big) = 0, \quad (23)$$

i.e. the smallest positive generalized eigenvalue. Finally, we obtain

$$C_{\min} \geq \lambda_{\max}\Delta c + c_{K_{\text{prune}}} \quad (24)$$

For $K_{\text{keep}}(n,N)$ to be worse than $K_{\text{prune}}(n,N)$, cost $V_{K_{\text{prune}}}(n,N)$ from time n to N has to satisfy

$$V_{K_{\text{prune}}}(n,N) \geq C_{\min}. \quad (25)$$

4.2 Pruning

If $C_{\min}$ is large, then for the optimizer to choose K_{prune} instead of K_{keep} in the best sequence, the cost from n to N has to be large ($\geq C_{\min}$). A simple pruning rule is then to remove K_{prune} if it is much worse than the best solution $V(n,N)$ from n to N (using the initial variance $P(0)$ for $P(n)$).

PROCEDURE 2—AGGRESSIVE TREE PRUNING
Choose a constant R. Let $\kappa(n,N) := \kappa'(n,N)$.
For all $K_{\text{keep}} \in \kappa$:

For all $K_{\text{prune}} \in \kappa$ with $c_{K_{\text{prune}}} \geq c_{K_{\text{keep}}}$

If $S^{\alpha}_{K_{\text{prune}}} \geq S_{K_{\text{keep}}}$ then remove K_{prune} from $\kappa(n,N)$

If $C_{\min}(K_{\text{prune}}) \geq R + V(n,N)$ then move K_{prune} from $\kappa(n,N)$ to $\kappa_{\text{prune}}(n,N)$.

□

Thus, a small R will give aggressive pruning of the search tree, but at the risk of not being able to find an α-optimal solution for our $P(0)$. As $R \to \infty$, Procedure 2 will approach Procedure 1.

4.3 Proving α-optimality

In this section, we will show how to prove α-optimality of the found solution even when the more aggressive pruning rule is used. The resulting test has to be performed each iteration of the search, and if it fails, R must be increased and the search restarted. Let V^*_κ denote the optimal cost when the switching sequence $K(0,N)$ can only be chosen from the set κ. (For example $V^*_{\kappa(n,N)}(P(0),0)$ means that only sequences which uses one of the sequences in $\kappa(n,N)$ for steps n to N are considered.) Then

$$\alpha V^*_{\kappa(n,N)}(P(0),0) \geq \min_{K(0,N)\in\kappa(0,N)} \mathrm{tr}\Big(P(0)S_{K(0,N)}\Big) + c_{K(0,N)}. \quad (26)$$

Since all sequences $K(0,N)$ must either go through $\kappa(n,N)$ or $\kappa_{\text{prune}}(n,N)$, the optimal cost $V^*(P(0),0)$ satisfies

$$\alpha V^*(P(0),0) = \min_{K(0,N)} \alpha V^*_{K(0,N)}(P(0),0) \geq$$
$$\min \begin{cases} V_{\kappa(n,N)}(P(0),0) \\ V_{\kappa_{\text{prune}}(n,N)}(P(0),0) \end{cases} \geq$$
$$\min \begin{cases} \min_{K(0,N)\in\kappa(0,N)} \mathrm{tr}\Big(P(0)S_{K(0,N)}\Big) + c_{K(0,N)} \\ V(0,n) + R + V(n,N) \end{cases}. \quad (27)$$

The last inequality needs some explanation: In Section 4.1, we showed that if $K_{\text{prune}}(n,N)$ is better than $K_{\text{keep}}(n,N)$, the cost $V_{K_{\text{keep}}(n,N)}(0,N) \geq R + V(n,N)$. This means, we can disregard $V_{\kappa_{\text{prune}}(n,N)}(P(0),0)$ in the minimization, unless its cost from n to N is

greater than $R + V(n,N)$. In this case the total cost $V^*_{\text{prune}}(0,N)$ from 0 to N must satisfy

$$\alpha V^*_{\text{prune}}(0,N) \geq \min_{K(0,n-1)} \sum_{m=0}^{n-1} \varphi(P(m),m,k(m)) + R + V(n,N) = V(0,n) + R + V(n,N), \quad (28)$$

where $V(0,n)$ is an α-optimal solution to the length-n-problem. Also note that we know $V(0,n)$ as we expand the tree one time-slot at a time, and thus solve all problems from length-0 to length-N.
Our cost function from 0 to N is thus

$$V(0,N) = \min \begin{cases} \min_{K(0,N)\in\kappa(0,N)} \text{tr}\Big(P(0)S_{K(0,N)}\Big) + c_{K(0,N)} \\ \min_n \Big\{V(0,n) + R + V(n,N)\Big\}. \end{cases} \quad (29)$$

If then

$$\min_n \Big\{V(0,n) + V(n,N) + R\Big\} \geq \min_{K(0,N)\in\kappa(0,N)} \text{tr}\Big(P(0)S_{K(0,N)}\Big) + c_{K(0,N)}, \quad (30)$$

the minimizing $K(0,N) \in \kappa(0,N)$ is α-optimal, and the cost $V(0,N)$ is actually achievable.

5. Examples

In this section, some example problems which can be solved using the algorithm are presented.

5.1 Example 1 – Scheduling with noise

Given three different stable, linear systems with

$$\Phi_1 = 0.9 \qquad \Gamma_1 = 1 \qquad G_1 = 1$$
$$\Phi_2 = \begin{pmatrix} 1 & 0.1 \\ -0.2 & 0.8 \end{pmatrix} \quad \Gamma_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad G_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$
$$\Phi_3 = \begin{pmatrix} 1 & 0.1 \\ -0.5 & 0.9 \end{pmatrix} \quad \Gamma_3 = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad G_3 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

and

$$Q_1 = \text{diag}\begin{pmatrix} 10 & 1 \end{pmatrix}$$
$$Q_2 = \text{diag}\begin{pmatrix} 10 & 1 & 1 \end{pmatrix}$$
$$Q_3 = \text{diag}\begin{pmatrix} 1 & 1 & 1 \end{pmatrix},$$

and zero initial variance. Only one plant can be accessed each time-slot, so the problem is to find a switching sequence and feedback laws which give reasonable performance of all systems.

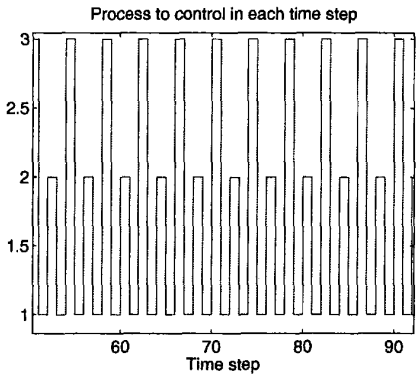

Figure 2: The resulting α-optimal sequence for Example 1.

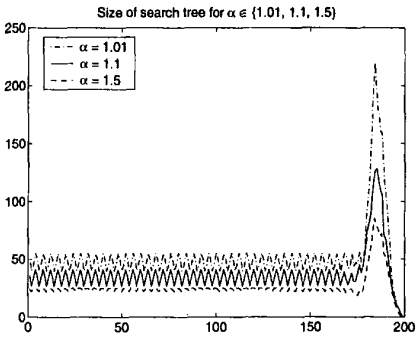

Figure 3: Size of the search tree: Number of candidates kept after each time step in the searches in Example 1. Note that the first step is at 200, as the tree is expanded backwards.

To show the effects of different choices of α, the problem was solved three times with $\alpha = \{1.01, 1.1, 1.5\}$ (i.e. for 1%, 10%, and 50% slack, respectively). Aggressive pruning was used with $R = 80$. The three found sequences are equal, showing that the algorithm with $\alpha = 1.5$ in this case really finds a solution which is 1.01-optimal. The found α-optimal switching sequence can be seen in Figure 2, and the search tree sizes in Figure 3.
As can be seen the search tree size seems to stabilize after a while, and in practice this problem can be solved for any time horizon. The problem is not very dependent on the initial variance, as the state noise gives most of the cost. Therefore the aggressive pruning works well to remove unlikely sequences – see Figure 4. In the next section, we present an example problem which lacks state noise.

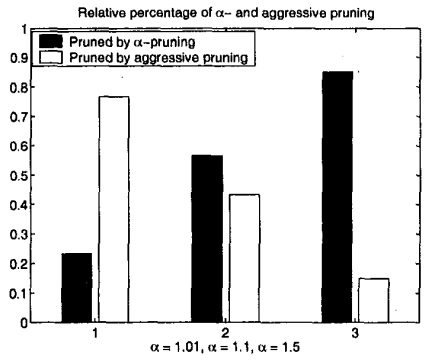

Figure 4: Relative usage of $\alpha-$ and aggressive pruning in Example 1. The aggressive pruning is only tried for a sequence if the α-pruning fails. With a larger α, more sequences can be α-pruned.

5.2 Example 2 – Scheduling without noise

Assume we have three equivalent inverted-pendulum processes, each with four states

$$x = \begin{pmatrix} \theta & \dot{\theta} & \varphi & \dot{\varphi} \end{pmatrix},$$

where θ and φ are the angles of the pendulum and the rotating base, respectively. The continuous dynamics are described by

$$\dot{x} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 31.3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -0.588 & 0 & 0 & 0 \end{pmatrix} x + \begin{pmatrix} 0 \\ -71.2 \\ 0 \\ 191 \end{pmatrix} u.$$

Again, the problem is to find a static access schedule for the controller, as it can only access one plant each time-slot. The time slot is 15 ms. In this example, there is *no* state noise, which means that the problem is deterministic, and the cost to go from 0 to N depends only on the initial state $x(0)$ (or $P(0)$). With

$$Q_1 = Q_2 = Q_3 = \text{diag}\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \end{pmatrix},$$

the problem is easily solved for $\alpha = 1.1$. Actually, the algorithm only has to keep the six permutations of the repeating sequence [1 2 3] as candidates, so the problem can be solved for any time horizon.
In practice, it means that for this problem, for any initial state $x(0)$ there is a round-robin scheduling which is at most 10% worse than any other scheduling. As $\alpha \to 1$, the tree size will grow and approach the full exponential size. This is due to the fact that for almost any sequence, there will be an initial state for which this sequence is optimal. Therefore, the α factor (slack) is *essential* in the noise-free case.

6. Conclusions

We have presented a method which efficiently searches for an α-optimal solution to a switched linear system problem. The algorithm can often solve the problem with reasonable effort, both in the deterministic and stochastic case.

7. References

[1] R. Brockett. "Stabilization of motor networks." In *Proceedings of the 34th Conference on Decision & Control*, pp. 1484–1488, 1995.

[2] D. Hristu and K. Morgansen. "Limited communication control." *System & Control Letters*, **No 37**, pp. 193–205, 1999.

[3] B. Lincoln and B. Bernhardsson. "Efficient pruning of search trees in LQR control of switched linear systems." In *Proceedings of the 39th Conference on Decision and Control*, 2000.

[4] H. Rehbinder and M. Sanfridson. "Scheduling of a limited communication channel for optimal control." In *Proceedings of the 39th Conference on Decision & Control*, 2000.

[5] E. Skafidas, R. Evans, and I. Marells. "Optimal controller switching for stochastic systems." In *Proceedings of the 36th Conf. on Decision and Control*, pp. 3950–3955, 1997.

[6] E. Skafidas and A. Nerode. "Optimal measurement scheduling in linear quadratic gaussian control problems." In *Proceedings of the 1998 IEEE Int. Conf. on Control Applications*, pp. 1225–1229, 1998.