Efficient Pruning of Search Trees in LQR Control of Switched Linear Systems

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

Abstract

This paper considers off-line optimization of a switching sequence for a given finite set of linear control systems and joint optimization of control laws. A linear quadratic full information criterion is optimized and dynamic programming is used to find the optimal switching sequence and control laws. 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.

1 Introduction and Motivation

In real-time control systems, there are often restrictions in communication bandwidth or CPU performance. The different control loops have to share some resource, which is often done by time-division-multiplexing, i.e. using some time slots for one loop and some other for another loop. When the shared media is a computer network, it is called a networked control system (NCS) and has been investigated in a number of recent papers [5, 9]. A special (and interesting) case is control over a wireless network environment such as Bluetooth [3], since the 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 Figure 1.

The scheduling, i.e the choice of control and measurements sequences, is normally optimized off-line. The possibility to use on-line information in the scheduling algorithms, such as local information about signal values, has also been suggested recently (see [1]). Such on-line scheduling is more complicated and will not be studied here.

Off-line scheduling of linear control systems under quadratic criteria has been treated recently in [7, 8], where a separation property between control and estimation is presented. Also [2, 4, 6] have similar set-ups. These references, however, do not present any efficient solving methods for finding a good sequence 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 paper is to present a tree search method that seems to decrease this complexity drastically. The reason for the improvements is believed to be that the tree pruning is closely tuned to the optimization criterion.

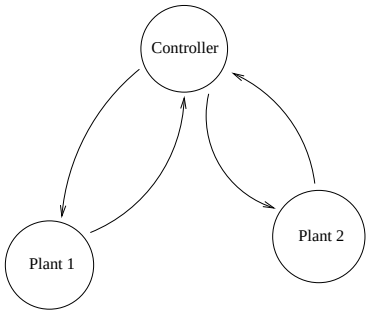

Figure 1: A simple problem. The controller can only access one plant each time slot. Which sequence gives the best expected cost?

2 Problem Formulation

The problems we are interested in can be formulated as finding the best switching sequence for discrete-time linear systems with respect to a quadratic cost function, given a time-varying linear system

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

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 $G(n)$ can be chosen by the controller in each step from a small set of M alternative systems $\{(\Phi_i, \Gamma_i, G_i)\}, i = 1, \ldots, M$.

For example, the problem of controlling one of several plants (A_i, B_i, G_i) in each time slot can be formulated

in this context by defining

$$\Phi = \begin{bmatrix} A_1 & \cdots & 0 \\ \vdots & \ddots & 0 \\ 0 & 0 & A_M \end{bmatrix}, \quad \Gamma_i = \begin{bmatrix} 0 \\ B_i \\ 0 \end{bmatrix}$$

$$G = \begin{bmatrix} G_1 & \cdots & 0 \\ \vdots & \ddots & 0 \\ 0 & 0 & G_M \end{bmatrix}.$$

The problem is to find the linear feedback law $u(n) = -L(n)z(n)$ 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

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

where $\mathbf{E}\{z(0)\} = 0$ and $\mathbf{E}\{z(0)z(0)^T\} = P_z(0)$.

3 Finding the Optimal Sequence

We will find the optimal scheduling sequence and control law by doing backwards recursion of the cost combined with tree pruning. The optimization to find the best expected cost is done off-line, so no feedback information is used in optimization of the scheduling.

3.1 Cost Representation and Feedback Gain

For a fixed choice of $K(0,N)$, the problem is a standard time-varying quadratic control problem. Therefore the best achievable cost can be written as

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

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

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

and

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

3.2 Finding a Candidate Sequence

Finding the optimal sequence is, as mentioned before, done by backwards iteration. At each step, the tree is first expanded by all new possible sequence choices (see Figure 2), and then pruned (branches are removed) using the algorithm below. The set $\kappa_{\text{cand}}(n+1,N)$ of possible control sequences from time $n+1$ to N is expanded by

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

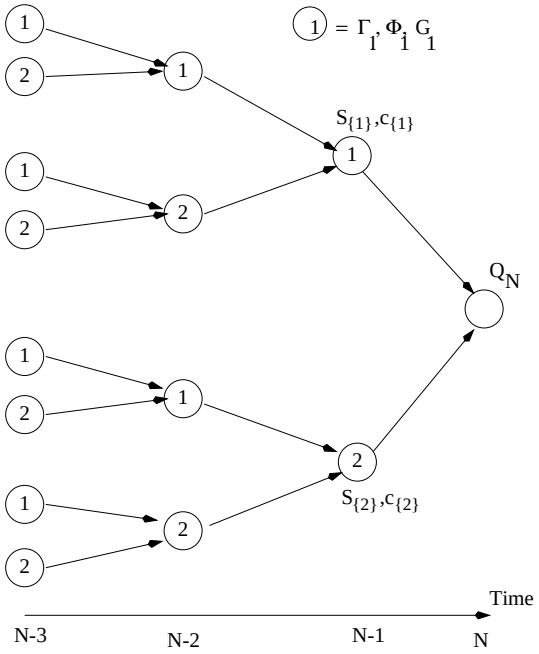

Figure 2: The control sequence tree for $M = 2$ when expanding all possibilities from $N-3$ to N.

The algorithm has one tuning parameter, $R > 0$, which must be chosen by hand. A higher R will result in a larger tree, but a too small R may not give the optimal solution (as is seen when trying to prove optimality using the method in the next section). The pruning algorithm is based on the optimality proving method, and therefore it may not be immediately intuitive (an explanation is given below). The idea of the algorithm is to make sure that pruned sequences would have resulted in a higher cost than those remaining after the pruning. If this holds for every pruned sequence, the found candidate is optimal, as shown in the next section.

The remaining sequences are kept in the set $\kappa_{\text{cand}}(n,N)$, and the removed are in $\kappa_{\text{prune}}(n,N)$. $M(n,N)$ is called *motivation data*, and contains data on a pruned sequence and on the sequence which was judged better. This data is used in the next section to prove optimality of the found sequence. A sequence which has been used to prune another sequence cannot itself be pruned (this makes the later proof simpler). See also Figure 3.

The pruning algorithm is as follows:

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 $\exists K_{\text{cand}} \in \kappa_{\text{expand}}(n,N) \cup \kappa_{\text{cand}}(n,N)$ and $\alpha \geq 0$ such that

$$\begin{cases} \alpha = \max\{\lambda \mid (S_{K_{\text{prune}}} - Q_N) \geq \lambda(S_{K_{\text{cand}}} - Q_N)\} \\ c_{\text{prune}} - c_{\text{cand}} - (1-\alpha)(c_{\min} + R - c_{\text{cand}}) \geq 0 \end{cases}$$

   then K_{prune} is pruned, i.e.

   - Move K_{prune} from the set $\kappa_{\text{expand}}(n,N)$ to the set $\kappa_{\text{prune}}(n,N)$.
   - Let $\kappa_{\text{cand}}(n,N) = \kappa_{\text{cand}}(n,N) \cup \{K_{\text{cand}}\}$.
   - 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.

In English, the algorithm removes a sequence that has a higher fixed cost c and has a cost matrix S which is close to the cost matrix of some other sequence (i.e. has a high α). If $\alpha \geq 1$, then the pruned sequence is worse than the candidate sequence for all states and therefore not optimal. The normal case though, is $\alpha \leq 1$, which means that the pruned sequence cost can be underestimated using the candidate sequence cost – the exact procedure is explained in the next section. The parameter R has the role of overestimating $V(0,N) - (V(0,n) + V(0,N-n))$, i.e. the cost gain from adding the optimal costs of two shorter problems to solving the whole problem.

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

$$K_{\text{cand}}(0,N) = \underset{K\in\kappa_{\text{cand}}(0,N)}{\text{argmin}} \left(\mathbf{E}(x(0)^T S_K x(0)) + c_K\right) = \underset{K\in\kappa_{\text{cand}}(0,N)}{\text{argmin}} \left(\mathbf{tr}(P_z(0) S_K) + c_K\right) \quad (7)$$

with cost

$$V_{\text{cand}} = V(P_z(0), L_{\text{cand}}(\cdot), K_{\text{cand}}(0,N)). \quad (8)$$

3.3 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 is now 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. We start with a lemma:

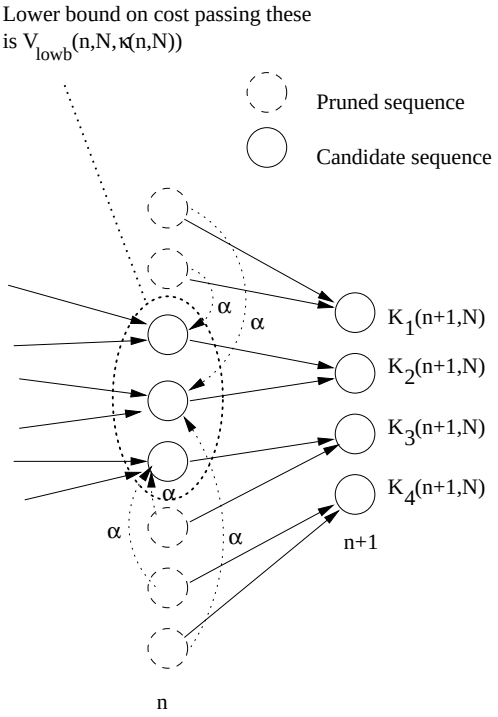

Figure 3: An example of a part of a sequence tree. The pruned sequences all have a remaining sequence to which they are compared. The proof of optimality of a sequence is based on underestimating the possible costs passing the pruned sequences.

Lemma 1 *Let* $M(n,N) = \{(c_{\text{prune}}, \alpha, c_{\text{cand}})_i,\ i \in [1..]\}$ *be the motivation set from the candidate finding algorithm above. Given* $V_{\text{lowb}}(0,N,\kappa_{\text{cand}}(n,N))$ *such that*

$$\min_{K(0,n),K(n,N)\in\kappa_{\text{cand}}(n,N),L} V(P_z(0), L, K(0,n) \oplus K(n,N)) \geq V_{\text{lowb}}(0,N,\kappa_{\text{cand}}(n,N)), \quad (9)$$

i.e. such that $V_{\text{lowb}}(0,N,\kappa_{\text{cand}}(n,N))$ *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 $\kappa_{\text{cand}}(n,N)$*). We also have* $V_{\text{lowb}}(0,n)$ *(a lower bound on the optimal cost in the length-n-problem) given. Then*

$$V_{\text{lowb}}(0,N,\kappa_{\text{cand}}(n,N) \cup \kappa_{\text{prune}}(n,N)) = \min_{(c_{\text{prune}},\alpha,c_{\text{cand}})\in M'(n,N)} (1-\alpha) V_{\text{lowb}}(0,n) + \alpha(V_{\text{lowb}}(0,N,\kappa_{\text{cand}}(n,N)) - c_{\text{cand}}) + c_{\text{prune}} \quad (10)$$

where $M'(n,N) = M(n,N) \cup \{(0,1,0)\}$ *to include the current lower bound. Now* $V_{\text{lowb}}(0,N,\kappa_{\text{cand}}(n,N) \cup$

$\kappa_{\text{prune}}(n,N))$ *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* $\kappa_{\text{cand}}(n,N) \cup \kappa_{\text{prune}}(n,N)$.

For notational convenience we put $S_0^{n-1} = \sum_{i=0}^{n-1} \begin{bmatrix} z(i) \\ u(i) \end{bmatrix}^T Q \begin{bmatrix} z(i) \\ u(i) \end{bmatrix}$ in what follows

Proof: First, we expand the first part of (9) as

$$\min_{K(0,n),K(n,N)\in\kappa_{\text{cand}}(n,N),L} V(P_z(0),L,K(0,n)\oplus K(n,N)) =$$
$$\min_{\dots} \mathbf{E}_v \left\{ S_0^{n-1} + z(n)^T S_{K(n,N)} z(n) + c_{K(n,N)} \right\} \geq$$
$$V_{\text{lowb}}(0,N,\kappa_{\text{cand}}(n,N)) \quad (11)$$
$$\Rightarrow \min_{K(0,n),L} \mathbf{E}_v \left\{ S_0^{n-1} + z(n)^T S_{K(n,N)} z(n) \right\} \geq$$
$$V_{\text{lowb}}(0,N,\kappa_{\text{cand}}(n,N)) - c_{K(n,N)} \quad (12)$$

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

$$S_{K_{\text{prune}}(n,N)} - Q_N \geq \alpha(S_{K_{\text{cand}}(n,N)} - Q_N) \quad (13)$$

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

$$\min_{K(0,n),L} \mathbf{E}_v \left\{ S_0^{n-1} + z(n)^T S_{K_{\text{prune}}(n,N)} z(n) \right\} \geq$$
$$\min_{K(0,n),L} \mathbf{E}_v \Big\{ S_0^{n-1} +$$
$$z(n)^T Q_N z(n) + \alpha z(n)^T (S_{K_{\text{cand}}(n,N)} - Q_N) z(n) \Big\} \geq$$
$$\alpha \min_{K(0,n),L} \mathbf{E}_v \left\{ S_0^{n-1} + z(n)^T S_{K_{\text{cand}}(n,N)} z(n) \right\} +$$
$$(1-\alpha) \min_{K(0,n),L} \mathbf{E}_v \left\{ S_0^{n-1} + z(n)^T Q_N z(n) \right\} \geq$$
$$\alpha(V_{\text{lowb}}(0,N,\kappa_{\text{cand}}(n,N)) - c_{K_{\text{cand}}(n,N)})$$
$$+(1-\alpha)V_{\text{lowb}}(0,n). \quad (14)$$

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 $\kappa_{\text{cand}}(n,N)$:

$$\min_{K(0,n),L} V(P_z(0),L,K(0,n)\oplus K_{\text{prune}}(n,N)) =$$
$$\min_{K(0,n),L} \mathbf{E}_v \left\{ S_0^{n-1} + z(n)^T S_{K_{\text{prune}}(n,N)} z(n) \right\} + c_{K_{\text{prune}}(n,N)} \geq$$
$$\alpha(V_{\text{lowb}}(0,N,\kappa_{\text{cand}}(n,N)) - c_{K_{\text{cand}}(n,N)}) +$$
$$(1-\alpha)V_{\text{lowb}}(0,n) + c_{K_{\text{prune}}(n,N)}. \quad (15)$$

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 (10). ∎

Theorem 1 *Given the candidate sequence* $K_{\text{cand}}(0,N)$ *from equation (7), lower bounds* $V_{\text{lowb}}(0,i)$ *for the length-i-problems,* $i \in [0..N-1]$, *and the pruned sequence motivations* $M(i,N)$ $i \in [0\dots N]$, *a lower bound* $V_{\text{lowb}}(0,N)$ *on the optimal cost can be found by iterating equation (10) from* $n=1$ *to* $n=N$.

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

Proof: Since all sequences $K(n,N) \in \kappa_{\text{cand}}(n,N) \cup \kappa_{\text{prune}}(n,N)$ pass $\kappa_{\text{cand}}(n+1,N)$, it holds that

$$V_{\text{lowb}}(0,N,\kappa_{\text{cand}}(n,N) \cup \kappa_{\text{prune}}(n,N)) =$$
$$V_{\text{lowb}}(0,N,\kappa_{\text{cand}}(n+1,N)), \quad (16)$$

which is used in the iteration. Let $V_{\text{lowb}}(0,N,\kappa_{\text{cand}}(0,N)) = V_{\text{cand}}$. Equation (16) and Lemma 1 gives $V_{\text{lowb}}(0,N,\kappa_{\text{cand}}(i,N))$, $i \in [0..N]$. A lower bound on the cost for the length-N-problem is then

$$V_{\text{lowb}}(0,N) = V_{\text{lowb}}(0,N,\kappa_{\text{cand}}(N,N)) \quad (17)$$

∎

Using Theorem 1 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 last problem length and expanded by one step for each iteration, keeping complexity low.

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 will not be found, or at least not proved using Theorem 1. A lower bound on the optimal solution is always found, though.

A special case of the problem can be proven to be NP-hard. For many problems, though, the number of sequences expanded can be kept low while still being able to prove optimality. Some examples are now presented to show the performance of the algorithm.

4 Examples

To show the feasibility of the method, some examples have been constructed. They are all based on the select-which-system-to-control problem, with different properties.

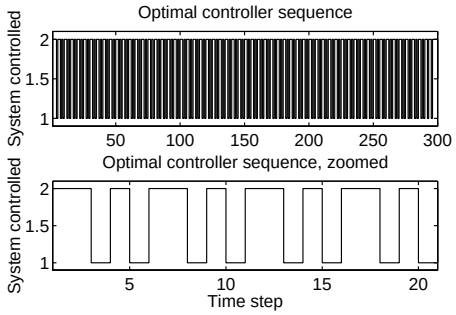

Figure 4: The optimal controller sequence for the example. Note that there are no assumptions about periodicity, but the optimal solution turns out to be periodic with period 5.

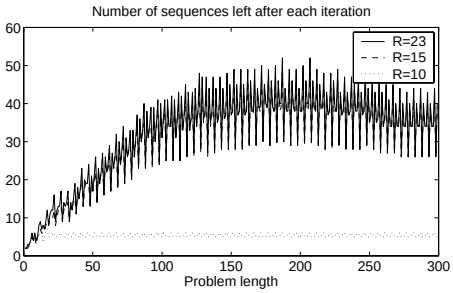

Figure 5: The number of sequences left after pruning in each iteration. After about 270 iterations, the tree size has "converged". Note that the $R = 15$ case is almost the same as for $R = 23$ and therefore not visible.

4.1 Example 1

Consider the following two simple linear systems:

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

The second order system is unstable, the the first order system is stable (but more expensive). 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 $N = 300$ produces 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 Figure 4 for the optimal sequence, and Figure 5 for the number of sequences left in each iteration of the algorithm. Figure 6 compares found costs and guaranteed lower bounds for the different choices of R. The choice of $R = 23$ is motivated in Figure 7.

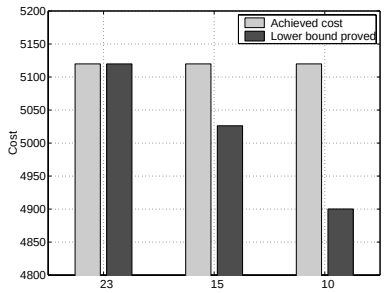

Figure 6: Found sequence cost and guaranteed lower bound when running the tree pruning algorithm four times with $R = 23$, $R = 15$, and $R = 10$, respectively. For the first two, the found sequence is identical (and proved optimal by $R = 23$), and for $R = 10$ the sequence found yields a cost which is slightly higher.

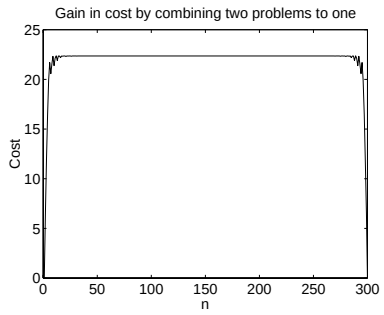

Figure 7: Since the proof is based on underestimating the remaining cost of a pruned sequence by using the solution of a shorter problem, this plot is interesting. It shows the gain from adding two shorter problems to one longer. For example, at 45 in the plot, the cost of adding a length-45 and a length-255 problem is almost 23 cheaper than the full length-300-problem.

4.2 Example 2

This example is the same as Example 1, except that the control signal of each system is held until the next control signal arrives. To model this, two extra state variables have been added to the augmented system, which is now of order 5 (we could make it 4 by reusing the held-state-variable for both systems). It turns out that the solution is easier to find and that the sequence is less complex, see Figure 8.

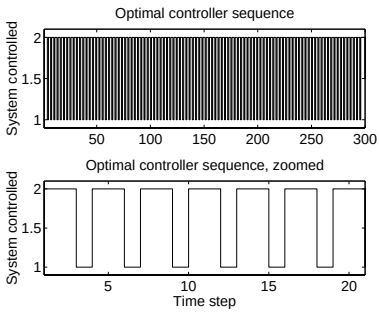

Figure 8: The optimal controller sequence for Example 2, where the control signal is held when the system is not controlled.

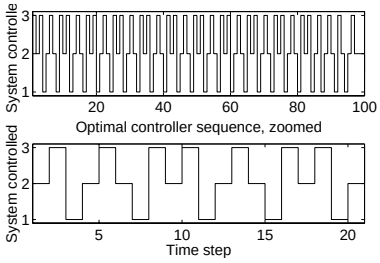

Figure 9: The optimal controller sequence for Example 3, with tree choices in each time step. The control signal is held when the system is not controlled, so the augmented system is of order 8.

4.3 Example 3

Finally, we 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 Figure 9. As can be seen in Figure 10, the search tree becomes large, but the problem is still solvable.

5 Conclusions

A method to find the optimal switching sequence in a linear-quadratic 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. It is interesting to note that the found scheduling sequences have been of low period in all our simulations (except for some initial and final transients). Future work could include formulating other problems in the same framework, such as for example choosing among distributed sensors. The problem of joint actuator scheduling and sensor scheduling is open.

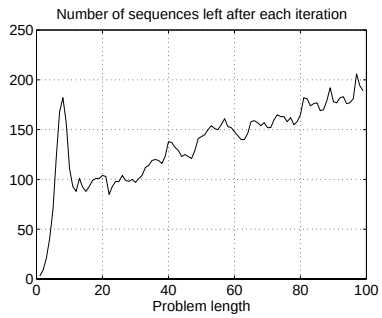

Figure 10: The number of sequences left after pruning in each iteration in Example 3. The tree becomes rather large, but it is still solvable, and a length-100-problem should give us a good insight in the steady-state behavior.

References

[1] K.-E. Årzén, A. Cervin, J. Eker, and L. Sha. "An introduction to control and real-time scheduling co-design." In Proceedings of the 39th Conference on Decision & Control, December 2000.

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

[3] J. Haartsen. "Bluetooth–the universal radio interface for ad hoc, wireless connectivity." Ericsson Review, **3**, pp. 110–117, 1998.

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

[5] H. Ishii and B. Francis. "Stabilization with control networks." In Control 2000, 2000.

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

[7] 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.

[8] 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.

[9] G. C. Walsh, H. Ye, and L. Bushnell. "Stability analysis of networked control systems." In Proceedings of the American Control Conference 1999, 1999.