Optimal State-Feedback Control Under Sparsity and Delay Constraints

Andrew Lamperski∗ Laurent Lessard∗∗

∗ Control and Dynamical Systems, California Institute of Technology, Pasadena, CA 91125 USA (e-mail: andyl@cds.caltech.edu)
∗∗ Department of Automatic Control, Lund University, Lund, Sweden (e-mail: laurent.lessard@control.liu.se)

Abstract: This paper presents the solution to a general decentralized state-feedback problem, in which the plant and controller must satisfy the same combination of delay constraints and sparsity constraints. The control problem is decomposed into independent subproblems, which are solved by dynamic programming. In special cases with only sparsity or only delay constraints, the controller reduces to existing solutions.

Keywords: decentralized control, dynamic programming, linear optimal control, state feedback, time delay.

NOTATION

\[ y_{0:t} \] Time history of \( y \): \{\( y_0, y_1, \ldots, y_t \)\}.

\[ a \rightarrow b \] There is a directed edge from node \( a \) to node \( b \).

\[ d_{ij} \] Shortest delay from node \( j \) to node \( i \).

\[ M^{sr} \] Block submatrix \((M_{ij})_{i<s,j<r}\), e.g.

\[
\begin{bmatrix}
M_{11} & M_{12} & M_{13} \\
M_{21} & M_{22} & M_{23} \\
M_{31} & M_{32} & M_{33}
\end{bmatrix}
^{(2,3);\{1\}} =
\begin{bmatrix}
M_{21} \\
M_{31}
\end{bmatrix}.
\]

1. INTRODUCTION

This paper studies a class of decentralized linear quadratic control problems. In decentralized control, inputs to a dynamic system are chosen by multiple controllers with access to different information. In this work, the dynamic system is decomposed into subsystems, each with a corresponding state and controller. A given controller has immediate access to some states, delayed access to other states, and no access to the rest. Since some controllers cannot access certain state components, the transfer matrix for the overall controller must satisfy sparsity constraints.

While decentralized synthesis is difficult in general, some classes of problems are known to be tractable and can be reduced to convex, albeit infinite dimensional, optimization problems. See Qi et al. (2004) and Rotkowitz and Lall (2006). These papers also suggest methods for solving the optimization problem. The former gives a sequence of approximate problems whose solutions converge to the global optimum, and the latter uses vectorization to convert the problem to a much larger one that is unconstrained. An efficient LMI method was also proposed by Rantzer (2006).

The class of problems studied in this paper have partially nested information constraints, which guarantees that linear optimal solutions exist, Ho and Chu (1972). Explicit state-space solutions have been reported for certain special cases. For state-feedback problems in which the controller has sparsity constraints but is delay-free, solutions were given in Shah and Parrilo (2010), and Swigart (2010). For state-feedback with delays but no sparsity, a special case of the dynamic programming argument of this paper was given by Lamperski and Doyle (2012).

In Sections 2 and 3, we state the problem and explain the state and input decomposition we will use. Main results appear in Section 4, followed by discussion and proofs.

2. PROBLEM STATEMENT

In this paper, we consider a network of discrete-time linear time-invariant dynamical systems. We will illustrate our notation and problem setup through a simple example, and then describe the problem setup in its full generality.

Example 1. Consider the state-space equations

\[
\begin{bmatrix}
x_{t+1}^1 \\
x_{t+1}^2 \\
x_{t+1}^3
\end{bmatrix}
= \begin{bmatrix}
A_{11} & A_{12} & 0 \\
A_{21} & A_{22} & 0 \\
A_{31} & A_{32} & A_{33}
\end{bmatrix}\begin{bmatrix}
x_t^1 \\
x_t^2 \\
x_t^3
\end{bmatrix}
+ \begin{bmatrix}
B_{11} & B_{12} & 0 \\
B_{21} & B_{22} & 0 \\
B_{31} & B_{32} & B_{33}
\end{bmatrix}\begin{bmatrix}
w_t^1 \\
w_t^2 \\
w_t^3
\end{bmatrix}
+ \begin{bmatrix}
w_t^1 \\
w_t^2 \\
w_t^3
\end{bmatrix}
\]

(1)

For quantities with a time dependence, we use subscripts to specify the time index, while superscripts denote subsystems or sets of subsystems. Thus, \( x_t^i \) is the state of subsystem \( i \) at time \( t \). The \( u_t^i \) are the inputs, and the \( w_t^i \) are independent Gaussian disturbances. The goal is to choose a state-feedback control policy that minimizes the standard finite-horizon quadratic cost

\[
\mathbb{E} \sum_{t=0}^{T-1} x_t^T Q_t x_t + x_T^T Q_f x_T
\]

(2)

with the usual requirement that \( R \) is positive definite, while \( Q - SR^{-1}S^T \) and \( Q_f \) are positive definite. Also, dif-
Fig. 1. Network graph for Example 1. Each node represents a subsystem, and the arrows indicate the sparsity of both the dynamical interactions (1) as well as the information constraints (3). Additionally, the labels indicate the propagation delay from one controller to another.

Fig. 2. The information graph for Example 1. The nodes of this graph are the subsets of nodes in the network graph (see Fig. 1) affected by different noises. For example, \( w^1 \) injected at node 2 affects nodes \{2, 3\} immediately, and affects \{1, 2, 3\} after one timestep.

\[ x_{t+1} = Ax_t + Bu_t + w_t \]

where \( A \) and \( B \) have sparsity patterns determined by the \( d_{ij} \). Namely, we can have \( A_{ij} \neq 0 \) whenever \( d_{ij} \in \{0, 1\} \) and similarly for \( B_{ij} \). This fact can be verified in Example 1 by comparing (1) and (4).

We assume the noise terms are Gaussian, IID for all time, and independent between subsystems. In other words, we have \( \mathbb{E}(w_{i\tau}w_{k\tau}^T) = 0 \) whenever \( \tau \neq t \) or \( i \neq k \), and \( \mathbb{E}(w_{i\tau}w_{i\tau}^T) = W_i \) for all \( t \geq 0 \).

State information propagates amongst subsystems at every timestep according to the graph topology and the delays along the links. Each controller may use any state information that has had sufficient time to reach it. More formally, \( u_i^t \) is a function of the form

\[ u_i^t = \gamma_i^t(x_{0,t-d_{i1}}, \ldots, x_{0,t-d_{iN}}). \]

Here if \( t < d_{ij} \) (e.g. when \( d_{ij} = \infty \)), then \( u_i^t \) has no access to \( x^j \) at all. As a technical point, we assume that \( G \) contains no directed cycles with zero length. If it did, we could collapse the nodes of that cycle into a single node.

The objective is to find a control policy that satisfies the information constraints (7) and minimizes the quadratic cost function (2). According to our formulation, the controller may be any function of the past information history, which grows with the size of the time horizon \( T \). However, we show in this paper how to construct an optimal controller that is linear and has a finite memory which does not depend on \( T \).

3. SPATIO-TEMPORAL DECOMPOSITION

The solution presented herein depends on a special decomposition of the states and inputs into independent components. In Subsection 3.1, we define the information graph, which describes how disturbances injected at each node propagate throughout the network. Then, in Subsection 3.2, we show how the information graph can be used to define a useful partition of the noise history. Finally, Subsection 3.3 explains the decomposition of states and inputs, which will prove crucial for our approach.

3.1 Information Graphs

The information graph a useful alternative way of representing the information flow in the network. Rather than thinking about nodes and their connections, we track the
The label sets \( \{L_s\}_{s \in U} \) partition the noise history \( H_t \):
\[
H_t = \bigcup_{s \in U} L_s \quad \text{and} \quad L_r \cap L_s = \emptyset \quad \text{whenever} \quad r \neq s
\]

Formally, we define the information graph as follows. Let \( s^i_k \) be the set of nodes reachable from node \( j \) within \( k \) steps:
\[
s^i_k = \{ i \in V : d_{ij} \leq k \}.
\]
The information graph, \( \hat{G}(U, F) \), is given by
\[
U = \{ s^i_k : k \geq 0 \}, \quad j \in V \}
\]
\[
F = \{ (s^i_k, s^j_{k+1}) : k \geq 0 \}, \quad j \in V \}.
\]
We will often write \( w^i \rightarrow s \) to indicate that \( s = s_h \), though we do not count the \( w^i \) amongst the nodes of the information graph, as a matter of convention.

The information graph can be constructed by tracking each of the \( w^i \) as they propagate through the network graph. For example, consider \( w^2 \), the noise injected into subsystem 2. In Fig. 1, we see that the nodes \( \{2, 3\} \) are affected immediately. After one timestep, the noise has reached all three nodes. This leads to the path \( w^2 \rightarrow \{2, 3\} \rightarrow \{1, 2, 3\} \) in Fig. 2. When two paths reach the same subset, we merge them into a single node. For example, the path starting from \( w^3 \) also reaches \( \{1, 2, 3\} \) after one step.

**Proposition 1.** Given an information graph \( \hat{G}(U, F) \), the following properties hold.

(i) Every node in the information graph has exactly one descendant. In other words, for every \( r \in U \), there is a unique \( s \in U \) such that \( r \to s \).

(ii) Every path eventually hits a node with a self-loop.

(iii) If \( d = \max_j \{d_{ij} : d_{ij} < \infty \} \), then an upper bound for the number of nodes in the information graph is given by \( |U| \leq N d \).

**Example 2.** Fig. 4 shows the network and information graphs for a more complex four-node network. We will refer to this example throughout the rest of the paper.

Note that the information graph may have several connected components. This happens whenever the network graph is not strongly connected. For example, Fig. 2 has two connected components because there is no path from node 3 to node 2.

Fig. 3. Noise partition diagram for Example 1 (refer to Fig. 1 and 2). Each row represents a different node of the network graph, and each column represents a different time index, flowing from left to right. For example, \( w^{2}_{t-2} \) will reach all three nodes at time \( t \), so it belongs to the label set \( \{1, 2, 3\} \).

The propagation of the noise signals \( w^i \) injected at the various nodes of the network graph. The information graph for Example 1 is shown in Fig. 2.

Formally, we define the information graph as follows. Let \( s^i_k \) be the set of nodes reachable from node \( j \) within \( k \) steps:
\[
s^i_k = \{ i \in V : d_{ij} \leq k \}.
\]
The information graph, \( \hat{G}(U, F) \), is given by
\[
U = \{ s^i_k : k \geq 0 \}, \quad j \in V \}
\]
\[
F = \{ (s^i_k, s^j_{k+1}) : k \geq 0 \}, \quad j \in V \}.
\]
We will often write \( w^i \rightarrow s \) to indicate that \( s = s_h \), though we do not count the \( w^i \) amongst the nodes of the information graph, as a matter of convention.

The information graph can be constructed by tracking each of the \( w^i \) as they propagate through the network graph. For example, consider \( w^2 \), the noise injected into subsystem 2. In Fig. 1, we see that the nodes \( \{2, 3\} \) are affected immediately. After one timestep, the noise has reached all three nodes. This leads to the path \( w^2 \rightarrow \{2, 3\} \rightarrow \{1, 2, 3\} \) in Fig. 2. When two paths reach the same subset, we merge them into a single node. For example, the path starting from \( w^3 \) also reaches \( \{1, 2, 3\} \) after one step.

**Proposition 1.** Given an information graph \( \hat{G}(U, F) \), the following properties hold.

(i) Every node in the information graph has exactly one descendant. In other words, for every \( r \in U \), there is a unique \( s \in U \) such that \( r \to s \).

(ii) Every path eventually hits a node with a self-loop.

(iii) If \( d = \max_j \{d_{ij} : d_{ij} < \infty \} \), then an upper bound for the number of nodes in the information graph is given by \( |U| \leq N d \).

**Example 2.** Fig. 4 shows the network and information graphs for a more complex four-node network. We will refer to this example throughout the rest of the paper.

Note that the information graph may have several connected components. This happens whenever the network graph is not strongly connected. For example, Fig. 2 has two connected components because there is no path from node 3 to node 2.
Proof. We proceed by induction. At \( t = 0 \), we have \( H_0 = \{x_0^1, \ldots, x_0^n\} \). For each \( i \in V \), \( s = s_{i0} \) is the unique set such that \( x_i = L_{i0} \). Thus \( \{L_{i0}s\}_{s \in U} \) partitions \( H_0 \). Now suppose that \( \{L_{is}\}_{s \in U} \) partitions \( H_t \) for some \( t \geq 0 \). By Proposition 1, for all \( r \in U \) there exists a unique \( s \in U \) such that \( r \rightarrow s \). Therefore each element \( u_i^r \in H_t \) is contained in exactly one set \( L_{it} \). Furthermore, for each
\[
i \in V, L_{it}^s \text{ is the unique label set containing } u_i^r.
\]
Therefore \( \{L_{it+1}^s\}_{s \in U} \) must partition \( H_{t+1} \).

The partition defined in Lemma 2 can be visualized using a noise partition diagram. Example 1 is shown in Fig. 3 and Example 2 is shown in Fig. 4(c). These diagrams show the partition explicitly at time \( t \) by indicating which parts of the noise history belong to which label set. Each label set is tagged with its corresponding node in the information graph.

3.3 State and Input Decomposition

We show in Appendix A that our problem setup is partially nested, which implies that there exists an optimal control policy that is linear. We also show there that \( x_t \) and \( u_t \) are linear functions of the noise history \( H_t \).

Individual components \( u_i^r \) will not depend on the full noise history \( H_t \), because certain noise signals will not have had sufficient time to travel to node \( i \). This fact can be read directly off the noise partition diagram. To see whether a noise symbol \( w_{t-k} \in L_i \) affects \( u_i^r \), we simply check whether \( i \in s \). We state this result as a lemma.

Lemma 3. The input \( u_i^r \) depends on the elements of \( L_i^s \) if and only if \( i \in s \). The state \( x_t^r \) depends on the elements of \( L_i^s \) if and only if \( i \in s \).

The noise partition described in Subsection 3.2 induces a decomposition of the input and state into components. We may write:
\[
u_t = \sum_{s \in U} I^{sr}v_t^s \quad \text{and} \quad x_t = \sum_{s \in U} I^{sr}x_t^s
\]
where \( v_t^s \) and \( x_t^s \) each depend on the elements of \( L_i^s \). Here \( I \) is a large identity matrix with block rows and columns conforming to the dimensions of \( x_t^s \) or \( u_t^s \) depending on the context. The notation \( I^{sr} \) indicates the submatrix in which the block-rows corresponding to \( r \in V \) and the block-columns corresponding to \( j \in s \) have been selected. For example, the state in Example 1 can be written as:
\[
x_t = \begin{bmatrix} I & 0 & 0 & 0 \\ 0 & I & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \end{bmatrix} \begin{bmatrix} \zeta_t^{(1)} \\ \zeta_t^{(2)} \\ \zeta_t^{(3)} \\ \zeta_t^{(1,2,3)} \end{bmatrix}
\]
The vectors \( \zeta_t^s \) each have different sizes, which is due to Lemma 3. For example, \( x_t^2 \) is only a function of \( \zeta_t^{(2,3)} \) and \( \zeta_t^{(1,2,3)} \), since these are the only label sets that contain 2.

We also use the superscript notation for other matrices. Taking (1) as an example, if \( s = \{3\} \) and \( r = \{2,3\} \), then \( A^{sr} = [A_{32} A_{33}] \).

The state equations that define \( x_t \) (5) have a counterpart in the \( \zeta \) coordinates. We state the following lemma without proof.

Lemma 4. The components \( \{\zeta_t^s\} \in U \) and \( \{\varphi_t^s\} \in U \) satisfy the recursive equations:
\[
\zeta_t^s = \sum_{w_t^r \rightarrow s} I^{sr} x_0^l
\]
\[
\zeta_{t+1}^s = \sum_{r \rightarrow s} (A^{sr} \zeta_t^r + B^{sr} \varphi_t^r) + \sum_{w_t^r \rightarrow s} I^{sr} w_t^l
\]

Two important properties of the input and state decomposition follow from results in this section. First, each \( u_t^r \) is a function of a particular subset of the information history \( H_t \), which follows directly from Lemma 3.

Corollary 5. The input \( u_t^r \) depends on the elements of \( \cup_{s \in L_i} L_i^s \). In particular, \( u_t^r \) has the ability to compute any state \( \zeta_t^s \) for which \( i \in s \).

Secondly, Lemma 2 implies that the label sets for a given time index consist of mutually independent noises. So our decomposition provides independent coordinates:

Corollary 6. Suppose \( r, s \in U \) and \( r \neq s \). Then:
\[
E\left[\begin{bmatrix} \zeta_t^r \\ \varphi_t^r \\ \zeta_t^s \\ \varphi_t^s \end{bmatrix} \right] = 0
\]

4. MAIN RESULTS

Consider the general problem setup described in Section 2, and generate the corresponding information graph \( G(U,F) \) as explained in Section 3.1. For every node \( r \in U \), define the matrices \( X_{rT} \) recursively as follows:
\[
X_{rf} = Q_{rf}
\]
\[
X_{rf} = Q_{rf} + A^{rr} X_{r+1} + A^{rr} X_{r+1} A^{rr} - (A^{rr} + A^{rr} X_{r+1} A^{rr})^{-1} (A^{rr} + A^{rr} X_{r+1} A^{rr})^T
\]
where \( s \in U \) is the unique node such that \( r \rightarrow s \). Finally, define the gain matrices \( K_{sT} \) for every \( r \in U \):
\[
K_{s} = - (A^{rr} + A^{rr} X_{r+1} A^{rr})^{-1} (A^{rr} + A^{rr} X_{r+1} A^{rr})^T
\]

Theorem 7. The optimal control policy is given by:
\[
u_t = \sum_{s \in U} I^{sr}v_t^s \text{ where } \varphi_t^s = K_{sT} \zeta_t^s
\]
where the states \( \zeta_t^s \) evolve according to (12). The corresponding optimal cost is
\[
V_0(x_0) = \sum_{i \in V} \text{trace} \left( (X_{i0}^{(1)}(t) : 1 \{x_0^i \}) \right)
\]
\[
+ \sum_{t=1}^T \text{trace} \left( (X_t^{(1)}(t) : 1 \{W_i \}) \right)
\]

Theorem 7 describes an optimal controller as a function of the disturbances \( w \). The controller can be transformed into an explicit state-feedback form as follows. For every \( s \in U \), define the subset \( s_w = \{ i : w_i \rightarrow s \} \). This set will be a singleton \( \{i\} \) when the node is a root node of the information graph, and will be empty otherwise. For example, consider Example 2 (Fig. 4(b)). When \( s = \{3,4\} \) we have \( s_w = \{4\} \), and when \( s = \{2,3,4\} \) we have \( s_w = \emptyset \). Each \( s \in U \) can be partitioned as \( s = s_w \cup \overline{s_w} \), where \( \overline{s_w} = s \setminus s_w \). The states of the controller are given by
\[
\eta_t^s = I^{tw} \zeta_t^s \quad \text{for all } s \in U \text{ satisfying } \overline{s_w} \neq \emptyset
\]
Theorem 8. Let $A_{sr}^T = A_{sr}^T + B_{sr}^T K_{r T}^T$. The optimal state-feedback controller is given by the state-space equations

$$
\begin{align*}
\eta_0 &= 0 \\
\eta_{t+1}^{(s)} &= \sum_{r \rightarrow s} I_{r \rightarrow s} A_{sr}^T I_{r \rightarrow s} \eta_t^{(r)} \\
&+ \sum_{r \rightarrow s} I_{r \rightarrow s} A_{sr}^T I_{r \rightarrow s} \eta_t^{(v)} \\
&+ \sum_{v \in U \setminus \{r\}} I_{v \rightarrow s} S_{r,v} T \eta_t^{(v)} \\
&+ \sum_{s \in V} I_{s \rightarrow r} S_{s,r} T \eta_t^{(v)}
\end{align*}
$$

where $\Gamma_{r t+1}$ is given by:

$$
J_{t+1}(\zeta_t, \varphi_{t+1}) = \sum_{s \in U} \left[ \zeta_s^T \begin{bmatrix} Q_{s,s} & S_{s,r} \end{bmatrix} \varphi_s^T + \zeta_s^T Q_f \zeta_s^T \right]
$$

where the future states $\zeta_{t+1, T}$ are defined recursively using (12). Now define the value function, which is the minimum expected cost-to-go:

$$
V_t(\zeta_t) = \min_{\varphi_t} J_t(\zeta_t, \varphi_{t+1})
$$

Note that $J_t$ is a random variable because it does not depend explicitly on $x_{t+1} T$. These future states are defined recursively using (6), and thus depend on the noise terms $w_t, T - 1$. Using the decomposition 10, the state is divided according to $x_t = \sum_{v \in U} I_{v \rightarrow s} \zeta_s^T$. We use the abridged notation $\zeta_t$ to denote $[\zeta_s^T : s \in U]$ and we use a similar notation to denote the decomposition of $u_t$ into $\varphi_t$. In these new coordinates, the cost-to-go becomes:

$$
V_t(\zeta_t) = \min_{\varphi_t} J_t(\zeta_t, \varphi_{t+1})
$$

Here, it is implied that the minimization is taken over admissible policies, namely $\varphi_t^*$ is a linear function of $L_t^*$ as explained in Section 3.3. Unlike the cost-to-go function, the value function is deterministic. The original minimum cost considered in the Introduction (2) is simply $V_0(x_0)$.

6. DISCUSSION

5. Discussion

Note that for a self-loop, $r \rightarrow r$, (13) defines a classical Riccati difference equation. Since the information graph can have at most $N$ self-loops, at most $N$ Riccati equations must be solved. For other edges, (13) propagates the solution of the self-loop Riccati equation down the information graph toward the root nodes.

The solution extends naturally to infinite horizon, provided that at the self-loops, $r \rightarrow r$, the matrices satisfy classical conditions for stabilizing solutions to the corresponding algebraic Riccati equations. Indeed, as $T \rightarrow \infty$, $X_t^T$ approach steady state values. Since all gains are computed from the self-loop Riccati equations, the gains must also approach steady state limits.

The results can also be extended to time-varying systems simply by replacing $A$, $B$, $Q$, $R$, $S$, and $W$ with $A_t$, $B_t$, $Q_t$, $R_t$, $S_t$, and $W_t$, respectively.

The results of Swigart (2010) and Shah and Parrilo (2010) on control with sparsity constraints correspond to the special case of this work in which all edges have zero delay. In this case, the information graph consists of $N$ self-loops. Thus, $N$ Riccati equations must be solved, but they are not propagated.

The results on delayed systems of Lamperski and Doyle (2012) correspond to the special case of strongly connected graphs in which all edges have a delay of one timestep. Here, all paths in the information graph eventually lead to the self-loop $V \rightarrow V$. Thus, a single Riccati equation is solved and propagated down the information graph.

6. PROOFS OF MAIN RESULTS

6.1 Proof of Theorem 7

The proof uses dynamic programming and takes advantage of the decomposition described in Section 3. Begin by defining the cost-to-go from time $t$ as a function of the current state and future inputs:

$$
J_t(x_t, u_{t:T-1}) = \sum_{\tau = t}^{T-1} x_{\tau}^T [Q R]^T [x_{\tau} R] + x_{T}^T Q_f x_{T}
$$

To solve this problem, we can use the Bellman equation:

$$
V_t(\zeta_t, \varphi_t) = \min_{u_t} \left[ \sum_{s \in U} I_{s \rightarrow r} S_{s,r} T \eta_t^{(v)} \right]
$$

where $\eta_t^{(v)}$ is the value function of the Bellman equation.

$$
J_t(\zeta_t, \varphi_t) = \min_{\varphi_t} J_t(\zeta_t, \varphi_{t+1})
$$

Here, it is implied that the minimization is taken over admissible policies, namely $\varphi_t^*$ is a linear function of $L_t^*$ as explained in Section 3.3. Unlike the cost-to-go function, the value function is deterministic. The original minimum cost considered in the Introduction (2) is simply $V_0(x_0)$.

6.2 Proof of Theorem 8

The proof uses induction proceeding backwards in time. At $t = T$, the cost-to-go is simply the terminal cost $x_T^T Q_f x_T$, and the value function is the expected value of this quantity. Decompose $x_T$ into its $\gamma_T$ coordinates. The $\zeta_T$ coordinates are mutually independent by Corollary 6, so we may write

$$
V_T(\zeta_T) = \sum_{s \in U} \left[ \zeta_s^T Q_f \zeta_s^T \right]
$$

Thus, (20) holds for $t = T$, by setting $X_T^T = Q_f$ for every $s \in U$ and $c_T = 0$. Now suppose (20) holds for $t + 1$. Equation (19) becomes:

$$
V_{t+1}(\zeta_{t+1}) = \sum_{s \in U} \left[ \zeta_s^T Q_f \zeta_s^T \right]
$$

Substitute the recursion for $\zeta_{t+1}$ defined in (12), and take advantage of the mutual independence of the $\zeta_t$ terms proved in Corollary 6. Finally, we obtain:

$$
V_t(\zeta_t) = \sum_{r \in U} \left[ \zeta_r^T \Gamma_r^{t+1} \zeta_r^T \right] + \sum_{i \in V} \text{trace} \left( (X_{t+1}^{r,i})^T (W_i) \right)
$$

where $\Gamma_{t+1}^r$ is given by:
\[ \Gamma_{t+1} = \begin{bmatrix} Q^{rr} & S^{rr} \\ S^{rt} & R^{rr} \end{bmatrix} + [A^{sr} B^{sr}]^T X_{t+1}^r [A^{sr} B^{sr}] \]

and \( s \) is the unique node in the information graph such that \( r \to s \), see Proposition 1. Equation (22) can be decomposed into independent quadratic optimization problems, one for each \( \varphi^r_s \):

\[ \varphi^r_s = \arg \min_{e^r_s} E \left[ \varphi^r_s \right] \Gamma_{t+1}^r \]

for all \( r \in U \).

The optimal cost is again a quadratic function, which verifies our inductive hypothesis.

The optimal inputs found by solving the quadratic optimization problems in Lemma 9 are given by \( \varphi^r_s = K^r_t \zeta^r_t \).

This policy is admissible by Corollary 5. Substituting the optimal policy, and comparing both sides of the equation, we obtain the desired recursion relation as well as the optimal cost (13)–(16).

6.2 Proof of Theorem 8

Consider a node \( s \in U \) with \( w^t \to s \). Thus the set \( s_w \) is nonempty, and \( s_w = (i, j) \). From (10),

\[ x^t_i = \sum_{v \supset i} I^{s_w v} \zeta^t_v = \sum_{v \supset s} I^{s_w v} \zeta^t_v \]

The second equality follows because \( v \supset i \) implies \( v \supset s \). Indeed, for any other \( j \in s \), we must have \( d_{ij} = 0 \). Thus, if \( v \supset i \), then \( v \supset j \) as well. Rearranging (23) gives

\[ I^{s_w v} \zeta^t_v = x^t_i - \sum_{v \supset s} I^{s_w v} \zeta^t_v \]

Since \( s_w \) and \( s_w \) partition \( s \), we may write \( \zeta^t_s \) as

\[ \zeta^t_s = I^{s_w t} \zeta^t_s + I^{s_w v} \zeta^t_v \]

where we substituted the definition \( \eta^t_s = I^{s_w t} \zeta^t_s \) in the final step and used the fact that \( i \in s_w \), because \( v \neq s \) and we can’t have both \( w^t \to v \) and \( w^t \to s \).

We can now find state-space equations in terms of the new \( \eta^t_s \) coordinates. The input \( u_t \) is found by substituting (24) into (15), and the \( \eta^t_s \) dynamics are found by substituting (24) into (12) and multiplying on the left by \( I^{s_w s} \). Note that the controller now depends on the state \( x_t \) rather than the disturbance \( w_t \).

REFERENCES


Appendix A. PARTIAL NESTEDNESS

In this section, we discuss partial nestedness, a concept first introduced by Ho and Chu (1972). Define the sets:

\[ T_i^j = \{ x_i^j : j \in V, 0 \leq \tau \leq t - d_{ij} \} \]

Comparing with (7), \( T_i^j \) is precisely the information that \( u_i^j \) has access to when making its decision.

Definition 10. A dynamical system (6) with information structure (7) is partially nested if for every admissible set of policies \( \gamma_i^t \), whenever \( u_i^j \) affects \( u_i^j \), then \( T_i^j \subset T_i^j \).

The main result regarding partial nestedness is given in the following lemma.

Lemma 11. (Ho and Chu (1972)). Given a partially nested structure, the optimal control for each member exists, is unique, and is linear.

In order to apply this result, we must first show that the problems considered in this paper are of the correct type.

Lemma 12. The information structure described in (7) is partially nested.

Proof. Suppose \( u_i^j \) affects \( u_i^j \) in the most direct manner possible. Namely, \( u_i^j \) directly affects \( x_{i+1}^j \), which affects a sequence of other states until it reaches \( x_{m}^j \) and \( x_{n}^j \) in \( T_i^j \).

Further suppose that \( x_{m}^j \in T_i^j \).

\[ u_i^j \text{ affects } x_{m}^j \text{ directly } \implies d_{kj} \leq \tau - \sigma \]

Adding (A.1)–(A.3) together and using the triangle inequality, we obtain \( d_{kj} \leq t - \rho \). Thus, \( x_{m}^j \in T_i^j \). It follows that \( T_i^j \subset T_i^j \), as required.

If \( u_i^j \) affects \( u_i^v \) via a more complicated path, apply the above argument to each consecutive pair of inputs along the path to obtain the chain of inclusions \( T_i^j \subset \cdots \subset T_i^j \).