

# Scalable Design of Structured Controllers using Chordal Decomposition

Yang Zheng, *Student Member, IEEE*, Richard P. Mason and Antonis Papachristodoulou, *Senior Member, IEEE*

**Abstract**—We consider the problem of designing static feedback gains subject to *a priori* structural constraints, which is a non-convex problem in general. Previous work has focused on either characterizing special structures that result into convex formulations, or employing certain techniques to allow convex relaxations of the original problem. In this paper, by exploiting the underlying sparsity properties of the problem, and using chordal decomposition, we propose a scalable algorithm to obtain structured feedback gains to stabilize a large-scale system. We first extend the chordal decomposition theorem for positive semidefinite matrices to the case of matrices with block-chordal sparsity. Then, a block-diagonal Lyapunov matrix assumption is used to convert the design of structured feedback gains into a convex problem, which inherits the sparsity pattern of the original problem. Combining these two results, we propose a sequential design method to obtain structured feedback gains clique-by-clique over a clique tree of the block-chordal matrix, which only needs local information and helps ensure privacy of model data. Several illustrative examples demonstrate the efficiency and scalability of the proposed sequential design method.

**Index Terms**—Scalable design, structured feedback gains, chordal decomposition, large-scale systems.

## I. INTRODUCTION

CONTROLLER synthesis for interconnected systems, where multiple subsystems are interacting over a network with limited communication, has received considerable attention in recent years [2]–[5]. This problem arises in several applications, such as the smart grid [6], unmanned aerial vehicles [7], and automated highways [8]. One key challenge in the case of decentralized systems is the design of structured control policies based on local information, aiming to stabilize the overall system and further minimize a certain cost function.

The general problem of designing linear feedback gains with structured constraints is NP-hard [9]. Previous approaches to synthesize decentralized controllers with information structures can be categorized into three cases: 1) finding exact solutions for special classes of structures [5], [10], [11]; 2) seeking tractable design approaches, using convex approximations [12], [13]; and 3) obtaining suboptimal solutions using non-convex optimization [14], [15]. In the first case, for the class of systems that are *quadratically invariant*, it is possible

to find optimal decentralized controllers in the frequency domain via a Youla parametrization [5], which in general results in infinite-dimensional convex programs. In another special class of structures modeled by partially ordered sets [10], Shah and Parrilo derived explicit state-space solutions by solving a number of uncoupled Riccati equations when the performance metric was the  $\mathcal{H}_2$  norm. More recently, Kim and Lall [11] presented a new factorization condition to split the overall decentralized control problem into independent subproblems, which can be explicitly solved. In the second case, the strategy is to derive a convex relaxation of the original problem, and obtain an approximate solution. For example, decentralized control can be cast as a rank-constrained semidefinite programming problem in the discrete-time and linear-quadratic setting: a convex relaxation can be obtained by dropping the rank constraint [12]. An alternative convex optimization objective is formulated in [13], which has the potential to solve problems with arbitrary structures. The third case concerns approaches which try to search for structured controllers by directly solving the original non-convex problem, using, *e.g.*, augmented Lagrangian [14] and alternating direction method of multipliers (ADMM) approaches [15]. Additionally, the notions of implementability and realizability over arbitrary graphs have been introduced in [16], where the Youla parametrization is used to characterize the set of stabilizing controllers that are implementable.

The aforementioned diverse body of research provides powerful tools for structured controller synthesis of decentralized systems. However, there is less focus on the algorithmic aspects that could make these methods practical for realistic large-scale systems, *e.g.*, systems having thousands of nodes interacting over a network. Consequently, most of the examples in the literature are relatively small-scale systems. In contrast, some practical decentralized systems, such as the electrical power grid [17] and mass transportation systems [18], could involve thousands of states and controls or more. Also, many of previous works implicitly assume there exists a central entity to collect the complete model data and perform centralized computation, which relatively ignore the privacy concerns. Here, we consider the problem of designing static state feedback gains with *a priori* structural constraints, and propose a sequential algorithm based on local model information for designing large-scale structured controllers via chordal decomposition, bringing together positive semidefinite matrices and chordal sparsity.

Chordal graphs are very well studied objects in graph theory [19], [20]: an undirected graph is chordal if every cycle of length greater than or equal to four has a chord.

Y. Zheng is supported by the Clarendon Scholarship and the Jason Hu Scholarship. R. P. Mason was supported by an EPSRC studentship under the Life Sciences Interface Doctoral Training Centre, grant EP/F500394/1. A. Papachristodoulou is funded in part by EPSRC projects EP/M002454/1, EP/J012041/1, and EP/J010537/1. A preliminary version of this paper was presented at the IEEE 55th Conference on Decision and Control [1].

Y. Zheng, R. P. Mason and A. Papachristodoulou are with the Department of Engineering Science, University of Oxford, Parks Road, Oxford OX1 3PJ, U.K. (e-mail: {yang.zheng, richard.mason, antonis}@eng.ox.ac.uk.)

Several important problems, which are hard on general graphs, can be solved in polynomial time when the graph is chordal, *e.g.*, graph colouring and finding maximal cliques [21]. Also, chordal graph theory has been widely applied to a number of fields. For example, the properties of chordal graphs were exploited to facilitate the solution of sparse linear systems via sparse Cholesky factorization in [22]. For problems in inference and machine learning, chordal graphs have been applied to maximum likelihood estimation for sparse graphical models [23], and to generalize the message passing algorithms to graphs with cycles [24]. In the field of semidefinite programming (SDP), Grone *et al.* [25] and Agler *et al.* [26] proved two important results which relate chordal graphs to sparse positive semidefinite matrices, equivalently reducing the problem of checking whether a sparse chordal matrix is positive semidefinite (or completable to a positive semidefinite matrix) to checking whether special submatrices are positive semidefinite. Moreover, Fukuda *et al.* [27] and Kim *et al.* [28] showed that the results in [25], [26] could be used to decompose the semidefinite constraints of primal and dual SDPs, respectively. These results have been recently applied to stability analysis of large-scale linear systems in [29], obtaining significantly faster solutions than using standard methods. Moreover, Andersen *et al.* have used ideas from chordal graphs to improve the efficiency of robust stability analysis of large-scale interconnected uncertain systems [30].

In this paper, we develop a scalable sequential design algorithm based on local model information to synthesize structured feedback gains for large-scale decentralized systems. We use two directed graphs to represent the networked system: a plant graph and a communication graph. This naturally results in block structured constraints in controller synthesis. This structured controller design is relaxed into a convex problem, leading to a decomposition of subsystems over the maximal cliques of the underlying graph. Our approach exploits block matrices with chordal structure to efficiently compute structured feedback gains for large-scale systems. A preliminary version of this paper was appeared in [1]. The main contributions of this paper are as follows:

- 1) We note that block-positive semidefinite matrices with chordal sparsity can be decomposed into an equivalent set of small-size block-positive semidefinite matrices with additional equality constraints. This is a direct extension of the decomposition by Agler *et al.* [26] to the case of block-chordal matrices. This result is more convenient for applications in the analysis and synthesis of networked systems, since it allows us to directly focus on the sparsity in terms of block entries (subsystems).
- 2) We apply block-chordal decomposition to design structured controllers for large-scale systems in a sequential fashion. This method first uses a block-diagonal Lyapunov matrix assumption to convert the design of structured feedback gains into a convex problem that inherits the sparsity pattern of the original problem. Then, by equally splitting the effects of overlapping subsystems in the chordal decomposition, we propose a sequential method to solve the structured feedback gains

clique-by-clique over a clique tree.

- 3) The sequential method not only greatly reduces the computational effort, making it scalable to large-scale systems, but also only needs the information of local network models, which helps preserve the model data privacy. This means the subsystems only need to share their dynamic models with their direct neighbours in a clique tree, not globally or centrally. The price of these advantages is that working with only local network models certainly introduces conservatism. We provide discussions on how to reduce the conservatism, and present possible extensions to some synthesis problems with performance guarantees.

The rest of this paper is organized as follows. In Section II we introduce the necessary background on chordal graphs, sparsity structures and large-scale systems over graphs. The results on the decomposition of block-chordal matrices are presented in Section III. In Section IV, we introduce a simple assumption to convert the design of structured feedback gains into a convex problem. Then, we propose a scalable sequential design algorithm to solve structured feedback gains in Section V. Several illustrative examples are presented in Section VII. We conclude the paper in Section VIII.

*Notation:*  $\mathbb{R}$  denotes the set of real numbers. The set of  $m \times n$  real matrices is denoted by  $\mathbb{R}^{m \times n}$ , and the set of symmetric matrices of order  $n$  is denoted by  $\mathbb{S}^n$ . Given a symmetric matrix  $X \in \mathbb{S}^n$ ,  $X \succ (\succeq) 0$  means that the matrix is positive (semi-) definite. The relation  $X_1 \succ (\text{or } \succeq) X_2$  for symmetric matrices means that  $X_1 - X_2 \succ (\text{or } \succeq) 0$ . The transpose of matrix  $C$  is denoted by  $C^T$ . Let  $C \in \mathbb{R}^{m \times n}$  and  $D \in \mathbb{R}^{p \times q}$ ; then  $C \otimes D$  denotes the Kronecker product of  $C$  and  $D$ . For two matrices of the same dimension,  $C, D \in \mathbb{R}^{m \times n}$ , we denote the Hadamard product as  $C \circ D$ , where each element is defined as  $(C \circ D)_{ij} = C_{ij}D_{ij}$ . In this paper,  $I(0)$  denotes an identity matrix (a zero matrix) of dimension compatible and clear from the context.  $\text{diag}(C_1, \dots, C_N)$  is a block diagonal matrix with blocks  $C_i$  on its main diagonal.

## II. PRELIMINARIES AND PROBLEM STATEMENT

We begin this section with a brief introduction on chordal graphs and sparsity structures. For a more comprehensive treatment, please refer to [19], [20]. The last part of this section presents the problem statement.

### A. Chordal Graphs

A directed graph  $\mathcal{G}$  is represented by a set of  $N$  vertices  $\mathcal{V} = \{1, 2, \dots, N\}$  and a set of directed edges  $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ : there is a directed edge from  $i$  to  $j$  if  $(i, j) \in \mathcal{E}$ . It is assumed that graph  $\mathcal{G}$  has no self-loops, *i.e.*,  $(i, i) \notin \mathcal{E}$ . For each vertex  $i \in \mathcal{V}$ , we define the set of its neighbours as  $\mathbb{N}_i = \{j \in \mathcal{V} \mid (j, i) \in \mathcal{E}\}$ . A graph  $\mathcal{G}$  is called *undirected* if  $(i, j) \in \mathcal{E} \Leftrightarrow (j, i) \in \mathcal{E}$ . A cycle of length  $k$  in  $\mathcal{G}$  is a sequence of pairwise distinct vertices  $(v_1, v_2, \dots, v_k)$  such that  $(v_k, v_1) \in \mathcal{E}$  and  $(v_i, v_{i+1}) \in \mathcal{E}$  for  $i = 1, \dots, k-1$ . A chord is an edge joining two non-adjacent vertices in a cycle.

*Definition 1: (Chordal Graph)* An undirected graph is chordal if every cycle of length greater than or equal to 4 has a chord.

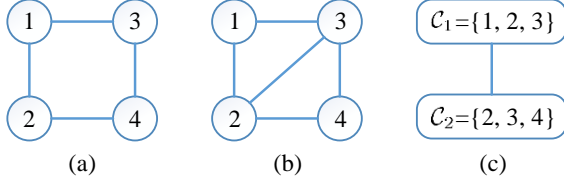


Fig. 1: Example of chordal extension and clique tree: (a) nonchordal graph, (b) chordal graph, (c) clique tree.

An immediate consequence of this definition is that all acyclic undirected graphs, undirected graphs with no cycles of length greater than three, and complete graphs are chordal. A clique of graph  $\mathcal{G}(\mathcal{V}, \mathcal{E})$  is a subset of vertices  $\mathcal{C} \subseteq \mathcal{V}$  such that  $(i, j) \in \mathcal{E}$  for any distinct vertices  $i, j \in \mathcal{C}$ , i.e., the subgraph induced by  $\mathcal{C}$  is complete. The clique is called maximal if it is not a subset of another clique. A related notion is that of a *simplicial* vertex. A vertex  $v$  of an undirected graph is called simplicial if the subgraph induced by all of its neighbours is complete.

**Proposition 1:** [20] Every chordal graph has at least one simplicial vertex.

Let  $\mathcal{G}(\mathcal{V}, \mathcal{E})$  be a connected chordal graph with set of maximal cliques  $\Gamma = \{\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_p\}$ . These cliques can be further arranged in a clique tree  $\mathcal{T} = (\Gamma, \Xi)$  with  $\Xi \subseteq \Gamma \times \Gamma$ , which satisfies the *running intersection property*, i.e.,  $\mathcal{C}_i \cap \mathcal{C}_j \subseteq \mathcal{C}_k$  if clique  $\mathcal{C}_k$  lies on the path between cliques  $\mathcal{C}_i$  and  $\mathcal{C}_j$  in the tree [19]. Note that there exist efficient algorithms to find the cliques as well as a clique tree for a chordal graph (corresponding procedures are given in Appendix A for the sake of completeness). Given a chordal graph, some maximal cliques have overlapping vertices, i.e.,  $\mathcal{C}_i \cap \mathcal{C}_j \neq \emptyset$  for some distinct cliques in  $\Gamma$ . Let  $\mathcal{C}$  be an arbitrary subset of  $\mathcal{V}$  and define a set  $J(\mathcal{C}) = \{(i, j) \in \mathcal{C} \times \mathcal{C} \mid i \leq j\}$ . Then, given a clique tree  $\mathcal{T} = (\Gamma, \Xi)$  that satisfies the running intersection property, we denote the minimal set of overlapping elements by  $\Lambda = \{(i, j, k, l) \mid (i, j) \in J(\mathcal{C}_k \cap \mathcal{C}_l), (\mathcal{C}_k, \mathcal{C}_l) \in \Xi\}$ .

Nonchordal graphs  $\mathcal{G}(\mathcal{V}, \mathcal{E})$  can be *chordal extended*, i.e., we can construct a chordal graph  $\mathcal{G}_{ex}(\mathcal{V}, \mathcal{E}')$  by adding additional edges to  $\mathcal{E}$ , such that  $\mathcal{G}_{ex}$  is chordal. Finding chordal extensions with minimal number of edges corresponds to sparse Cholesky factorization with minimum fill-ins, which is known to be NP-complete [31]. However, several heuristics, such as the minimum degree ordering followed by a symbolic Cholesky factorization, are known to generate a good chordal extension efficiently [20] (see the Appendix A).

Fig. 1 illustrates these notions. It is easy to see that the graph in Fig. 1(b) is a chordal extension of that in Fig. 1(a). In this example, there are two maximal cliques  $\mathcal{C}_1 = \{1, 2, 3\}, \mathcal{C}_2 = \{2, 3, 4\}$ , and a clique tree is shown in Fig. 1(c). For this clique tree, we have  $\Lambda = \{(2, 2, 1, 2), (2, 3, 1, 2), (3, 3, 1, 2)\}$ . Also, vertices 1 and 4 are simplicial vertices, since their neighbours  $\{2, 3\}$  induce a complete subgraph.

## B. Sparsity Structures

Since we consider decentralized systems represented by directed graphs, we need to characterize matrices with certain sparsity structures. Given a directed graph  $\mathcal{G}(\mathcal{V}, \mathcal{E})$  with no

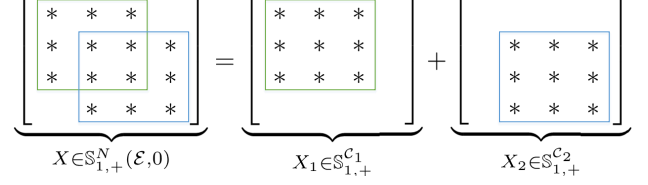


Fig. 2: Illustration of Proposition 2 for the graph shown in Fig. 1(b).  $X \in \mathbb{S}_{1,+}^N(\mathcal{E}, 0)$  can be decomposed as a sum of  $X_i$ , where  $X_i \in \mathbb{S}_{1,+}^{\mathcal{C}_i}$ ,  $i = 1, 2$ .

self-loops, we define  $\hat{\mathcal{E}} = \mathcal{E} \cup \{(i, i), i \in \mathcal{V}\}$ . The set of matrices with a sparsity structure defined by  $\mathcal{G}$  is denoted as

$$\mathbb{R}_{m,n}^N(\mathcal{E}, 0) = \{X \in \mathbb{R}^{mN \times nN} \mid X_{ij} = 0 \text{ if } (j, i) \notin \hat{\mathcal{E}},$$

where each entry  $X_{ij}$  is a block of size  $m \times n$ . If each block is square, i.e.,  $m = n$ , we simplify the notation  $\mathbb{R}_{n,n}^N(\mathcal{E}, 0)$  to  $\mathbb{R}_n^N(\mathcal{E}, 0)$ . If  $\mathcal{G}$  is undirected, we further define the following sets of symmetric (block) matrices with particular sparsity as

$$\mathbb{S}_n^N(\mathcal{E}, 0) = \{X \in \mathbb{S}_n^N \mid X_{ij} = 0 \text{ if } (j, i) \notin \hat{\mathcal{E}},$$

$$\mathbb{S}_{n,+}^N(\mathcal{E}, 0) = \{X \in \mathbb{S}_n^N(\mathcal{E}, 0) \mid X \succeq 0\},$$

$$\mathbb{S}_n^{\mathcal{C}} = \{X \in \mathbb{S}_n^N \mid X_{ij} = 0 \text{ if } (i, j) \notin \mathcal{C} \times \mathcal{C} \text{ for } \mathcal{C} \subseteq \mathcal{V},$$

$$\mathbb{S}_{n,+}^{\mathcal{C}} = \{X \in \mathbb{S}_n^{\mathcal{C}} \mid X \succeq 0\}.$$

where  $X_{ii} \in \mathbb{S}^n$ , and  $X_{ij} \in \mathbb{R}^{n \times n}$ ,  $i \neq j$ . Moreover, it will be convenient to define (block) submatrices based on the subsets of  $\mathcal{V}$ . Given a block matrix  $X \in \mathbb{R}^{mN \times nN}$  and two subsets  $\mathcal{C}_1, \mathcal{C}_2 \subseteq \mathcal{V}$ , we define

$$X(\mathcal{C}_1, \mathcal{C}_2) = \left\{ \hat{X} \in \mathbb{R}^{mN \times nN} \mid \hat{X}_{ij} = X_{ij} \text{ if } (i, j) \in \mathcal{C}_1 \times \mathcal{C}_2, \right. \\ \left. \text{otherwise, } \hat{X}_{ij} = 0 \right\}.$$

Note that  $X(\mathcal{C}_1, \mathcal{C}_2)$  contains many blocks with all entries equal to zero. To eliminate these zero blocks, we construct the submatrix  $X_{\mathcal{C}_1, \mathcal{C}_2}$  by eliminating the rows of  $X$  not in  $\mathcal{C}_1$  and the columns of  $X$  not in  $\mathcal{C}_2$ . When  $\mathcal{C}_1 = \mathcal{C}_2$ , we let  $X_{\mathcal{C}_1} = X_{\mathcal{C}_1, \mathcal{C}_1}$ . Let  $E_{ij}$  be the  $N \times N$  matrix with 1 in  $(i, j)$ th entry and 0 elsewhere. Then,  $\{E_{ij}\}$  forms a basis for  $\mathbb{R}^{N \times N}$ . Further, we observe the following identity.

$$X(\mathcal{C}_1, \mathcal{C}_2) = \sum_{(i,j) \in \mathcal{C}_1 \times \mathcal{C}_2} (E_{ij} \otimes X_{ij}), \forall \mathcal{C}_1, \mathcal{C}_2 \subseteq \mathcal{V}.$$

The notation above is a natural extension to the notation in [25]–[29], for block matrices. When each entry is a scalar, i.e.,  $n = 1$ , we have the following result first proved by Agler *et al.* [26]:

**Proposition 2:** Let  $\mathcal{G}(\mathcal{V}, \mathcal{E})$  be a chordal graph with a set of maximal cliques  $\Gamma = \{\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_p\}$ .  $X \in \mathbb{S}_1^N(\mathcal{E}, 0)$  is positive semidefinite if and only if there exists a set of matrices  $X_k \in \mathbb{S}_{1,+}^{\mathcal{C}_k}$ ,  $k = 1, \dots, p$  which decompose  $X$  as  $X = \sum_{k=1}^p X_k$ .

A direct application of this result is to decompose a large semidefinite constraint into a set of small semidefinite constraints with additional equality constraints, which can improve the efficiency of computing Newton steps for some SDP problems [28]. A dual result of Proposition 2 is a theorem by Grone *et al.* [25], which has been successfully used to decompose primal SDPs (see [27] for details).

To help illustrate some of the notation, consider the graph in Fig. 1(b) and we have the following matrices:

$$X = \begin{bmatrix} X_{11} & X_{12} & X_{13} & 0 \\ X_{21} & X_{22} & X_{23} & X_{24} \\ X_{31} & X_{32} & X_{33} & X_{34} \\ 0 & X_{42} & X_{43} & X_{44} \end{bmatrix} \in \mathbb{R}_n^N(\mathcal{E}, 0),$$

$$X(\mathcal{C}_1, \mathcal{C}_1) = \begin{bmatrix} X_{11} & X_{12} & X_{13} & 0 \\ X_{21} & X_{22} & X_{23} & 0 \\ X_{31} & X_{32} & X_{33} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Fig. 2 gives an illustration of Proposition 2 for the chordal graph shown in Fig. 1(b). There are two maximal cliques in this graph. Hence, the matrix  $X \in \mathbb{S}_{1,+}^N(\mathcal{E}, 0)$  is decomposed into two semidefinite matrices  $X_1 \in \mathbb{S}_{1,+}^{\mathcal{C}_1}$ ,  $X_2 \in \mathbb{S}_{1,+}^{\mathcal{C}_2}$ . Note that  $\Lambda = \{(2, 2, 1, 2), (2, 3, 1, 2), (3, 3, 1, 2)\}$  describes the overlapping elements between  $X_1$  and  $X_2$ .

### C. Problem Statement: Large-scale Systems over Graphs

We consider interconnected systems of heterogeneous subsystems over graphs with vertex set  $\mathcal{V}$ : each vertex in  $\mathcal{V}$  represents a subsystem and a corresponding controller. In reality, a large-scale system consists of two underlying graph structures (see the example of hierarchical systems in Fig. 3):

- a plant graph  $\mathcal{G}^p(\mathcal{V}, \mathcal{E}^p)$ , which determines the dynamic coupling of the plants;
- a communication graph  $\mathcal{G}^c(\mathcal{V}, \mathcal{E}^c)$ , which indicates the allowable communication of the controllers.

In general,  $\mathcal{G}^p$  and  $\mathcal{G}^c$  are different directed graphs. Some previous work focused on special graph structures. For example, it is assumed that  $\mathcal{G}^p$  is contained in the transitive closure of  $\mathcal{G}^c$  in [3];  $\mathcal{G}^p, \mathcal{G}^c$  shared the same graph structure in [16]. Shah and Parrilo assumed these graphs could be modelled by partial order sets [10]. Note that for dynamically decoupled plants, such as in the platoon control problem [32],  $\mathcal{G}^p$  has no edges. Also,  $\mathcal{G}^c$  would have no edges if there exists no communication between subsystems (referred as fully decentralized systems).

For each subsystem  $i \in \mathcal{V}$ , the state  $x_i(t) \in \mathbb{R}^n$  evolves according to

$$\dot{x}_i(t) = A_{ii}x_i(t) + \sum_{j \in \mathbb{N}_i^p} A_{ij}x_j(t) + B_i u_i(t),$$

where  $u_i(t) \in \mathbb{R}^m$  is the control input, and  $\mathbb{N}_i^p$  denotes the neighbours of vertex  $v_i$  in  $\mathcal{G}^p$ , i.e., those vertices that exert influence on the dynamics of vertex  $i$ . The overall state-space system is then given by

$$\dot{x}(t) = Ax(t) + Bu(t), \quad (1)$$

where  $x(t) = [x_1(t)^T, \dots, x_N(t)^T]^T$  and similarly for  $u(t)$ . We also have  $A \in \mathbb{R}_n^N(\mathcal{E}^p, 0)$ , and  $B = \text{diag}\{B_1, \dots, B_N\}$ .

Our goal is to stabilize (1) by designing the control input  $u(t)$  based on the limited communication defined by graph  $\mathcal{G}^c$ . In this paper, static state feedback is used, as in [14], [15]. Additionally, we assume communication conditions are

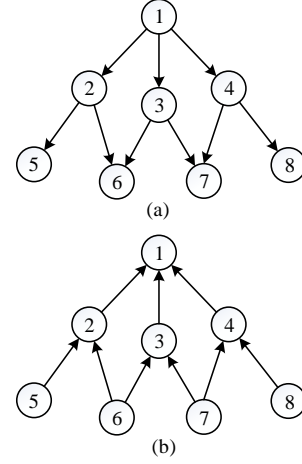


Fig. 3: Example of hierarchical systems. (a) Plant graph  $\mathcal{G}^p(\mathcal{V}, \mathcal{E}^p)$ ; here, only dynamics of subsystems in the upper layer have influence on those in the lower layer. (b) Communication graph  $\mathcal{G}^c(\mathcal{V}, \mathcal{E}^c)$ ; here, only the nodes in the upper layer can use the state information of nodes in the lower layer.

perfect, i.e., there are no time-delays or bandwidth restrictions. As a result, we are looking for controllers of the form

$$u_i(t) = k_{ii}x_i(t) + \sum_{j \in \mathbb{N}_i^c} k_{ij}x_j(t), \quad (2)$$

where  $\mathbb{N}_i^c$  denotes the neighbours of vertex  $i$  in graph  $\mathcal{G}^c$ , i.e., those vertices that send their state information to vertex  $i$ . Similarly, the compact form of the overall controller is

$$u(t) = Kx(t), \quad K \in \mathbb{R}_{m,n}^N(\mathcal{E}^c, 0), \quad (3)$$

and the closed-loop system is

$$\begin{aligned} \dot{x}(t) &= (A + BK)x(t), \\ A &\in \mathbb{R}_n^N(\mathcal{E}^p, 0), K \in \mathbb{R}_{m,n}^N(\mathcal{E}^c, 0). \end{aligned} \quad (4)$$

Concisely, the problem considered in this paper is as follows

$$\begin{aligned} &\text{Find } K \in \mathbb{R}_{m,n}^N(\mathcal{E}^c, 0), \\ &\text{such that } A + BK \text{ is asymptotically stable.} \end{aligned} \quad (5)$$

Note that without structural constraints, there exist many well-known methods to centrally synthesize the controller in (5). However, sparsity constraints arise naturally for decentralized control system design. In general, such seemingly mild and natural requirements can actually make the problem challenging [10], [14]. Some previous work imposed either special structures or used certain relaxation techniques to solve this problem, as well as to minimize a certain cost function (typically  $\mathcal{H}_2$  or  $\mathcal{H}_\infty$  norm) [5], [10]–[14]. Also, we notice that many of previous works require the complete model information of all the subsystems in a network (full  $A$  and  $B$ ), implicitly assuming the existence of a central entity.

However, the sparsity in matrices  $A, K$  have the potential to bring certain benefits from the perspective of numerical computations. The speed and accuracy of numerically computing a controller can actually be improved if this sparsity is taken advantage of. Besides, it is favourable to exploit the sparsity in graphs  $\mathcal{G}^p, \mathcal{G}^c$  such that the feedback gains can be computed locally, which helps ensure model privacy because each subsystems only need to share its dynamic model with

a small subset of other subsystems, and there is no need of a central entity that knows the overall system. In this paper, we focus on the structured stabilization problem (5), and propose a scalable sequential algorithm based on local model information for large-scale decentralized systems, by exploiting properties between the chordal graphs and sparse positive semidefinite matrices.

### III. DECOMPOSITION OF BLOCK MATRICES WITH CHORDAL STRUCTURE

The main objective of this paper is to facilitate the controller synthesis by decomposing the graphs  $\mathcal{G}^p$  and  $\mathcal{G}^c$  using chordal decomposition. Since the problem data in problem (5) is block structured in  $\mathcal{G}^p$  and  $\mathcal{G}^c$ , we state a key result which extends the results in Proposition 2 (Agler's theorem) [26] into block matrices before presenting the scalable algorithm.

**Proposition 3:** Let  $\mathcal{G}(\mathcal{V}, \mathcal{E})$  be a chordal graph with  $\Gamma = \{\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_p\}$  its set of maximal cliques. The following statements are equivalent.

- 1)  $X = [X_{ij}]_{N \times N} \in \mathbb{S}_n^N(\mathcal{E}, 0)$  is positive semidefinite, i.e.,  $X \in \mathbb{S}_{n,+}^N(\mathcal{E}, 0)$ .
- 2) There exists a set of matrices  $X_k \in \mathbb{S}_{n,+}^{C_k}$  such that

$$X = \sum_{k=1}^p X_k.$$

- 3) The system of linear matrix inequalities (LMIs)

$$X_k - L_k(z) \succeq 0, k = 1, 2, \dots, p, \quad (6)$$

is feasible, where  $X_k \in \mathbb{S}_{n,+}^{C_k}$  ( $k = 1, \dots, p$ ) satisfies  $X = \sum_{k=1}^p X_k$ ,  $z$  is a collection of matrices  $\{z_{ijkl} \in \mathbb{R}^{n \times n} \mid (i, j, k, l) \in \Lambda\}$ , and

$$L_k(z) = \sum_{(i,j,l) \mid (i,j,k,l) \in \Lambda} (E_{ij} \otimes z_{ijkl} + E_{ji} \otimes z_{ijkl}^T) - \sum_{(i,j,h) \mid (i,j,h,k) \in \Lambda} (E_{ij} \otimes z_{ijhk} + E_{ji} \otimes z_{ijhk}^T).$$

This proposition can be proved by adapting the proofs in [26], [33], [34]. Here, we briefly state two methods<sup>1</sup> to prove Proposition 3 (the detailed proof is omitted for conciseness<sup>2</sup>). One method is based on the graph interpretation for matrix  $X$  in terms of each scalar entry: we consider the case in which all of the blocks  $X_{ij}$  are dense, indicating the subgraph corresponding to each block is complete; then, the super-graph for  $X$  is also chordal with trivial cliques, meaning the decomposition follows the results of Proposition 2 [26]. Another way is to adapt the inductive proof of [33], [34] in a block-wise fashion. Note that the proof in [33], [34] follows the idea of sparse Cholesky decomposition for positive semidefinite matrices with chordal sparsity [20, Section 9], [22].

Proposition 3 does not impose any restrictions on the size of each block, i.e.,  $n$  can be any integer. In fact, the first two

statements extend the results in Proposition 2, where  $n = 1$ , to the block matrix level.

**Remark 1:** The result in Proposition 3 presents an attractive connection between chordal graphs and block-positive semidefinite matrices. This is important for decomposing semidefinite programs, and therefore improving the computational efficiency of structured feedback gains for large-scale systems. Also, this proposition allows us to solely focus on the connection of subsystems and ignore the details within each subsystem, which is more convenient for applications in the analysis and synthesis of networked systems.

We conclude this section by applying Proposition 3 to the chordal graph  $\mathcal{G}(\mathcal{V}, \mathcal{E})$  shown in Fig. 1 (b), which indicates (7) is equivalent to the feasibility of (8) or (9).

$$X = \begin{bmatrix} X_{11} & X_{12} & X_{13} & 0 \\ X_{12}^T & X_{22} & X_{23} & X_{24} \\ X_{13}^T & X_{23}^T & X_{33} & X_{34} \\ 0 & X_{24}^T & X_{34}^T & X_{44} \end{bmatrix} \in \mathbb{S}_{n,+}^N(\mathcal{E}, 0). \quad (7)$$

$$\left\{ \begin{aligned} & \begin{bmatrix} X_{11} & X_{12} & X_{13} & 0 \\ X_{12}^T & Y_1 & Y_2 & 0 \\ X_{13}^T & Y_2^T & Y_3 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \in \mathbb{S}_{n,+}^{C_1}, \\ & \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & Z_1 & Z_2 & X_{24} \\ 0 & Z_2^T & Z_3 & X_{34} \\ 0 & X_{24}^T & X_{34}^T & X_{44} \end{bmatrix} \in \mathbb{S}_{n,+}^{C_2}, \\ & Y_i + Z_i = X_{ii}, i = \{1, 3\}, Y_2 + Z_2 = X_{23} \end{aligned} \right. \quad (8)$$

$$\left\{ \begin{aligned} & \begin{bmatrix} X_{11} & X_{12} & X_{13} & 0 \\ X_{12}^T & X_{22} - z_{2212} & X_{23} - z_{2312} & 0 \\ X_{13}^T & X_{23}^T - z_{2312}^T & X_{33} - z_{3312} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \succeq 0 \\ & \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & z_{2212} & z_{2312} & X_{24} \\ 0 & z_{2312}^T & z_{3312} & X_{34} \\ 0 & X_{24}^T & X_{34}^T & X_{44} \end{bmatrix} \succeq 0 \end{aligned} \right. \quad (9)$$

### IV. DESIGN OF STRUCTURED FEEDBACK GAINS USING CONVEX RELAXATION

In this section, we present a relaxation technique to convert problem (5) into an LMI which inherits the problem's sparsity properties. By this way, the scalable design algorithm that uses chordal decomposition can be applied.

Recall that conditions for stability can be equivalently expressed as the following inequalities

$$\begin{cases} QA^T + AQ + R^T B^T + BR \prec 0 \\ RQ^{-1} \in \mathbb{R}_{m,n}^N(\mathcal{E}^c, 0), \quad Q \succ 0 \end{cases} \quad (10)$$

The steps to obtain the above condition (10) are well known, and involve the use of a Lyapunov function as

$$V(x) = x^T P x,$$

where  $P$  is a positive definite matrix of compatible dimensions;  $Q = P^{-1}$ , and  $R = KQ$ .

<sup>1</sup>The authors would like to thank one anonymous reviewer for pointing out the first method. We are also grateful to another reviewer for bringing Rf. [33] to our attention and for the encouragement of investigating sparse Cholesky decomposition.

<sup>2</sup>The interested reader can refer to a detailed proof via <http://sysos.eng.ox.ac.uk/wiki/images/e/ea/Proof.pdf>, which adapts the method in [34].

---

**Algorithm 1** Computing structured gains in a centralized way
 

---

**Input:** Matrices  $A \in \mathbb{R}_n^N(\mathcal{E}^p, 0)$ ,  $B = \text{diag}\{B_1, \dots, B_N\}$ , graphs  $\mathcal{G}^p(\mathcal{V}, \mathcal{E}^p)$ ,  $\mathcal{G}^c(\mathcal{V}, \mathcal{E}^c)$ .

**Output:** Structured feedback gains  $K \in \mathbb{R}_{m,n}^N(\mathcal{E}^c, 0)$ .

**Step 0.** Obtain the size of subsystems  $m, n$ , graph size  $N$ .

**Step 1.** Construct the basis matrices:

In (12),  $Q$  and  $R \in \mathbb{R}_{m,n}^N(\mathcal{E}^c, 0)$  have up to  $c_1 = Nn(n+1)/2$  and  $c_2 = mn|\mathcal{E}^c|$  free variables, respectively. Let  $E_1, \dots, E_{c_1}$  be basis matrices for  $Q$  and  $F_1, \dots, F_{c_2}$  be basis matrices for  $R$ . Compute basis matrices  $H_1, \dots, H_{c_1+c_2} \in \mathbb{S}^{2nN}$  as

$$H_i = \begin{bmatrix} -E_i & \\ & E_i A^T + A E_i \end{bmatrix}, i = 1, \dots, c_1.$$

$$H_{c_1+j} = \begin{bmatrix} 0 & \\ & F_j^T B^T + B F_j \end{bmatrix}, j = 1, \dots, c_2.$$

**Step 2.** Convert the problem into a standard SDP form.

We may reformulate (12) as

$$\begin{aligned} & \text{maximize} && b^T y \\ & \text{subject to} && \sum_{i=1}^{c_1+c_2} y_i H_i + Z = H_0 \\ & && Z \succeq 0 \end{aligned} \quad (13)$$

where  $y \in \mathbb{R}^{c_1+c_2}$ ,  $b = 0$ , and  $H_0 = -\varepsilon I$ ,  $\varepsilon > 0$ . The dual SDP to (13) is

$$\begin{aligned} & \text{minimize} && H_0 \bullet X \\ & \text{subject to} && H_i \bullet X = b_i, i = 1, \dots, c_1 + c_2 \\ & && X \succeq 0 \end{aligned} \quad (14)$$

where  $H_i \bullet X = \text{Tr}(H_i^T X)$ . Note (14) and (13) are standard primal and dual SDPs.

**Step 3.** Solve (13) to obtain vector  $y$  using SeDuMi [35].

**Step 4.** Convert vector  $y$  into matrices  $Q$  and  $R$ .

**Step 5.** Return structured feedback gains  $K = RQ^{-1}$ .

---

The structural constraint of communication graph  $\mathcal{G}^c$ , which is nonlinear, can be relaxed if we assume that  $Q$  (and hence  $Q^{-1}$ ) is block diagonal with block sizes compatible to those of the subsystems, which results in the following equivalence:

$$RQ^{-1} \in \mathbb{R}_{m,n}^N(\mathcal{E}^c, 0) \Leftrightarrow R \in \mathbb{R}_{m,n}^N(\mathcal{E}^c, 0). \quad (11)$$

This assumption convexifies the problem (10) into

$$\begin{cases} QA^T + AQ + R^T B^T + BR \prec 0 \\ R \in \mathbb{R}_{m,n}^N(\mathcal{E}^c, 0) \\ Q \succ 0, Q \text{ is block diagonal} \end{cases}, \quad (12)$$

but this is still centralized. We have Algorithm 1 to solve it.

Our assumption that the closed-loop system admits a block diagonal Lyapunov function would introduce conservativeness for general large-scale systems. However, examples of large-scale systems, such as transportation networks and power systems, are positive systems, whose stability is equivalent to the existence of diagonal Lyapunov functions [36]. Tanaka and Langbort further proved that this introduces no conservatism

when computing the  $\mathcal{H}_\infty$  norm of positive systems [37]. Therefore, our assumption is practical and acceptable, but more importantly, it endows that the resulting convex problem (12) has the same sparsity pattern with (5), which allows the subsequent chordal decomposition.

*Remark 2:* The convex problem (12) can be solved to obtain structured feedback gains using general conic solvers, such as SeDuMi [35] and SDPA [38]. However, both the computational efficiency and quality of the solution will degrade for larger systems, since the size of the resulting SDP scales as  $nN$ . We notice that (12) inherits the sparsity pattern of (5). There are some techniques in interior-point methods [28], [39] and first-order methods [40], [41] that exploit chordal sparsity to improve the efficiency of solving sparse SDPs. Some existing packages are SparseCoLO [28], SMCP [39], SDPA-C [38] and CDCS [42]. Using these techniques, the efficiency of solving (12) can be improved. However, these methods require the complete model information of the overall system, implicitly assuming a central entity exists. Also, note that an efficient distributed algorithm was proposed to solve a special coupled SDP in [43]. In the next section, we establish a scalable sequential algorithm based on Proposition 3 to solve (12) locally for sparse  $\mathcal{G}^p$  and  $\mathcal{G}^c$ .

## V. THE SCALABLE SOLUTION VIA CHORDAL DECOMPOSITION

We are now ready to introduce a sequential method to obtain structured feedback gains based on the convex problem (12), which is scalable for sparse decentralized systems. First, we present a way to form a chordal characterization of system data in (12), directly leading to decomposition of the positive semidefinite constraints by applying Proposition 3. A sequential method is then derived by *a priori* equally dividing the overlapping elements in the decomposed subsystems, which is able to compute the feedback gains locally in a clique-by-clique fashion. Finally, we discuss the feasibility and complexity of the proposed sequential design method.

### A. Chordal Characterization of System Data

The matrices  $A, K$  in original problem (5) have a sparsity pattern given by the directed graphs  $\mathcal{G}^p$  and  $\mathcal{G}^c$ , respectively. However, the sparsity pattern in the Lyapunov condition (12) is one of an undirected super-graph covering both  $\mathcal{G}^p$  and  $\mathcal{G}^c$ . Due to the assumption of block diagonal  $Q$ , we have

$$AQ \in \mathbb{R}_n^N(\mathcal{E}^p, 0), BR \in \mathbb{R}_n^N(\mathcal{E}^c, 0). \quad (15)$$

To handle the symmetry in the Lyapunov condition, we introduce mirror graphs as follows.

*Definition 2: (Mirror Graph)* Let  $\mathcal{G}(\mathcal{V}, \mathcal{E})$  be a directed graph. We define  $\mathcal{E}_m$  as a set of reverse edges of  $\mathcal{G}$  obtained by reversing the order of nodes in all the pairs in  $\mathcal{E}$ . The mirror of  $\mathcal{G}$  denoted by  $\mathcal{G}_m = \mathcal{M}(\mathcal{G})$  is a directed graph in the form  $\mathcal{G}_m(\mathcal{V}, \mathcal{E}_m)$  with the same set of nodes  $\mathcal{V}$  and the set of reverse edges  $\mathcal{E}_m$ .

As an example, it is easy to see that the graphs in Fig. 3 (a) and (b) are mirror graphs of each other. Then, we have

$$QA^T \in \mathbb{R}_n^N(\mathcal{E}_m^p, 0), R^T B^T \in \mathbb{R}_n^N(\mathcal{E}_m^c, 0), \quad (16)$$

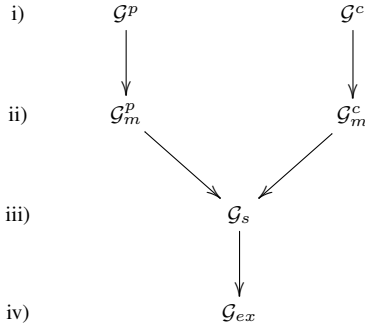


Fig. 4: Illustrative diagram for the steps of chordal characterization. i) Define  $G^p, G^c$  for plant and communication structure; ii) Get mirror graphs  $G_m^p, G_m^c$ ; iii) Define a super-graph  $G_s$  to characterise the whole structure; iv) Finally, obtain  $G_{ex}$  by making a chordal extension to  $G_s$ .

where  $G_m^p(\mathcal{V}, \mathcal{E}_m^p), G_m^c(\mathcal{V}, \mathcal{E}_m^c)$  are the mirror graphs of  $G^p$  and  $G^c$ , respectively. We further define a undirected super-graph  $G_s(\mathcal{V}, \mathcal{E}_s)$  to cover both the dynamical coupling of plants and communication connections of controllers:

$$G_s = G^p \cup G_m^p \cup G^c \cup G_m^c, \quad (17)$$

where  $\mathcal{E}_s = \mathcal{E}^p \cup \mathcal{E}_m^p \cup \mathcal{E}^c \cup \mathcal{E}_m^c$ . Combining (15) and (16), we have

$$QA^T + AQ + R^T B^T + BR \in \mathbb{S}_n^N(\mathcal{E}_s, 0).$$

Next, we construct a chordal graph  $G_{ex}(\mathcal{V}, \mathcal{E}_{ex})$  by making a chordal extension to  $G_s$ . Define a graph  $G_0(\mathcal{V}, \mathcal{E}_0)$  which only contains nodes, but no edges. Then, (12) can be rewritten into (18),

$$\begin{cases} -(QA^T + AQ + R^T B^T + BR + \varepsilon I) \in \mathbb{S}_{n,+}^N(\mathcal{E}_{ex}, 0) \\ Q - \varepsilon I \in \mathbb{S}_{n,+}^N(\mathcal{E}_0, 0) \\ R \in \mathbb{R}_{m,n}^N(\mathcal{E}^c, 0) \end{cases} \quad (18)$$

where  $\varepsilon > 0$  is a constant number. See Fig. 4 for an illustration of the above steps. For example, Fig. 5 (a) shows a chordal graph  $G_{ex}$  for the system shown in Fig. 3.

As stated in the preliminaries, it is hard to find a chordal extension with the minimal number of additional edges [31]. But there exist several simple heuristics, such as the minimum degree ordering followed by a symbolic Cholesky factorization, to efficiently generate a good approximation (see Procedure 1 in the Appendix). Note also that many systems do not need chordal extension, *e.g.*, chains, trees and banded graphs, since they are already chordal.

### B. Decomposition of the Positive Semidefinite Constraints

Having established the chordal characterization, we now turn to apply the results in Proposition 3 to decompose the positive semidefinite constraints in (18).

Let  $\Gamma = \{C_1, C_2, \dots, C_p\}$  be the set of maximal cliques in graph  $G_{ex}$ , and  $\mathcal{T} = (\Gamma, \Xi)$  with  $\Xi \subseteq \Gamma \times \Gamma$  be a clique tree that satisfies the running intersection property. The corresponding minimal set of overlapping elements is denoted by  $\Lambda$ . In (18), for notational simplicity, define

$$J_{Q,R} = -(QA^T + AQ + R^T B^T + BR + \varepsilon I).$$

Then, according to the first two statements in Proposition 3, we can equivalently reduce (18) into (19).

$$\begin{cases} \sum_{k=1}^p J_k = J_{Q,R}, \\ J_k \in \mathbb{S}_{n,+}^{C_k}, k = 1, \dots, p \\ Q - \varepsilon I \in \mathbb{S}_{n,+}^N(\mathcal{E}_0, 0) \\ R \in \mathbb{R}_{m,n}^N(\mathcal{E}^c, 0) \end{cases} \quad (19)$$

Further, according to the third statement in Proposition 3, (18) is also equivalent to (20).

$$\begin{cases} J_k - L_k(z) \succeq 0, k = 1, 2, \dots, p \\ Q - \varepsilon I \in \mathbb{S}_{n,+}^N(\mathcal{E}_0, 0) \\ R \in \mathbb{R}_{m,n}^N(\mathcal{E}^c, 0) \end{cases} \quad (20)$$

where  $J_k \in \mathbb{S}_n^{C_k}$  satisfies  $J_{Q,R} = \sum_{k=1}^p J_k$ ,  $z$  is a collection of matrices  $\{z_{ijkl} \in \mathbb{R}^{n \times n} \mid (i, j, k, l) \in \Lambda\}$ , and  $L_k(z)$  is defined in (6).

The key feature in both (19) and (20) is that they only involve a set of positive semidefinite constraints of small size (corresponding to the maximal cliques) rather than one large positive semidefinite constraint in (18). The price is that additional equality constraints are added in (19) and the set of positive semidefinite constraints is coupled in (20). These constraints and coupling can be further relaxed, resulting in the sequential design method in the next subsection. Cliques and a corresponding clique tree play central roles the upcoming sequential design. For the sake of completeness, in the Appendix, we present two efficient algorithms (*i.e.*, Maximal clique search and Maximum-weight spanning tree) to compute cliques and a clique tree, respectively (see Procedures 2 and 3).

### C. Sequential Design Method over a Clique Tree

Our sequential design method involves solving the feedback gains that only correspond to one maximal clique each time. The order of the design sequence corresponds to a clique tree that satisfies the running intersection property.

1) *Basic ideas of sequential design:* Note that the additional equality constraints in (19) and coupling term in (20) only affect the set of overlapping elements  $\Lambda$  in graph  $G_{ex}$ . If there are no elements in  $\Lambda$ , which means the maximal cliques are disjoint, then the design of structured feedback gains for a large-scale system can be naturally decomposed into several small sub-problems according to the maximal cliques.

For the case where  $\Lambda$  is non-empty, the idea to decompose (5) is that we equally split the coupling dynamic effect into several parts according to the maximal cliques that contain those overlapping elements. Essentially, we *a priori* choose the overlapping elements for the equality constraints  $\sum_{k=1}^p J_k = J_{Q,R}$  in (19) such that this constraint is satisfied. To illustrate this idea, consider the chordal graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  in Fig. 1(b). In (8), we choose

$$Y_1 = Z_1 = \frac{1}{2}X_{22}, Y_2 = Z_2 = \frac{1}{2}X_{23}, Y_3 = Z_3 = \frac{1}{2}X_{33}.$$

Then, the feasibility of (8) is reduced to the feasibility of (21) and (22). Similarly, we can specifically choose  $z$  in (9) to obtain (21) and (22).



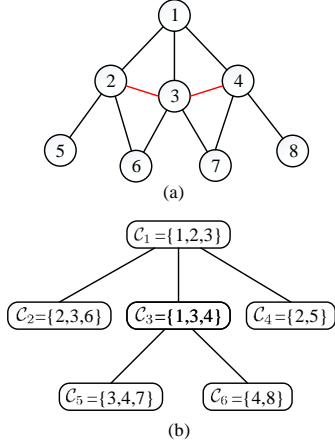


Fig. 5: Chordal extension and clique tree for the hierarchical system in Fig. 3. (a) Chordal graph  $\mathcal{G}_{ex}$ , where two undirected edges (blue ones) are added. (b) a clique tree. For the breadth-first tree traversal, we start from the root node  $\mathcal{C}_1$ , and then explore the neighbouring cliques  $\mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4$  in the second layer before moving to the next level neighbours  $\mathcal{C}_5, \mathcal{C}_6$ .

$$\begin{bmatrix} X_{11} & X_{12} & X_{13} \\ X_{12}^T & 0.5X_{22} & 0.5X_{23} \\ X_{13}^T & 0.5X_{23}^T & 0.5X_{33} \end{bmatrix} \succeq 0, \quad (21)$$

$$\begin{bmatrix} 0.5X_{22} & 0.5X_{23} & X_{24} \\ 0.5X_{23}^T & 0.5X_{33} & X_{34} \\ X_{24}^T & X_{34}^T & X_{44} \end{bmatrix} \succeq 0. \quad (22)$$

Note that for the problem of synthesizing structured controllers over graphs, each entry  $X_{ij}$  contains elements of  $Q$  and  $R$  defined in (10), which can be tuned such that (21) and (22) are satisfied. According to the clique tree in Fig. 1(c), we can first solve the clique  $\mathcal{C}_1$  (i.e., (21)), and then solve the clique  $\mathcal{C}_2$  (i.e., (22)) by embedding the corresponding parameters from  $\mathcal{C}_1$  into the overlapping elements.

#### 2) Framework for sequential design over a clique tree:

Here, we introduce a formal description of the aforementioned ideas for decentralized systems over general graphs.

##### Step 1: Obtain averaging factor for overlapping elements

Given  $\Gamma = \{\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_p\}$  as the set of maximal cliques in graph  $\mathcal{G}_{ex}$ , we define  $\gamma \in \mathbb{S}_1^N$  as the number of times nodes and edges in  $\Gamma$  are repeated, i.e.,

$$\begin{cases} \gamma_{ii} = \text{the times node } i \text{ appears in } \Gamma \\ \gamma_{ij} = \text{the times edge } (i, j) \text{ appears in } \Gamma \end{cases}.$$

It is easy to check  $\gamma \in \mathbb{S}_1^N(\mathcal{E}_{ex}, 0)$ . Correspondingly, we define an averaging factor  $\gamma' \in \mathbb{S}_1^N(\mathcal{E}_{ex}, 0)$  for graph  $\mathcal{G}_{ex}$  as

$$\begin{cases} \gamma'_{ij} = \frac{1}{\gamma_{ij}}, & \text{if } \gamma_{ij} \neq 0 \\ \gamma'_{ij} = 0, & \text{otherwise} \end{cases}.$$

Then the averaging factor for decomposing the overlapping elements is defined as  $\beta = \gamma' \otimes \mathbf{1}_{n \times n} \in \mathbb{S}_n^N(\mathcal{E}_{ex}, 0)$ , where  $\mathbf{1}_{n \times n}$  is an  $n \times n$  matrix with all entries being 1.

##### Step 2: Derive a set of LMIs over maximal cliques.

In this step, we *a priori* choose  $J_k$  in (19) as

$$J_k = J_{Q,R}(\mathcal{C}_k, \mathcal{C}_k) \circ \beta(\mathcal{C}_k, \mathcal{C}_k), k = 1, \dots, p. \quad (23)$$

Based on this construction, we have  $\sum_{k=1}^p J_k = J_{Q,R}$ . Thus, (19) is reduced into a set of small-size LMIs  $\mathcal{L}_k, k = 1, \dots, p$  over maximal cliques, where each  $\mathcal{L}_k$  is defined as

$$\mathcal{L}_k : \begin{cases} J_{Q,R}(\mathcal{C}_k, \mathcal{C}_k) \circ \beta(\mathcal{C}_k, \mathcal{C}_k) \succeq 0, \\ Q_j - \varepsilon I \succeq 0, j \in \mathcal{C}_k, \\ R(\mathcal{C}_k, \mathcal{C}_k) \in \mathbb{R}_{m,n}^N(\mathcal{E}^c, 0). \end{cases} \quad (24)$$

#### Step 3: Sequential solution over a clique tree

The dimension of  $\mathcal{L}_k$  is the size of the corresponding maximal clique. But there may exist some common design parameters among different  $\mathcal{L}_k$ . Here, we can sequentially solve them clique-by-clique over a clique tree  $\mathcal{T}$ . Starting from the root clique in  $\mathcal{T}$ , we perform a tree traversal by embedding the overlapping parameters from cliques on the layer above. Thanks to the running intersection property, the ordering of the maximal cliques suggested by  $\mathcal{T}$  guarantees that there always exist free parameters in  $\mathcal{L}_k$  when computing feedback gains sequentially. There are two major strategies for tree traversal:

- Depth-first, which starts at the root, and explores as far as possible along each branch before backtracking.
- Breadth-first, which starts at the root, and explores the neighbour nodes first before moving to the next level.

For our problem, any strategy for tree traversal with low complexity is applicable. Here, breadth-first strategy is used in our simulations. Take Fig. 5 as an example to demonstrate this strategy. We first solve the root clique  $\mathcal{C}_1 = \{1, 2, 3\}$  to get the feedback gains in nodes 1, 2, 3. Embedding these gains to the cliques in the second layer of the clique tree, i.e.,  $\mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4$ , we can get the feedback gains corresponding to nodes 6, 4 and 5, respectively. Algorithm 2 summarizes the detailed steps for the sequential design method (breadth-first strategy is used).

For the breadth-first strategy, the maximal cliques in the same layer can be computed in a parallel way in addition to sequentially. This is due to the fact that all the overlapping elements among such cliques belong to the neighbouring clique in the upper layer where the parameters have already been computed. Besides, any clique can serve as a root clique due to the structure of the clique tree, which means that this sequential design can start from any maximal clique and then explore other maximal cliques.

*Remark 3:* In the proposed sequential method, the maximal cliques (determined by the chordal extended graph  $\mathcal{G}_{ex}$ ) play an important role regarding both efficiency and feasibility. It is desirable to find a chordal extension with minimal number of added edges, where the sizes of maximal cliques are small, thus improving efficiency. On the other hand, we can iteratively merge two cliques if these two cliques share many common nodes, resulting a hierarchy of sequential solutions, which can improve the feasibility in general. Note that the merging strategy was discussed in [27], [41].

*Remark 4:* Our sequential method heavily depends on a clique tree of the chordal graph as well. The structure of clique tree may affect the feasibility of our approach due to the equal splitting strategy. Given a chordal graph, its clique tree is not unique. In principle, we can search for a clique tree with small tree depth (i.e., small number of layers), since it would reduce the iterations of message-passing. The detailed relationship



---

**Algorithm 2** Sequential design method over a clique tree
 

---

**Input:** Matrices  $A \in \mathbb{R}_n^N(\mathcal{E}^p, 0)$ ,  $B = \text{diag}\{B_1, \dots, B_N\}$ , graphs  $\mathcal{G}^p(\mathcal{V}, \mathcal{E}^p)$ ,  $\mathcal{G}^c(\mathcal{V}, \mathcal{E}^c)$ .

**Output:** Structured feedback gains  $K \in \mathbb{R}_{m,n}^N(\mathcal{E}^c, 0)$ .

**Phase 1.** Graph operation (global information).

- 1.1 Get mirror graphs  $\mathcal{G}_m^p, \mathcal{G}_m^c$ , and super-graph  $\mathcal{G}_s$ ;
- 1.2 Get a chordal graph  $\mathcal{G}_{ex}$  using Procedure 1;
- 1.3 Get a set of maximal cliques  $\Gamma = \{\mathcal{C}_1, \dots, \mathcal{C}_p\}$  and a clique tree  $\mathcal{T} = (\Gamma, \Xi)$  for  $\mathcal{G}_{ex}$  using Procedures 2 and 3;
- 1.4 Obtain the number of layers in the clique tree  $n_1$ , and Compute the averaging-factor  $\beta$ ;

**Phase 2.** Sequentially solve (24) over  $\mathcal{T}$  (local information).

**For**  $i = 1 : n_1$  **do**

**For** each maximal clique  $\mathcal{C}_k$  in  $i$ -th layer, **do**

- 1) Get dynamic matrices  $A_{\mathcal{C}_k}, B_{\mathcal{C}_k}$  of this clique;
- 2) Get basis matrices  $E_1, \dots, E_{c_1}$  for the free variables of  $Q_{\mathcal{C}_k}$ , and  $F_1, \dots, F_{c_2}$  for those of  $R_{\mathcal{C}_k}$ ; Compute basis matrices  $H_1, \dots, H_{c_1+c_2} \in \mathbb{S}^{2n|\mathcal{C}_k|}$  as

$$H_i = \begin{bmatrix} -E_i & \\ & \beta_{\mathcal{C}_k} \circ (E_i A_{\mathcal{C}_k}^T + A_{\mathcal{C}_k} E_i) \end{bmatrix}, i = 1, \dots, c_1,$$

$$H_{c_1+j} = \begin{bmatrix} 0 & \\ & \beta_{\mathcal{C}_k} \circ (B_{\mathcal{C}_k}^T F_j^T + B_{\mathcal{C}_k} F_j) \end{bmatrix}, j = 1, \dots, c_2,$$

where  $\beta_{\mathcal{C}_k}$  is the averaging factor of clique  $\mathcal{C}_k$ .

- 3) Convert the problem into a standard SDP form;

We can reformulate (24) as

$$\begin{aligned} & \text{minimize} && H_0 \bullet X \\ & \text{subject to} && H_i \bullet X = b_i, i = 1, \dots, c_1 + c_2, \\ & && X \succeq 0, \end{aligned} \quad (25)$$

where  $b_i = 0$ , and

$$H_0 = - \begin{bmatrix} -Q'_{\mathcal{C}_k} + \varepsilon I & \\ & \beta_{\mathcal{C}_k} \circ (T_1 + T_2 + \varepsilon I) \end{bmatrix}. \quad (26)$$

In (26),  $T_1, T_2$  are defined as  $T_1 = B_{\mathcal{C}_k}^T R_{\mathcal{C}_k}'^T + B_{\mathcal{C}_k} R_{\mathcal{C}_k}'$ ,  $T_2 = A_{\mathcal{C}_k} Q_{\mathcal{C}_k}' + Q_{\mathcal{C}_k}'^T A_{\mathcal{C}_k}^T$ , and  $Q_{\mathcal{C}_k}', R_{\mathcal{C}_k}'$  denote the embedding of overlapping parameters from the layer above.

- 4) Solve (25) using an SDP solver to obtain parameters  $Q_{\mathcal{C}_k}, R_{\mathcal{C}_k}$  of clique  $\mathcal{C}_k$ .

**end for**

**end for**

**Step 3.** Return structured feedback gains  $K = RQ^{-1}$ .

---

between a clique tree and the feasibility of the sequential design is beyond the scope of current work. We notice that the clique tree structure has also been used to compute search directions distributedly for SDPs in [43].

#### D. Feasibility and Complexity of the Sequential Design over a Clique Tree

We now turn to the feasibility analysis and complexity analysis of the proposed sequential design method.

1) *Feasibility analysis of the sequential design:* A necessary condition for the feasibility of the original problem (5) is that every pair  $(A_{ii}, B_i)$  is controllable, which is also sufficient

for special types of systems, such as decoupled systems, some hierarchical systems and partially nested systems [3]. In this paper, rather than explicitly restricting the types of systems, we consider the assumption that the closed-loop system admits a block diagonal Lyapunov function, *i.e.*, (12) is feasible.

As for the feasibility of the proposed sequential design, we have the following result.

*Proposition 4:* The feasibility of sequential design approach is equivalent to feasibility of the following problems.

$$(Q_{\mathcal{C}_k} A_{\mathcal{C}_k}^T + A_{\mathcal{C}_k} Q_{\mathcal{C}_k} + R_{\mathcal{C}_k}^T B_{\mathcal{C}_k}^T + B_{\mathcal{C}_k} R_{\mathcal{C}_k}) \circ \beta_{\mathcal{C}_k} \prec 0, \quad (27)$$

where  $\mathcal{C}_k, k = 1, \dots, p$  are the maximal cliques of  $\mathcal{G}_{ex}$ , and

$$\begin{aligned} A_{\mathcal{C}_k} &= \begin{bmatrix} A_p & A_{pn} \\ A_{np} & A_n \end{bmatrix}, B_{\mathcal{C}_k} = \begin{bmatrix} B_p & \\ & B_n \end{bmatrix} \\ Q_{\mathcal{C}_k} &= \begin{bmatrix} Q_e & \\ & Q_f \end{bmatrix}, R_{\mathcal{C}_k} = \begin{bmatrix} R_e & R_{f1} \\ R_{f3} & R_{f2} \end{bmatrix}. \end{aligned} \quad (28)$$

$Q_f \succeq 0, Q_f$  is block diagonal.

In (28),  $A_p, B_p$  represent the node dynamics in the previously computed clique in the upper layer ( $Q_e, R_e$  are the corresponding embedding parameters);  $A_n, B_n$  denote the node dynamics of the current layer ( $Q_f, R_{f2}$  are the corresponding free parameters); and  $A_{pn}, A_{np}$  are the coupling dynamics between these nodes ( $R_{f1}, R_{f3}$  represent the free parameters).

*Proof:* The proposed sequential design tries to solve  $\mathcal{L}_k$  in (24) sequentially according to the order suggested by a clique tree. In every step, there are some parameters from the cliques in the upper layer which need to be embedded into  $\mathcal{L}_k$ . We can see that (27) is a concise reformulation of  $\mathcal{L}_k$ , where the zero rows and columns are removed. This observation validates Proposition 4. ■

2) *Complexity analysis of the sequential design:* Here, we have the following remark.

*Remark 5:* The computational cost of solving the SDPs in Algorithm 2 scales linearly with the graph size (*i.e.*,  $\mathcal{O}(N)$ ), if the size of the largest maximal clique in  $\mathcal{G}_{ex}$  is bounded and independent of graph size. The proof of this statement is intuitive. In Algorithm 2, the size of the resulting SDPs is determined by the size of the maximal cliques. Since the size of the largest maximal clique in  $\mathcal{G}_{ex}$  is bounded and independent of the graph size, we know that the computational complexity of each small-scale problem  $\mathcal{L}_k$  in (24) is constant. Also, chordal graphs can have at most  $N$  maximal cliques [20]. Thus, the computational complexity of solving the SDP sequentially scales linearly with the graph size.

## VI. DISCUSSION AND EXTENSION OF THE SEQUENTIAL DESIGN

This section summarizes the key features of the proposed sequential design using chordal decomposition, in which both the positive sides (scalability and local computation ability) and negative sides (conservatism) are discussed. Also, we provide some extensions to consider performance guarantees.

### A. Discussion on Complexity and Local Computation Ability

In general, the proposed sequential design consists of two phases (see Algorithm 2):

- Graph operations, *i.e.*, chordal extension of the undirected super-graph  $\mathcal{G}_s$ , and searching maximal cliques and a clique tree (chordal decomposition);
- Solving the resulting SDPs (24) clique-by-clique over the clique tree.

There are two favourable features in this method: 1) low complexity, which is scalable to large sparse decentralized systems; 2) local computation ability, which helps preserve the privacy of model data for the subsystems.

1) *Complexity issue*: The graph operations (chordal extension and chordal decomposition) are well-studied in graph theory, and there exist some efficient algorithms to compute them, which are usually of low time complexity. We notice that most of them scale linearly with the number of nodes and edges in the graph (see the algorithms in the Appendix A). The most time-consuming part, therefore, lies in the process of solving the SDPs (24). For this issue, Remark 5 shows that the computational cost of solving the SDPs (24) scales linearly with the graph size when the system has a bounded size for the largest maximal clique in  $\mathcal{G}_{ex}$ . It should be noted that there are some common graphs that satisfy the assumption in Remark 5, *e.g.*,

- 1) chains (the size of the largest maximal clique is two);
- 2) trees (the size of the largest maximal clique is two);
- 3) banded graphs (the size of the largest maximal clique corresponds to the bandwidth);
- 4) certain block graphs (the size of the largest maximal clique corresponds to the size of the largest blocks).

Some engineering systems do have sparse graphs with small maximum clique size, such as platoon formation [8] and the smart grid [44]. For example, the chordal sparsity in the smart grid was exploited to reduce the complexity for semidefinite relaxations of optimal power flow problems in [44].

2) *Local model information*: In addition to reducing computation effort, another advantage of our method is the ability to preserve model data privacy for the subsystems. In our method, only the graph operations need global information, and the feedback gains can be computed locally within each clique in a sequential order. Besides, the global information only corresponds to the connectivity of dynamic coupling and communication between subsystems, and the detailed dynamic model of each subsystem can be kept private. Therefore, as opposed to previous works [12]–[15] which implicitly need the complete global model of the overall system, our method only requires the subsystems share their dynamic model within each clique locally. This feature makes our method more flexible in practice for large-scale systems, where privacy is important or global model information is hard to collect.

### B. Discussion on the Conservatism of the Sequential Design

The aforementioned benefits of sequential design come at a price, particularly due to the introduction of conservatism. There are two points that make our method conservative/restrictive compared to the original problem (5): a) the block-diagonal Lyapunov matrix assumption (see Section IV); b) the equal splitting strategy (see Section V-C).

1) *Block-diagonal Lyapunov matrix assumption*: This assumption requires that the closed-loop system admits a block-diagonal Lyapunov function. One related concept is so-called *diagonal stability* or *block diagonal stability*, which is to search for a *diagonal* (or *block diagonal*) Lyapunov function to certify stability. This is an interesting problem that attracted considerable research attention in the literature (see [45]–[47]). Typically, additional structural requirements are added to guarantee (block) diagonal stability, two of which are: 1) positive systems, where the system matrix  $A$  is Metzler; 2) hierarchical systems with a directed acyclic graph topology, where there exists a permutation matrix  $P$  such that  $P^T A P$  is a lower block-triangular matrix. The first type of system adds nonnegative constraints on certain entries, while the second type of system introduces sparsity requirements on the connections of subsystems. It is well-known that the stability of positive systems is equivalent to the existence of diagonal Lyapunov functions, which has been widely used to facilitate controller design [36]. Also, stable block-triangular system matrices admit a block-diagonal Lyapunov function [47]. These two types of systems are able to represent a wide range of engineering applications: in the former case, examples include irrigation networks, air-traffic flows and chemical networks [36], [37]; in the latter case, it includes *poset-causal* systems [10]. Hence, we argue that the block-diagonal matrix assumption is practical and acceptable (note it also results in a convex problem preserving the original sparsity pattern).

2) *Equal splitting strategy*: This idea contributes to a sequential solution, which is scalable and preserves model privacy. However, it may compromise the feasibility of (12): feasibility of the sequential design would lead to feasibility of (12), while the converse may be not true for general decentralized systems. This is mainly due to the unidirectional solving process, *i.e.*, the information is non-reversing, passing down along the clique tree. To ameliorate this problem, two possible methods are: 1) adding a backtracking process into the solving process; 2) introducing a ‘negotiating’ process for different cliques to achieve consensus for the overlapping elements. These two methods might lead to an iterative solution rather than a sequential one. Here, the alternating direction method of multipliers (ADMM) seems to offer a promising computational framework, as suggested in [40], [43]. When using ADMM, one challenge is how to preserve the local computation ability, which is an interesting topic for future work.

*Remark 6*: Despite the conservatism highlighted above, the proposed sequential method would be always feasible for two classes of systems: 1) dynamical decoupled systems and 2) systems with lower/upper triangular dynamics, if each local isolated system  $(A_{ii}, B_{2i})$  is controllable. The reason is that these systems can be stabilized by choosing an isolated gain  $K_{ii} = R_{ii}Q_i^{-1}$ , *i.e.*, by making the diagonal components  $(A_{ii} - B_{2i}K_{ii})$ ,  $i = 1, \dots, N$  stable.

### C. Extensions to Some Performance Guarantees

1) *Sparsity invariance*: When designing a static state feedback controller  $K$ , a standard change of variables  $R = KQ$  is commonly used to derive a set of LMIs. However, the

sparsity constraints  $K \in \mathbb{R}_{m,n}^N(\mathcal{E}^c, 0)$  arising in decentralized systems also results in a new nonlinear constraint  $RQ^{-1} \in \mathbb{R}_{m,n}^N(\mathcal{E}^c, 0)$ . In general, this makes it difficult to convert the sparsity constraint of  $K$  to the sparsity constraint of  $R$  and  $Q$ .

Assuming  $Q$  is block-diagonal, we have the invariance of sparsity (see (11)), indicating the sparsity of  $K$  is exactly translated to the sparsity of  $R$ . Hence, it is easy to derive LMI formulations for several synthesis problems with structural constraints, such as structured LQR,  $\mathcal{H}_2$ ,  $\mathcal{H}_\infty$  and pole clustering. Then, one key point of using our sequential design is to guarantee the resulting LMI preserves the sparsity pattern, such that chordal decomposition can still be applied. For example, given the following linear system,

$$\begin{cases} \dot{x} = Ax + B_1d + B_2u \\ z = Cx + Du \end{cases},$$

where  $d$  denotes the disturbance, and  $z$  is the performance measure, the structured  $\mathcal{H}_2$  problem is stated as

$$\begin{aligned} \min_K \quad & \|T_{dz}\|_2 \\ \text{subject to } & K \in \mathbb{R}_{m,n}^N(\mathcal{E}^c, 0) \end{aligned} \quad (29)$$

It is well-known that (29) is equivalent to

$$\begin{aligned} \min_{Q,R,W} \quad & \text{Tr}(W) \\ \text{subject to } & (AQ + B_2R) + (AQ + B_2R)^T + B_1B_1^T \prec 0, \\ & \begin{bmatrix} Q & (CQ + DR)^T \\ (CQ + DR) & W \end{bmatrix} \succ 0, \\ & Q \succ 0, RQ^{-1} \in \mathbb{R}_{m,n}^N(\mathcal{E}^c, 0). \end{aligned} \quad (30)$$

Using the block-diagonal assumption, we have the *sparsity invariance*:  $RQ^{-1} \in \mathbb{R}_{m,n}^N(\mathcal{E}^c, 0) \Leftrightarrow R \in \mathbb{R}_{m,n}^N(\mathcal{E}^c, 0)$ , which directly leads to a convex problem. To further use sequential design, one challenge is to preserve the sparsity pattern by carefully selecting the weighting matrices  $C$  and  $D$ .

2) *Sequential design with guaranteed minimum decay rate*: One straightforward extension of the sequential design method is to add guaranteed performance of minimum decay rate, using the following result:

*Lemma 1*: Suppose there is a function  $V$  and constant  $\alpha > 0$  such that  $V$  is positive definite and  $\dot{V}(x) \leq -\alpha V(x)$  for all  $x$ . Then, there is an  $M$  such that every solution of  $\dot{x} = f(x)$  satisfies  $\|x(t)\| \leq Me^{-\frac{\alpha}{2}t}\|x(0)\|$ .

Under the block-diagonal Lyapunov matrix assumption, it is easy to derive the following convex problem for the system described in (1).

$$\begin{cases} (AQ + BR) + (AQ + BR)^T + \alpha Q \prec 0 \\ R \in \mathbb{R}_{m,n}^N(\mathcal{E}^c, 0) \\ Q \succ 0, Q \text{ is block diagonal} \end{cases} \quad (31)$$

Also, this formulation has the same sparsity pattern with (12), and the proposed sequential design can be applied here. If it is feasible, the resulting controller  $K = RQ^{-1}$  guarantees the closed-loop system has a certain minimum decay rate.

In addition, to consider practical control efforts, we can add certain bounds on local feedback gains, such as

$$Q_i \succ \kappa_Q I, \begin{bmatrix} -\kappa_R I & R_{ij}^T \\ R_{ij} & -I \end{bmatrix} \prec 0, \quad (32)$$

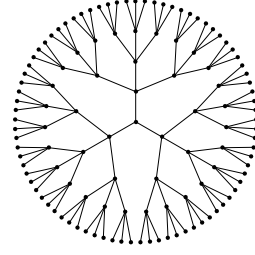


Fig. 6: Hierarchical systems over a circular tree with 4 layers and 3 branches. Each node has linear dynamics as in (33). The information flow is bottom-up but only dynamics of nodes in the upper layer have influence on those in the lower layer.

where  $\kappa_R > 0, \kappa_Q > 0$  are constant numbers. Then, we have

$$k_{ij}^T k_{ij} = Q_i^{-1} R_{ij}^T R_{ij} Q_i^{-1} \prec \frac{\kappa_R}{\kappa_Q^2} I.$$

This means the local feedback gains are all upper bounded. In Section VII-B, a network of coupled inverted pendula is used to demonstrate the extension of sequential design with guaranteed decay rate and bounded feedback gains.

## VII. ILLUSTRATIVE EXAMPLES

In this section, we present three illustrative examples to demonstrate the scalability of the proposed sequential design method<sup>3</sup>. In particular, we consider special hierarchical systems, a network of coupled inverted pendula, and general decentralized systems over directed graphs. All simulations were run on a computer with an Intel(R) Core(TM) i7 CPU, 2.8 GHz processor and 8GB of RAM. The SDPs were considered to be solved when the primal-dual gap had been reduced to less than  $10^{-8}$ .

### A. Hierarchical Systems

Consider the hierarchical system shown in Fig. 3. Here, motivated by [15], we assume each node is an unstable second order system coupled with its neighbouring nodes through an exponentially decaying function of the Euclidean distance  $\alpha(i, j)$  between them,

$$\dot{x}_i = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} x_i + \sum_{j \in \mathbb{N}_i^p} e^{-\alpha(i,j)} x_j + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_i, \quad (33)$$

where,  $\alpha(i, j)$  is chosen as  $\frac{1}{10}(i - j)^2$  in the simulations. The feedback gains are chosen in the form of (2). We first solve this problem in a centralized way using Algorithm 1, which took about 0.087 s to get the following stabilizing controller:

- node 1:  $k_{11} = [-22.3, 6.07]$ ,  $k_{12} = [-1.05, 1.09]$ ,  $k_{13} = [-0.77, 0.81]$ ,  $k_{14} = [-0.46, 0.50]$ ,
- node 2:  $k_{22} = [-9.77, 4.88]$ ,  $k_{25} = [-0.25, 0.48]$ ,  $k_{26} = [-0.09, 0.24]$ ,
- node 3:  $k_{33} = [-9.75, 4.84]$ ,  $k_{36} = [-0.23, 0.47]$ ,  $k_{37} = [-0.10, 0.23]$ ,
- node 4:  $k_{44} = [-9.64, 4.79]$ ,  $k_{47} = [-0.23, 0.46]$ ,  $k_{48} = [-0.11, 0.23]$ ,

<sup>3</sup>The numerical examples presented in this section and additional examples are available from <https://github.com/zhengy09/sdc>.

TABLE I: Computing sequences, structured gains and computing time for the hierarchical system shown in Fig. 3

Seq.	Cliques	Embedding parameters	Adjustable parameters	Computed gains	Time (s)
1	$\mathcal{C}_1$	$\emptyset$	$Q_1, Q_2, Q_3, R_{11}, R_{12}, R_{13}, R_{22}, R_{33}$	$\begin{cases} k_{11} = -[30.83 \ 7.26] \\ k_{13} = -[1.19 \ 0.98] \\ k_{33} = -[9.99 \ 6.28] \end{cases}, k_{12} = -[1.64 \ 1.30], k_{22} = -[9.05 \ 5.88]$	0.0339
2	$\mathcal{C}_2$	$Q_2, Q_3, R_{22}, R_{33}$	$Q_6, R_{66}, R_{26}, R_{36}$	$\begin{cases} k_{66} = -[6.75 \ 4.51] \\ k_{36} = -[0.06 \ 0.25] \end{cases}, k_{26} = -[0.08 \ 0.13]$	0.0307
3	$\mathcal{C}_3$	$Q_1, Q_3, R_{11}, R_{33}, R_{13}$	$Q_4, R_{44}$	$k_{44} = -[9.15 \ 5.77]$	0.0313
4	$\mathcal{C}_4$	$Q_2, R_{22}$	$Q_5, R_{55}, R_{25}$	$k_{55} = -[6.64 \ 4.41], k_{25} = -[0.12 \ 0.23]$	0.0310
5	$\mathcal{C}_5$	$Q_3, Q_4, R_{33}, R_{44}$	$Q_7, R_{77}, R_{37}, R_{47}$	$\begin{cases} k_{77} = -[6.74 \ 4.50] \\ k_{47} = -[0.04 \ 0.29] \end{cases}, k_{37} = -[0.03 \ 0.13]$	0.0316
6	$\mathcal{C}_6$	$Q_4, R_{44}$	$Q_8, R_{48}$	$k_{88} = -[6.57 \ 4.32], k_{48} = -[0.01 \ 0.15]$	0.0302

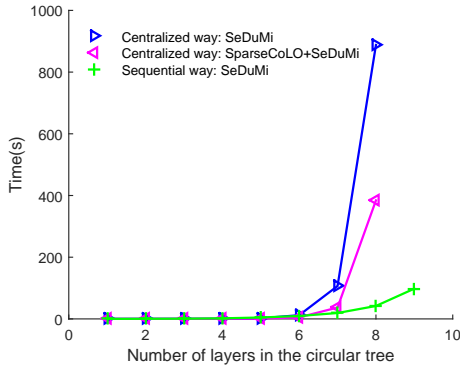


Fig. 7: Time in seconds that Algorithm 1 (called SeDuMi and SparseCoLO+SeDuMi, respectively) and Algorithm 2 (called SeDuMi) need, in order to solve for the structured feedback gains over circular trees.

- node 5:  $k_{55} = -[6.57, 4.32]$ ,
- node 6:  $k_{66} = -[6.58, 4.34]$ ,
- node 7:  $k_{77} = -[6.58, 4.34]$ ,
- node 8:  $k_{88} = -[6.55, 4.29]$ .

Then, we used the proposed sequential design method to solve this problem (using Algorithm 2). The corresponding chordal extension and clique tree are shown in Fig. 5. TABLE I lists the solving sequences, computed gains and time consumed for each clique. The total time was 0.233 s using the sequential design method. For this special small-size problem, computing the gains in a centralized way was faster than that using sequential design. However, it took less time for solving each maximal clique, as listed in TABLE I. The sequential design method is more beneficial when the system size is large.

To illustrate this point, we considered another class of hierarchical systems over a circular tree (see Fig. 6). The dynamics flow is top-down, while the information flow is bottom-up. Each node is assumed to have dynamics evolving as in (33). Fig. 7 shows the comparison between Algorithm 1 and Algorithm 2 for different number of layers (where each node has two branches). Here, we used two different ways, *i.e.*, SeDuMi and SparseCoLO+SeDuMi, to solve the resulting SDP in Algorithm 1. SparseCoLO can detect chordal sparsity in an SDP, and then call SeDuMi to solve the problem. As shown in Fig. 7, even though chordal sparsity has been exploited in SparseCoLO, our sequential method is more

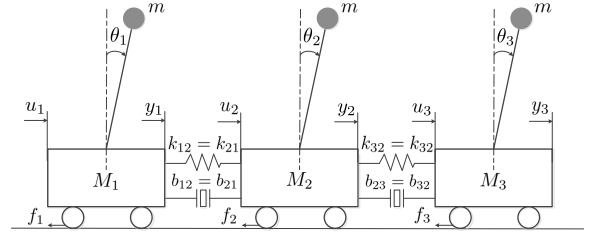


Fig. 8: A network of three coupled inverted pendula.

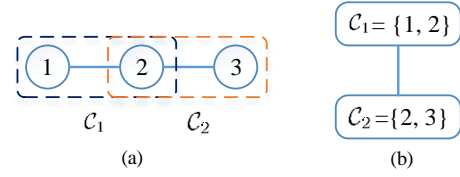


Fig. 9: Chordal decomposition of the coupled inverted pendula: (a) maximal cliques; (b) clique tree.

efficient to compute structured gains for large-scale systems.

### B. A Practical Example: Coupled Inverted Pendula

Here, we consider a practical example: a network of three coupled inverted pendula (see Fig. 8), to demonstrate the extension of sequential design with minimum decay rate and bounds on the feedback gains. The linearized dynamics around the equilibrium point can be described as [48]:

$$A_i = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{M_i+m}{M_i l} g & 0 & \frac{k_i}{M_i l} & \frac{c_i+b_i}{M_i l} \\ 0 & 0 & 0 & 1 \\ -\frac{m}{M_i} g & 0 & -\frac{k_i}{M_i} & -\frac{c_i+b_i}{M_i} \end{bmatrix},$$

$$A_{ij} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{k_{ij}}{M_i l} & \frac{b_{ij}}{M_i l} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{k_{ij}}{M_i} & \frac{b_{ij}}{M_i} \end{bmatrix}, B_i = \begin{bmatrix} 0 \\ -\frac{1}{M_i l} \\ 0 \\ \frac{1}{M_i} \end{bmatrix},$$

for  $i = 1, 2, 3; (i, j) = (1, 2), (2, 1), (2, 3), (3, 2)$ , where  $k_i = \sum_{j \in \mathbb{N}_i^p} k_{ij}, b_i = \sum_{j \in \mathbb{N}_i^p} b_{ij}$ . Here, the local state variable is  $x_i = [\theta_i, \dot{\theta}_i, y_i, \dot{y}_i]^T$ , and  $c_i, b_{ij} = b_{ji}, k_{ij} = k_{ji}$  are friction, damper and spring coefficients, respectively. We have assumed that the moment of inertia of the pendula is zero.

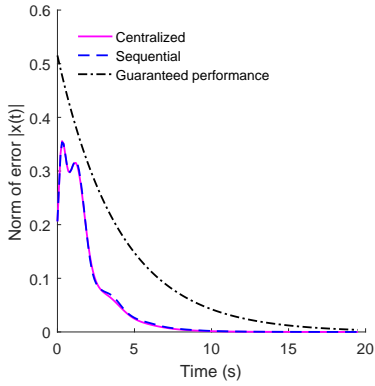


Fig. 10: Exponential decay of  $\|x(t)\|$  using the centralized computation and the sequential computation.

The plant graph for this example is a chain of three nodes. We assume that the communication graph coincides with its plant graph, indicating each node has access to its nearest neighbor's state information. Fig. 9 shows the chordal decomposition. The parameters in our simulations are  $M_1 = 0.6, M_2 = 0.8, M_3 = 1, m = 0.1, g = 10, l = 0.5, k_{12} = k_{21} = 0.2, k_{23} = k_{32} = 0.4, b_{12} = b_{21} = 0.4, b_{23} = b_{32} = 0.2, c_1 = 1, c_2 = 0.2$  and  $c_3 = 1$ . The requirement of minimum decay rate is set as  $\alpha = 0.5$  and bounds on feedback gains are  $\kappa_R = 10$  and  $\kappa_Q = 0.1$ . Then, we can compute the structured feedback gain by solving (31) centrally. This way requires all of the model information. Alternatively, we can solve (31) according to the clique tree shown in Fig. 9 (b): for clique 1, we only need the model information of node 1 and node 2, while only the models of node 2 and node 3 are required for clique 2. In our simulation, both these two cases are feasible, and Fig. 10 shows the performance of the computed structured controllers for the initial condition  $x_1(0) = [0.2, 0, 0.5, 0], x_2(0) = 0$  and  $x_3(0) = 0$ .

### C. General Decentralized Systems with Different Sizes of Largest Maximal Clique

Finally, we present simulation results for decentralized systems with general directed graphs. It is assumed that each node is an unstable second order system coupled with its neighbouring nodes, as shown in (33). In the simulations, we first generate a random chordal graph  $\mathcal{G}_1 = (\mathcal{V}, \mathcal{E}_1)$  with a bound on the size of its largest maximal clique, and then randomly remove some edges of  $\mathcal{G}_1$  to form the plant graph  $\mathcal{G}^p$ . To improve the feasibility, we ensure that the communication graph satisfies  $\mathcal{E}^p \subseteq \mathcal{E}^c \subseteq \mathcal{E}_1$ . Under these constructions,  $\mathcal{G}^p, \mathcal{G}^c$  are general directed graphs such that the largest maximal clique of  $\mathcal{G}_{ex}$  has limited size. When this is set to five, Fig. 11 shows a comparison between the performance of Algorithm 1 and Algorithm 2 for different graph sizes, which clearly shows the efficiency of our sequential method for large-scale systems.

Next, we consider 100-node graphs, and vary the size of the largest maximal clique. As shown in Fig. 12, the time consumption increases as the largest maximal clique size increases for both Algorithms 1 and 2. The efficiency of the sequential design method becomes worse as the size of

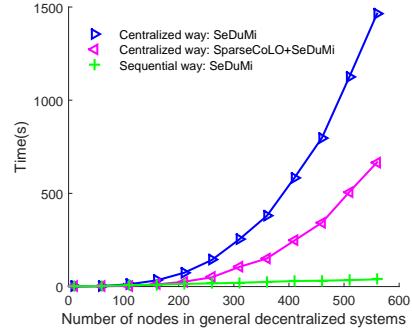


Fig. 11: Time in seconds that Algorithm 1 (called SeDuMi and SparseCoLO+SeDuMi, respectively) and Algorithm 2 (called SeDuMi) need, in order to solve a general decentralized control design problem (the size of largest maximal clique is five in the extended graph  $\mathcal{G}_{ex}$ ).

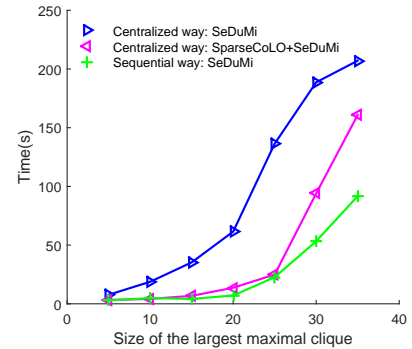


Fig. 12: Algorithm 1 (called SeDuMi and SparseCoLO+SeDuMi, respectively) versus Algorithm 2 (called SeDuMi) for general systems with 100 nodes, when varying the size of the largest maximal clique.

largest maximal clique increases, as expected. This illustrates that the efficiency of the proposed sequential design method is determined by the size of largest maximal clique in the extended graph. If this is small and independent of graph size, the sequential design method scales a lot better.

## VIII. CONCLUSION

This paper considered the synthesis of static structured feedback gains for large-scale decentralized systems over directed graphs. A sequential design method has been proposed by exploiting the chordal decomposition of block structured semidefinite matrices, which improves the scalability and helps preserve model data privacy for large-scale applications. One future work is to introduce a backtracking or negotiating process in the sequential design, which is very interesting and promising as it might not only increase the feasibility but also provide a way to compute gains that incorporate certain global constraints. Another direction of future work is to investigate how to include a performance index, *e.g.*,  $\mathcal{H}_2$  and  $\mathcal{H}_\infty$ , into the solution that preserves the sparsity pattern.

## APPENDIX A

### ALGORITHMS FOR CHORDAL GRAPHS

#### A. Chordal Extension

In practice, a chordal extension can be constructed by computing a symbolic Cholesky factorization, where the amount of fill-ins (*i.e.*, added edges) depends heavily on the ordering

---

**Procedure 1** Chordal extension
 

---

**Input:** Graph  $\mathcal{G}_s(\mathcal{V}, \mathcal{E}_s)$ .

**Output:** Chordal graph  $\mathcal{G}_{ex}(\mathcal{V}, \mathcal{E}_{ex})$  with  $\mathcal{E}_s \subseteq \mathcal{E}_{ex}$ 

1. Create a positive definite matrix  $X \in \mathbb{S}_{1,+}^N(\mathcal{E}_s, 0)$ ;
  2. Compute a minimum degree ordering  $\alpha$  of  $X$ ;
  3. Permute the rows and columns of  $X$  according to  $\alpha$ ;
  4. Perform a Cholesky factorization of  $X$  to get a factor  $L$ ;
  5. Matrix  $L + L^T$  has a chordal sparsity pattern that is a chordal extension of  $\mathcal{G}_s$ ;
  6. Return  $\mathcal{G}_{ex}$  according to the pattern of  $L + L^T$ .
- 

---

**Procedure 2** Maximal clique search
 

---

**Input:** Graph  $\mathcal{G}_{ex}(\mathcal{V}, \mathcal{E}_{ex})$ .

**Output:** A set of maximal cliques  $\Gamma = \{\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_p\}$ 

 Obtain a perfect elimination ordering  $\alpha = \{\alpha_1, \dots, \alpha_N\}$ 

 Initialization:  $\mathcal{C}_0 = \emptyset$ 
**for**  $i = 1$  to  $N$  **do**
 $\mathcal{C}_i = \{\alpha_i\} \cup \{u \in \mathbb{N}_{\alpha_i} \mid u \text{ lies behind } \alpha_i \text{ in } \alpha\}$ ;

**if**  $\mathcal{C}_i$  is not a subset of  $\mathcal{C}_0$  **then**
 $\mathcal{C}_i$  is a maximal clique

 $\mathcal{C}_0 = \mathcal{C}_i$ 
**end if**
**end for**


---

of the nodes. The problem of finding a minimum fill ordering, which corresponds to finding the minimum number of added edges, is NP-complete [31]. However, there are effective heuristics for finding the minimum chordal extension, such as minimum degree ordering and approximate minimum degree ordering [20]. The strategy of minimum degree ordering repeatedly chooses a node of minimum degree in the symbolic Cholesky factorization. Procedure 1 presents the major steps for this strategy used in the chordal extension.

---

**Procedure 3** Maximum-weight spanning tree
 

---

**Input:** A weighted Graph  $\mathcal{G}(\mathcal{V}, \mathcal{E})$ .

**Output:** Maximum-weight spanning tree  $\mathcal{G}'(\mathcal{V}', \mathcal{E}')$ 

 Initialization:  $\mathcal{V}' = \alpha$ , for some arbitrary  $\alpha \in \mathcal{V}$ 
**while**  $\mathcal{V}' \neq \mathcal{V}$  **do**

 Find an edge  $(u, v)$  with maximum weight such that  $u \in \mathcal{V}'$  and  $v \notin \mathcal{V}'$ ;

 $\mathcal{V}' = \mathcal{V}' \cup v$ ;

 $\mathcal{E}' = \mathcal{E}' \cup (u, v)$ ;

**end while**


---

**B. Maximal Cliques and Clique Tree**

One computational effective way to list the maximal cliques for a chordal graph is to exploit the properties of so-called *perfect elimination orderings* (see Procedure 2). An ordering  $\alpha = \{\alpha_1, \dots, \alpha_N\}$  of graph  $\mathcal{G}(\mathcal{V}, \mathcal{E})$  is a perfect elimination ordering if, for each  $i \in \{1, \dots, N\}$ ,  $\alpha_i$  is a simplicial vertex of the subgraph induced by  $\mathcal{V}_i = \{\alpha_i, \dots, \alpha_N\}$ . It is well-known that a graph  $\mathcal{G}$  is chordal if and only if it has a perfect elimination ordering [49], which can be computed in  $\mathcal{O}(|\mathcal{V}| + |\mathcal{E}|)$  time. Let  $\Gamma = \{\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_p\}$  be the set of maximal

cliques for a chordal graph  $\mathcal{G}$ . To compute a clique tree over the set of cliques, we first define a weighted clique graph  $\mathcal{W}(\mathcal{G})$  with set of nodes  $\Gamma$  and edge weights  $w_{ij} = |\mathcal{C}_i \cap \mathcal{C}_j|$ . The problem of computing a clique tree that satisfies the running intersection property can be reduced to computing a maximum-weight spanning tree in  $\mathcal{W}(\mathcal{G})$  [19], which can be carried out using Prim's algorithm [50] (see Procedure 3).

**ACKNOWLEDGMENT**

The authors would like to thank the Associate Editor and the anonymous reviewers for their valuable comments and suggestions that helped us improve the quality of the manuscript.

**REFERENCES**

- [1] Y. Zheng, R. P. Mason, and A. Papachristodoulou, "A chordal decomposition approach to scalable design of structured feedback gains over directed graphs," in *IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 2016, pp. 6909–6914.
- [2] A. Zečević and D. Šiljak, "Control design with arbitrary information structure constraints," *Automatica*, vol. 44, no. 10, pp. 2642–2647, 2008.
- [3] J. Swigart and S. Lall, "Optimal controller synthesis for decentralized systems over graphs via spectral factorization," *Automatic Control, IEEE Transactions on*, vol. 59, no. 9, pp. 2311–2323, 2014.
- [4] Y. Zheng, S. Eben Li, J. Wang, D. Cao, and K. Li, "Stability and scalability of homogeneous vehicular platoon: Study on the influence of information flow topologies," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 17, no. 1, pp. 14–26, 2016.
- [5] M. Rotkowitz and S. Lall, "A characterization of convex problems in decentralized control," *Automatic Control, IEEE Transactions on*, vol. 51, no. 2, pp. 274–286, 2006.
- [6] F. Dorler, M. R. Jovanovic, M. Chertkov, and F. Bullo, "Sparsity-promoting optimal wide-area control of power networks," *IEEE Trans. Power Syst.*, vol. 29, no. 5, pp. 2281–2291, 2014.
- [7] D. M. Stipanović, G. Inalhan, R. Teo, and C. J. Tomlin, "Decentralized overlapping control of a formation of unmanned aerial vehicles," *Automatica*, vol. 40, no. 8, pp. 1285–1296, 2004.
- [8] Y. Zheng, S. E. Li, K. Li, and L.-Y. Wang, "Stability margin improvement of vehicular platoon considering undirected topology and asymmetric control," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 4, pp. 1253–1265, 2016.
- [9] V. Blondel and J. N. Tsitsiklis, "NP-hardness of some linear control design problems," *SIAM Journal on Control and Optimization*, vol. 35, no. 6, pp. 2118–2127, 1997.
- [10] P. Shah and P. Parrilo, "H2-optimal decentralized control over posets: A state-space solution for state-feedback," *Automatic Control, IEEE Transactions on*, vol. 58, no. 12, pp. 3084–3096, 2013.
- [11] J.-H. Kim and S. Lall, "Explicit solutions to separable problems in optimal cooperative control," *Automatic Control, IEEE Transactions on*, vol. 60, no. 5, pp. 1304–1319, 2015.
- [12] G. Fazelnia, R. Madani, A. Kalbat, and J. Lavaei, "Convex relaxation for optimal distributed control problems," *IEEE Transactions on Automatic Control*, vol. 62, no. 1, pp. 206–221, 2017.
- [13] K. Dvijotham, E. Todorov, and M. Fazel, "Convex structured controller design in finite horizon," *Control of Network Systems, IEEE Transactions on*, vol. 2, no. 1, pp. 1–10, 2015.
- [14] F. Lin, M. Fardad, and M. R. Jovanovic, "Augmented Lagrangian approach to design of structured optimal state feedback gains," *Automatic Control, IEEE Transactions on*, vol. 56, no. 12, pp. 2923–2929, 2011.
- [15] —, "Design of optimal sparse feedback gains via the alternating direction method of multipliers," *Automatic Control, IEEE Transactions on*, vol. 58, no. 9, pp. 2426–2431, 2013.
- [16] A. Satya Mohan Vamsi and N. Elia, "Optimal distributed controllers realizable over arbitrary networks," *Automatic Control, IEEE Transactions on*, vol. 61, no. 1, pp. 129–144, Jan 2016.
- [17] W. Su, H. Eichi, W. Zeng, and M.-Y. Chow, "A survey on the electrification of transportation in a smart grid environment," *Industrial Informatics, IEEE Transactions on*, vol. 8, no. 1, pp. 1–10, 2012.
- [18] J. Zhang, F.-Y. Wang, K. Wang, W.-H. Lin, X. Xu, and C. Chen, "Data-driven intelligent transportation systems: A survey," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 12, no. 4, pp. 1624–1639, 2011.



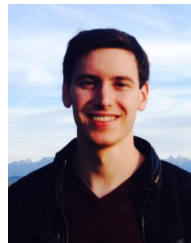
- [19] J. R. Blair and B. Peyton, "An introduction to chordal graphs and clique trees," in *Graph theory and sparse matrix computation*. Springer, 1993, pp. 1–29.
- [20] L. Vandenberghe and M. S. Andersen, "Chordal graphs and semidefinite optimization," *Foundations and Trends® in Optimization*, vol. 1, no. 4, pp. 241–433, 2014.
- [21] F. Gavril, "Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph," *SIAM Journal on Computing*, vol. 1, no. 2, pp. 180–187, 1972.
- [22] D. J. Rose, "Triangulated graphs and the elimination process," *J. Math. Anal. Appl.*, vol. 32, no. 3, pp. 597–609, 1970.
- [23] J. Dahl, L. Vandenberghe, and V. Roychowdhury, "Covariance selection for nonchordal graphs via chordal embedding," *Optimization Methods & Software*, vol. 23, no. 4, pp. 501–520, 2008.
- [24] M. J. Wainwright and M. I. Jordan, "Graphical models, exponential families, and variational inference," *Foundations and Trends® in Machine Learning*, vol. 1, no. 1–2, pp. 1–305, 2008.
- [25] R. Grone, C. R. Johnson, E. M. Sá, and H. Wolkowicz, "Positive definite completions of partial hermitian matrices," *Linear algebra and its applications*, vol. 58, pp. 109–124, 1984.
- [26] J. Agler, W. Helton, S. McCullough, and L. Rodman, "Positive semidefinite matrices with a given sparsity pattern," *Linear algebra and its applications*, vol. 107, pp. 101–149, 1988.
- [27] M. Fukuda, M. Kojima, K. Murota, and K. Nakata, "Exploiting sparsity in semidefinite programming via matrix completion I: General framework," *SIAM J. Optim.*, vol. 11, no. 3, pp. 647–674, 2001.
- [28] S. Kim, M. Kojima, M. Mevissen, and M. Yamashita, "Exploiting sparsity in linear and nonlinear matrix inequalities via positive semidefinite matrix completion," *Mathematical programming*, vol. 129, no. 1, pp. 33–68, 2011.
- [29] R. P. Mason and A. Papachristodoulou, "Chordal sparsity, decomposing SDPs and the Lyapunov equation," in *American Control Conference (ACC)*, 2014. IEEE, 2014, pp. 531–537.
- [30] M. Andersen, S. Pakazad, A. Hansson, and A. Rantzer, "Robust stability analysis of sparsely interconnected uncertain systems," *Automatic Control, IEEE Transactions on*, vol. 59, no. 8, pp. 2151–2156, 2014.
- [31] M. Yannakakis, "Computing the minimum fill-in is NP-complete," *SIAM Journal on Algebraic Discrete Methods*, vol. 2, no. 1, pp. 77–79, 1981.
- [32] S. E. Li, Y. Zheng, K. Li, and J. Wang, "An overview of vehicular platoon control under the four-component framework," in *Intelligent Vehicles Symposium (IV)*, 2015 IEEE. IEEE, 2015, pp. 286–291.
- [33] A. Rantzer, "Distributed performance analysis of heterogeneous systems," in *49th IEEE Conference on Decision and Control (CDC)*. IEEE, 2010, pp. 2682–2685.
- [34] N. Kakimura, "A direct proof for the matrix decomposition of chordal-structured positive semidefinite matrices," *Linear Algebra and its Applications*, vol. 433, no. 4, pp. 819–823, 2010.
- [35] J. F. Sturm, "Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones," *Optimization methods and software*, vol. 11, no. 1–4, pp. 625–653, 1999.
- [36] A. Rantzer, "Scalable control of positive systems," *European Journal of Control*, vol. 24, pp. 72 – 80, 2015.
- [37] T. Tanaka and C. Langbort, "The bounded real lemma for internally positive systems and H-infinity structured static state feedback," *IEEE transactions on automatic control*, vol. 56, no. 9, pp. 2218–2223, 2011.
- [38] M. Yamashita, K. Fujisawa, M. Fukuda, K. Kobayashi, K. Nakata, and M. Nakata, "Latest developments in the SDPA family for solving large-scale SDPs," in *Handbook on semidefinite, conic and polynomial optimization*. Springer, 2012, pp. 687–713.
- [39] M. S. Andersen, J. Dahl, and L. Vandenberghe, "Implementation of nonsymmetric interior-point methods for linear optimization over sparse matrix cones," *Math. Prog. Computat.*, vol. 2, no. 3, pp. 167–201, 2010.
- [40] Y. Zheng, G. Fantuzzi, A. Papachristodoulou, P. Goulart, and A. Wynn, "Fast ADMM for semidefinite programs with chordal sparsity," *Proc. Amer. Control Conf. (ACC)*, pp. 3335–3340, 2017.
- [41] Y. Sun, M. S. Andersen, and L. Vandenberghe, "Decomposition in conic optimization with partially separable structure," *SIAM Journal on Optimization*, vol. 24, no. 2, pp. 873–897, 2014.
- [42] Y. Zheng, G. Fantuzzi, A. Papachristodoulou, P. Goulart, and A. Wynn, "CDCS: Cone decomposition conic solver, version 1.1," <https://github.com/giofantuzzi/CDCS>, Sep. 2016.
- [43] S. K. Pakazad, A. Hansson, M. S. Andersen, and A. Rantzer, "Distributed semidefinite programming with application to large-scale system analysis," *arXiv preprint arXiv:1504.07755*, 2015.
- [44] M. S. Andersen, A. Hansson, and L. Vandenberghe, "Reduced-complexity semidefinite relaxations of optimal power flow problems," *IEEE Trans. Power Syst.*, vol. 29, no. 4, pp. 1855–1863, 2014.
- [45] E. Kaszkurewicz and A. Bhaya, *Matrix diagonal stability in systems and computation*. Springer Science & Business Media, 2012.
- [46] M. Arcak, "Diagonal stability on cactus graphs and application to network stability analysis," *IEEE Transactions on Automatic Control*, vol. 56, no. 12, pp. 2766–2777, 2011.
- [47] A. Sootla and J. Anderson, "On existence of solutions to structured lyapunov inequalities," in *American Control Conference (ACC)*, 2016. IEEE, 2016, pp. 7013–7018.
- [48] M. Razeghi-Jahromi and A. Seyedi, "Stabilization of networked control systems with sparse observer-controller networks," *IEEE Transactions on Automatic Control*, vol. 60, no. 6, pp. 1686–1691, 2015.
- [49] R. E. Tarjan and M. Yannakakis, "Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs," *SIAM Journal on computing*, vol. 13, no. 3, pp. 566–579, 1984.
- [50] R. C. Prim, "Shortest connection networks and some generalizations," *Bell system technical journal*, vol. 36, no. 6, pp. 1389–1401, 1957.



**Yang Zheng** received his B.E. and M.E. degrees from Tsinghua University, Beijing, China, in 2013 and 2015, respectively. He is currently working toward the DPhil. (Ph.D.) degree in the Department of Engineering Science at the University of Oxford, United Kingdom. His research interests include distributed control of dynamical system over networks, and exploiting sparsity in large-scale semidefinite programs and sum-of-squares programs.



Mr. Zheng received the Best Student Paper Award at the 17th International IEEE Conference on Intelligent Transportation Systems in 2014, and Best Paper Award at the 14th Intelligent Transportation Systems Asia-Pacific Forum in 2015. He was a recipient of the National Scholarship, Outstanding Graduate in Tsinghua University, and the Clarendon Scholarship at the University of Oxford.



**Richard P. Mason** completed his DPhil degree in Engineering Science under the supervision of Professor Papachristodoulou at the University of Oxford. His PhD research focussed on exploiting sparsity within optimisation problems in control theory. After his PhD, Richard took part in Entrepreneur First and Y-Combinator Fellowship programs, before joining Alpha-I as a researcher.

He received the Student Best Paper Award at the 2014 American Control Conference.



**Antonis Papachristodoulou** received the M.A./M.Eng. degree in Electrical and Information Sciences from the University of Cambridge, U.K., and the Ph.D. degree in Control and Dynamical Systems (with a minor in aeronautics) from the California Institute of Technology, Pasadena, CA, USA. Currently, he is an Associate Professor in Engineering Science at the University of Oxford, U.K., and a Tutorial Fellow at Worcester College, Oxford, U.K. He is an EPSRC Fellow for Growth in Synthetic Biology and Director of the EPSRC & BBSRC Centre for Doctoral Training in Synthetic Biology. His research interests include large-scale nonlinear systems analysis, sum of squares programming, synthetic and systems biology, networked systems, and flow control.

He received the 2015 European Control Award for his contributions to robustness analysis and applications to networked control systems and systems biology. In the same year, he received the O. Hugo Schuck Best Paper Award.