Chordal Sparsity in Control and Optimization of Large-scale Systems

Yang Zheng
Balliol College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy

February 2019

To my wife, Mohen Zhang

Acknowledgements

I was very fortunate to have Prof. Antonis Papachristodoulou as my DPhil supervisor, and I would like to thank him for his continued support, guidance, and encouragements. Throughout our meetings and discussions, his insights and intuitions have shaped my understanding of many topics in control and optimization. I will really miss those discussions. I want to thank him for giving me the freedom in exploring different directions during my DPhil study. He has provided many opportunities for me to travel and introduced me to many people. This thesis and my future career would not have been the same without him. I would also like to thank Prof. Mark Cannon and Prof. Anders Rantzer for being my thesis examiners.

My thanks also go to Prof. Maryam Kamgarpour at ETH Zürich, who gave me the opportunity to visit her group in September 2016. I would like to thank her for the kind support and advice she offered, and for sharing insightful ideas in distributed control. Many results in Part I and Part III of this thesis have been obtained in collaboration with Giovanni Fantuzzi. It is my great pleasure to thank Giovanni for everything I learned from him. I have really enjoyed all the discussions we had, and I look forward to my next project with him. I am also grateful to Prof. Paul Goulart and Prof. Andrew Wynn for their invaluable suggestions and inputs on first-order algorithms.

I have met many other wonderful people during my time at Oxford. I am grateful to Mohamadreza Ahmadi for his kind advice during my first year. At that time, I shared an office with James Scott-Brown, Goran Banjac, Bartolomeo Stellato, Gareth Pease, and John Martin, who have always been very kind to provide help of all kinds. I would particularly like to thank Richard Mason for his excellent ideas on chordal graphs, and Ross Drummond for his warm support and careful proofreading. A special thank you goes to Aivar Sootla for our interesting discussions on block factor-width-two matrices. I enjoyed meeting with Luca Furieri during his visit in August 2018, when we had very inspiring discussions on decentralized optimal control. Many thanks also to James Anderson, Dhruva Raman, Harrison Steel, Licio Romao, Michael Garstka, Xiaoqing Chen, Shuhao Yan, and all the other members of Control Group for making it such an enjoyable place to work.

I would also like to acknowledge Prof. Lieven Vandenberghe, Prof. Michal Kočvara, Prof. Na Li, Prof. Pablo Parrilo, Prof. Mario Sznaier, and Prof. Ufuk Topcu, for inviting me to give seminars, and for making time to talk with me and offering supportive advice.

Most of all, I wish to thank my family for their support and encouragements, especially my parents Jianxin Zheng and Fuyan Wu. Finally, to Mohen Zhang, with all of my love, I dedicate this thesis to you. My journey would not have been the same without you.

The work in this thesis was supported in part by the Clarendon Scholarship and the Jason Hu Scholarship.

Abstract

Many large-scale systems have inherent structures that can be exploited to facilitate their analysis and design. This thesis investigates how chordal graph properties can be used to develop scalable methods for solving three classes of problems: sparse semidefinite programs (SDPs), distributed control of networked systems, and sum-of-squares (SOS) programs. By exploiting the properties of chordal graphs and sparse positive semidefinite matrices, we present decomposition methods that are able to scale these problems to much larger instances.

The first part of this thesis proposes a new conversion framework for large-scale SDPs characterized by chordal sparsity. This framework is analogous to standard conversion techniques for interior-point methods, but is more suitable for the application of first-order methods. We develop efficient algorithms based on the *alternating direction method of multipliers* (ADMM) for sparse SDPs in either primal or dual standard form, and for their homogeneous self-dual embedding. The algorithms are made available in the open-source conic solver CDCS. CDCS is the first open-source first-order solver that exploits chordal decomposition and can detect infeasible problems. We demonstrate the performance of CDCS in standard analysis problems of large-scale networked systems.

The second part of this thesis is concerned with solution scalability and model privacy in distributed control of networked systems. Specifically, we apply chordal decomposition in sparse Lyapunov-type linear matrix inequalities arising from structured stabilization and optimal decentralized control of networked systems. We introduce a sequential method based on the clique-intersection property of a clique tree for structured stabilization. This strategy greatly improves the computational efficiency for large-scale sparse systems. In addition, we propose an ADMM algorithm that can maintain model privacy when solving the optimal decentralized control problem.

Finally, we consider large-scale SOS programs. We identify an inherent partial orthogonality in the formulation when recasting general SOS programs using the standard monomial basis. In addition, we extend the well-established chordal decomposition/completion results for sparse positive semidefinite matrices to a subset of SOS matrices. Using a new graph-theoretic viewpoint, we build an explicit relationship between chordal decomposition in SOS optimization and two other recent techniques — DSOS and SDSOS optimization. We demonstrate that chordal decomposition can bring significant speed-ups for large-scale sparse SOS programs.

Contents

List of Figures

List of Tables

Notation

Sets

$\mathbb{N}$	Natural integers $\{0, 1, 2, \ldots\}$
$\mathbb{R}$	Real numbers
$\mathbb{R}^n$	Real vectors of dimension n ($n \times 1$ matrices)
$\mathbb{R}^{n\times m}$	Real matrices of dimension $n \times m$
$\mathbb{R}_+$	Nonnegative real numbers
$\mathbb{R}^n_+$	Nonnegative orthant
$\mathbb{S}^n$	Symmetric matrices of dimension $n \times n$
$\mathbb{S}^n_+ (\mathbb{S}^n_{++})$	Symmetric positive semidefinite (definite) matrices of dimension $n \times n$

Definition and inequalities

$A \equiv B$	A and B are equivalent
$A := B$	A is defined by B
$A - B \succeq 0$	$A - B$ is positive semidefinite
$A - B \succ 0$	$A - B$ is positive definite
$A - B \preceq 0$	$B - A$ is positive semidefinite
$A - B \prec 0$	$B - A$ is positive definite

Vectors and matrices

I	Identity matrix
X^T	Transpose of matrix X
$\mathrm{Trace}(X)$	Trace of matrix X
$A \otimes B$	Kronecker product between matrices A and B
$\mathrm{diag}(X_1, \ldots, X_n)$	Block-diagonal matrix with $X_1, \ldots, X_n$ on its diagonal blocks
$\mathrm{vec}(A)$	Vector by stacking columns of $A \in \mathbb{S}^n$
$\mathrm{mat}(x)$	Inverse operator of vec
$\langle x, y\rangle$	Inner product of vectors x and y, *i.e.*, $\langle x, y\rangle = x^\mathsf{T} y$
$\langle A, B\rangle$	Inner product of matrices A and B, *i.e.*, $\langle A, B\rangle = \mathrm{Trace}(A^\mathsf{T} B)$

Acronyms

ADMM	Alternating direction method of multipliers
FOM	First-order method
HSDE	Homogeneous self-dual embedding
IPM	Interior-point method
LMI	Linear matrix inequality
LQR	Linear quadratic regulator
PSD	Positive semidefinite
QI	Quadratic invariance
SDP	Semidefinite program
SOS	Sum-of-squares

1

Introduction

1.1 Motivation

Large-scale systems have attracted increasing attention in recent years [1, 2], since they appear in a wide range of engineering applications, including communication networks, vehicle formations, and the smart grid. Despite differences in their nature, these systems share some common features, *e.g.*, dynamical properties imposed by the relevant physical laws and sparsity in the interconnection topology. The objective of this thesis is to investigate how this inherent sparsity can be used to develop scalable methods for control and optimization of large-scale systems.

One common but non-trivial approach is to find a convex formulation or a tight convex relaxation, since typical convex optimization problems can be solved using well-established interior-point methods in polynomial time [3, 4]. In the past decades, significant efforts have been devoted to casting real-world problems into a convex optimization framework. For instance, a *quadratic invariance* (QI) property was identified in [5] to recast the problem of optimal decentralized control into a convex problem. A zero-duality-gap condition for optimal power flow problems was derived in [6], for which a global optimum solution to the optimal power flow problem can be retrieved by solving semidefinite programs (SDPs). The *sum-of-squares* (SOS) technique was introduced in [7, 8], which can handle many analysis and synthesis problems of nonlinear polynomial systems using SDPs [9]. The interested reader is referred to [10] for extensive applications of convex optimization in linear systems theory, and to [11] for an excellent survey on applying semidefinite optimization in real algebraic geometry.

The framework of convex optimization (especially semidefinite optimization) offers powerful capabilities for the analysis and design of many engineering applications. However, in practice, the scale of systems that can be analyzed and designed is still limited by computational resources. This is because the polynomials that bound the computational complexity of SDPs grow rapidly as a function of the instance size. The main motivation

for this thesis is that many real-world large-scale systems have inherent structures and that many of them are sparse in their topological connections. Moreover, in many applications, this sparsity structure can be inherited in the resulting optimization problems, such as network node localization [12], convex relaxations of optimal power flow problems [6], and sparse linear systems analysis [13]. Therefore, it is important to exploit this inherent sparsity to solve the associated optimization problems more efficiently.

In particular, this thesis focuses on exploiting the relationship between chordal graphs and positive semidefinite matrices to solve sparse SDPs that arise in control and optimization of large-scale systems more efficiently. Chordal graphs are a class of undirected graphs where every cycle of length greater than three has a chord (a precise definition will be given in Chapter 2) [14]. Chordal graphs are a well-studied object in graph theory [15]. Several combinatorial optimization problems that are hard on general graphs can be solved efficiently for chordal graphs. Examples include the graph coloring problem and the problem of finding the largest maximum clique in a graph [16]. In numerical algebra, chordal graphs are strongly related to the zero fill-in property during Cholesky factorization of sparse positive definite matrices, and can facilitate the solution of sparse linear equations [17]. Also, chordal graph properties have been applied to maximum likelihood estimation for sparse graphical models [18].

The theory between chordal graphs and sparse SDPs originates from two important decomposition results, which were proven in [19–22]. These two decomposition results essentially reduce a large sparse positive semidefinite (PSD) cone or positive semidefinite completable cone into a set of smaller and coupled cones (see Section 2.3 for details). This decomposition idea was first applied in interior-point methods for sparse semidefinite optimization by Fukuda *et al.* [23]. A conversion method utilizing the decomposition of both the sparse PSD cone and PSD completable cone was introduced for the primal side and dual side of SDPs in [24]. Then, choral graph techniques were exploited to develop fast recursive algorithms to evaluate the function values and derivatives of the barrier functions for sparse SDPs in [25]. The chordal decomposition results that underpin the conversion method of [23, 24] are also important for first-order algorithms [26, 27]. Recently, this line of results has been used in a number of applications, including optimal power flow problems [28–30], SOS optimization [31, 32], and sparse systems analysis [13, 33, 34]. A comprehensive survey on chordal graphs and semidefinite optimization can be found in [15].

This thesis exploits chordal graph properties to develop new scalable methods for control and optimization of sparse large-scale systems. The results in this thesis are categorized into three parts: large-scale sparse semidefinite programs (SDPs) (Part I), distributed control of networked systems (Part II), and large-scale sum-of-squares (SOS) programs (Part III).

1.2 Outline and contributions

We now provide an outline of the thesis, summarizing the main contributions of each chapter. Details of relevant previous work are discussed separately in each chapter.

Chapter 2: Convex optimization and chordal decomposition play a central role throughout the thesis. This chapter first covers a brief overview of convex optimization, focusing on Lagrangian duality, linear matrix inequalities, and semidefinite programs. Some preliminaries on chordal graphs and their relation to sparse matrix decomposition are discussed subsequently, with a focus on the duality relationship between the matrix decomposition results. This chapter also introduces basic definitions of block partitioned matrices, and extends two key chordal matrix decomposition theorems into the case of sparse block matrices.

Part I: large-scale sparse semidefinite programs (SDPs)

The first part of this thesis is devoted to exploiting the potential of chordal decomposition in general sparse SDPs and consists of Chapters 3 and 4.

Chapter 3: In this chapter, we employ chordal decomposition to reformulate a large and sparse semidefinite program (SDP), either in primal or dual standard form, into an equivalent SDP with smaller positive semidefinite (PSD) constraints. In contrast to previous approaches, the decomposed SDP is suitable for the application of first-order operator-splitting methods, enabling the development of efficient and scalable algorithms. In particular, we apply the alternating direction method of multipliers (ADMM) to solve decomposed primal, dual, and homogeneous self-dual embedding forms of SDPs. All algorithms are implemented in the open-source MATLAB solver CDCS [35]. Numerical experiments demonstrate the computational advantages of the proposed methods compared to common state-of-the-art solvers. The results in this chapter have been published in [36–38][1].

Chapter 4: This chapter demonstrates the performance of CDCS [35] on scalable analysis of linear networked systems, including stability, $\mathcal{H}_2$, and $\mathcal{H}_\infty$ performance. The main strategy is to exploit any sparsity within these analysis problems and use chordal decomposition. By choosing block-diagonal Lyapunov functions, we decompose large PSD constraints in all of the analysis problems into multiple smaller ones depending on the maximal cliques of the system graph. This makes the solutions more computationally efficient via CDCS [35]. The results in this chapter have been published in [39].

[1] This line of work was in collaboration with Giovanni Fantuzzi. I initiated the research, and we worked together on the formulation. Giovanni Fantuzzi implemented the primal and dual algorithms, and I focused on the implementation of the homogeneous self-dual embedding algorithm, complexity analysis, and extensive numerical testing. I led the writing of [36–38].

Part II: distributed control of networked systems

Part II of the thesis applies chordal decomposition in the distributed control of networked systems, focusing on solution scalability and model privacy. This part of the thesis includes the following two chapters.

Chapter 5: In this chapter, we consider the problem of designing static feedback gains subject to *a priori* structural constraints. By exploiting the underlying sparsity properties of the problem, and using chordal decomposition, we propose a scalable sequential algorithm to obtain structured feedback gains that stabilize a large-scale system. Several examples demonstrate the efficiency and scalability of the proposed design method. The results in this chapter have been published in [40, 41].

Chapter 6: Synthesizing decentralized controllers in a distributed fashion is desirable due to privacy concerns of model data in certain complex systems. In this chapter, we propose a distributed design method for optimal decentralized control by exploiting the underlying sparsity properties of the problem. Our method combines chordal decomposition of sparse block matrices with the ADMM framework. In our algorithm, the subsystems only need to share their model data with their neighbours, not centrally or globally. The results in this chapter have been summarized in [42].

Part III: large-scale sum-of-squares (SOS) programs

This part of the thesis, consisting of Chapters 7 — 9, focuses on exploiting sparsity in SOS programs to facilitate the solution scalability.

Chapter 7: This chapter focuses on general SOS programs. We first show that the constraint matrices of the SDP arising in SOS programs possess a structural property that we call *partial orthogonality*. Then, we leverage partial orthogonality to develop a fast first-order method for the solution of the homogeneous self-dual embedding of SDPs describing SOS programs. The resulting algorithm has been implemented as a new package in the solver CDCS. The results in this chapter have been published in [43].

Chapter 8: This chapter introduces a notion of decomposition and completion of SOS matrices. We show that a subset of sparse SOS matrices with chordal sparsity patterns can be equivalently decomposed into a sum of multiple SOS matrices that are nonzero only on certain principal submatrices. Also, the completion of an SOS matrix is equivalent to a set of SOS conditions on its principal submatrices and a consistency condition on the Gram representation of the principal submatrices. These results are partial extensions of chordal decomposition and completion of constant matrices to matrices with polynomial entries. Chapter 8 is based on the work in [44].

Chapter 9: This chapter investigates the relation between three tractable relaxations for optimizing over *sparse* non-negative polynomials: sparse sum-of-squares (SSOS)

optimization, diagonally dominant sum-of-squares (DSOS) optimization, and scaled diagonally dominant sum-of-squares (SDSOS) optimization. We show that for polynomials with chordal correlative sparsity, DSOS/SDSOS optimization is provably more conservative than SSOS optimization. Also, SSOS optimization promises better scalability compared to standard SOS optimization. Therefore, SSOS optimization bridges the existing theoretical and computational gaps between DSOS/SDSOS and SOS optimization for sparse instances. Chapter 9 is based on the work in [45].

Chapter 10: This chapter summarizes the main results of this thesis and suggests some future research topics.

Appendix A: We present some discussions on block-diagonal Lyapunov functions and strongly decentralized stabilization, which are key concepts in Chapters 4, 5, and 6.

Other publications

The work of this thesis has also led to the following publications.

- Y. Zheng, G. Fantuzzi, and A. Papachristodoulou. "Decomposition methods for large-scale semidefinite programs with chordal aggregate sparsity and partial orthogonality". In: *Large-Scale and Distributed Optimization*. Ed. by P. Giselsson and A. Rantzer. Springer International Publishing, 2018. Chap. 3
- A. A. Ahmadi, G. Hall, A. Papachristodoulou, J. Saunderson, and Y. Zheng. "Improving efficiency and scalability of sum of squares optimization: Recent advances and limitations". In: *Decision and Control (CDC), IEEE 56th Annual Conference on*. IEEE. 2017, pp. 453–462

In addition, the contributions of the following papers are not explicitly covered in this thesis.

- Y. Zheng, G. Fantuzzi, and A. Papachristodoulou. "Exploiting sparsity in the coefficient matching conditions in sum-of-squares programming using ADMM". In: *IEEE Control System Letter* 1.1 (2017), pp. 80–85
- A. Sootla, Y. Zheng, and A. Papachristodoulou. "Block-diagonal solutions to Lyapunov inequalities and generalisations of diagonal dominance". In: *Decision and Control (CDC), IEEE 56th Annual Conference on*. IEEE. 2017, pp. 6561–6566
- A. Sootla, Y. Zheng, and A. Papachristodoulou. "Block factor-width-two matrices in semidefinite programming". In: *European Control Conference (ECC), accepted* (2019)
- L. Furieri, Y. Zheng, A. Papachristodoulou, and M. Kamgarpour. "On separable quadratic Lyapunov functions for convex design of distributed controllers". In: *European Control Conference (ECC), accepted* (2019)

Code, software, and numerical examples

To facilitate the reproducibility of the results in the thesis, the code for all the numerical examples is available to download from

https://github.com/zhengy09.

In particular, the work of this thesis has led to the development of two open-source conic solvers, and I am one of the two main developers of these solvers[2].

1. CDCS: Cone Decomposition Conic Solver. An open-source first-order MATLAB solver for sparse conic programs by exploiting chordal sparsity, available at https://github.com/oxfordcontrol/CDCS.

2. SOSADMM. An open source first-order MATLAB solver for conic programs with row sparsity, especially for the SDPs arising from SOS programs, available at https://github.com/oxfordcontrol/SOSADMM.

[2]Another main developer of CDCS is Giovanni Fantuzzi.

2

Preliminaries: convex optimization, chordal graphs, and sparse matrix decomposition

In this chapter, we first present some preliminaries on convex optimization, focusing on Lagrangian duality, linear matrix inequalities, and semidefinite programs. These concepts are well-known in the control and optimization communities. Convex optimization, together with the notion of chordal graphs, plays a fundamental role in this thesis. We next cover some mathematical preliminaries on chordal graphs and their relation to sparse matrix decomposition. We also present the notion of sparse block matrices. This chapter aims to give a self-contained introduction to material that will be used in subsequent chapters. For a detailed treatment, we refer the reader to the excellent works on convex optimization [3, 4, 11] and the comprehensive surveys on chordal graphs [14, 15].

2.1 Convex optimization

2.1.1 Convex sets and convex functions

To define a convex optimization problem, we first need to define convex sets and convex functions. A set $\mathcal{S} \subseteq \mathbb{R}^n$ is convex if $\forall x_1, x_2 \in \mathcal{S}$ and $0 \leq \theta \leq 1$, we have

$$\theta x_1 + (1-\theta)x_2 \in \mathcal{S}.$$

An intersection of two convex sets is convex, *i.e.*, if $\mathcal{S}_1$ and $\mathcal{S}_2$ are convex, then $\mathcal{S}_1 \cap \mathcal{S}_2$ is convex. Also, this property extends to the intersection of an infinite number of convex sets. An important subclass of convex sets is convex cones. A set $\mathcal{S} \subseteq \mathbb{R}^n$ is a convex cone if it is convex and closed under nonnegative scaling, *i.e.*, $\forall x_1, x_2 \in \mathcal{S}$ and $\theta_1, \theta_2 \geq 0$, we have

$$\theta_1 x_1 + \theta_2 x_2 \in \mathcal{S}.$$

The nonnegative orthant $\mathbb{R}^n_+ := \{x \in \mathbb{R}^n \mid x \geq 0\}$ and the set of real $n \times n$ positive semidefinite matrices $\mathbb{S}^n_+$ are examples of convex cones.

Let $K \subseteq \mathbb{R}^n$ be a cone. The dual cone of K is defined as

$$K^* = \{y \in \mathbb{R}^n \mid x^\mathsf{T} y \geq 0, \forall x \in K\}.$$

The dual cone K^* is always closed and convex. If $K_1 \subseteq K_2$, then $K_2^* \subseteq K_1^*$. Also, the nonnegative orthant $\mathbb{R}^n_+$ and the positive semidefinite cone $\mathbb{S}^n_+$ are self-dual, meaning that $(\mathbb{R}^n_+)^* = \mathbb{R}^n_+$ and $(\mathbb{S}^n_+)^* = \mathbb{S}^n_+$.

A function $f : \Omega \subseteq \mathbb{R}^n \mapsto \mathbb{R}$ is convex if its domain Ω is convex and $\forall x, y \in \Omega$ and $0 \leq \theta \leq 1$, we have

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta) f(y).$$

Important examples of convex functions include affine functions, norms, and the indicator function of a convex set $\mathcal{S}$, defined as

$$\mathbb{I}_{\mathcal{S}}(x) := \begin{cases} 0, & x \in \mathcal{S}, \\ +\infty, & \text{otherwise.} \end{cases}$$

A *convex optimization* problem is a problem of the form

$$\begin{aligned} \text{minimize} \quad & f_0(x) \\ \text{subject to} \quad & f_i(x) \leq 0, \ i = 1, \ldots, m, \\ & h_i(x) = 0, \ i = 1, \ldots, p, \end{aligned} \tag{2.1}$$

where the functions $f_0, f_1, \ldots, f_m : \Omega \subseteq \mathbb{R}^n \mapsto \mathbb{R}$ are convex, and the functions $h_1, \ldots, h_p : \Omega \subseteq \mathbb{R}^n \mapsto \mathbb{R}$ are affine. The function f_0 is called the *cost function* or *objective function*, and the remaining functions f_i and h_i are referred to as *constraint functions*. The *feasible set* of (2.1) is the set of points satisfying all the constraints. If the feasible set is empty, we say problem (2.1) is infeasible.

2.1.2 Lagrangian duality

Lagrangian duality provides a framework to study (2.1) by augmenting the objective function f_0 with a weighted sum of the constraint functions $f_1, \ldots, f_m, h_1, \ldots, h_p$.

Given $\lambda \in \mathbb{R}^m, \nu \in \mathbb{R}^p$, the *Lagrangian function* associated with (2.1) is defined as

$$L(x, \lambda, \nu) := f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x), \tag{2.2}$$

where λ_i and ν_i are known as the *Lagrangian multipliers* for the inequality constraint $f_i(x) \leq 0$ and the equality constraint $h_i(x) = 0$, respectively. The *dual function* is defined as

$$g(\lambda, \nu) := \inf_{x \in \Omega} L(x, \lambda, \nu). \tag{2.3}$$

The dual function $g(\lambda, \nu)$ is always concave since it is the pointwise infimum of a family of affine functions. Also, $\forall \lambda \geq 0$ and ν, $g(\lambda, \nu)$ provides a lower bound of the optimal value of (2.1). Indeed, assuming that (2.1) is feasible and that there exists an optimal point x^* achieving the optimal value $p^* := f(x^*)$, $\forall \lambda \geq 0$ and ν, one has

$$\begin{aligned} g(\lambda, \nu) = \inf_{x \in \Omega} L(x, \lambda, \nu) &\leq L(x^*, \lambda, \nu) \\ &= f_0(x^*) + \sum_{i=1}^{m} \lambda_i f_i(x^*) + \sum_{i=1}^{p} \nu_i h_i(x^*) \\ &\leq f_0(x^*) = p^*, \end{aligned} \tag{2.4}$$

where the last inequality holds since $\lambda_i \geq 0, f_i(x^*) \leq 0$ and $h_i(x^*) = 0$. In particular, we can look for the best lower bound on p^* by solving the following problem

$$\begin{aligned} \text{maximize} \quad & g(\lambda, \nu) \\ \text{subject to} \quad & \lambda \geq 0, \end{aligned} \tag{2.5}$$

which is known as the *dual problem* associated with (2.1). We denote the optimal value of (2.5) as d^*. Following (2.4), we have an important inequality $d^* \leq p^*$, which is called *weak duality*, and it holds for a general problem (convex or not). If the equality $d^* = p^*$ is achieved, we say that *strong duality* holds. Strong duality does not hold in general. Given a convex problem (2.1), one sufficient condition for *strong duality* is *Slater's condition* [3, Section 5.2.3], [4, Section 5.3].

2.1.3 Linear matrix inequalities

Many linear systems analysis and synthesis problems naturally involve linear matrix inequalities (LMIs) [10]. Specifically, an LMI is a constraint of the form

$$A(x) := A_0 + \sum_{i=1}^{m} A_i x_i \succeq 0, \tag{2.6}$$

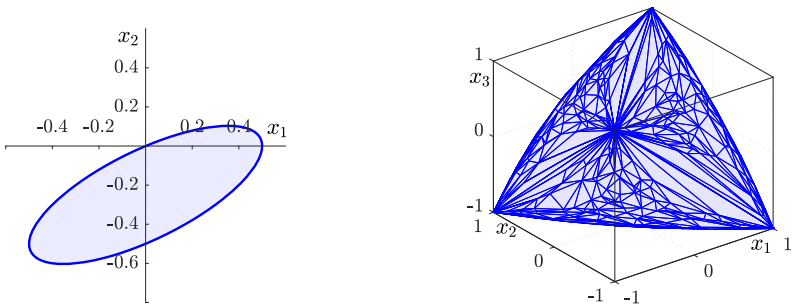
where $x \in \mathbb{R}^m$ is a variable, and $A_0, A_1, \ldots, A_m \in \mathbb{S}^n$ are given symmetric matrices. It is not difficult to see that the LMI (2.6) defines a convex set on x, *i.e.*, the set

$$\mathcal{S} := \{x \in \mathbb{R}^m \mid A(x) \succeq 0\}$$

is convex, *i.e.*, $\forall y, z \in \mathcal{S}$ and $0 \leq \theta \leq 1$, we have $\theta y + (1-\theta) z \in \mathcal{S}$. The convexity of $\mathcal{S}$ can also be seen from the equivalence that $A(x) \succeq 0 \Leftrightarrow v^{\mathsf{T}}(A_0 + \sum_{i=1}^{m} A_i x_i) v \geq 0, \forall v \in \mathbb{R}^n$ (the intersection of an infinite number of convex sets is convex).

Example 2.1. Let $x \in \mathbb{R}^2$ and consider a 2×2 LMI

$$A(x) := \begin{bmatrix} 1 - x_1 + 2x_2 & x_1 \\ x_1 & x_1 - 2x_2 \end{bmatrix} \succeq 0,$$



(a) (b)

Figure 2.1: (a) Feasible set of the LMI in Example 2.1, enclosed with the ellipse $x_1 - 2x_2 - 2x_1^2 + 4x_1x_2 - 4x_2^2 \geq 0$. (b) Feasible set of the LMI in Example 2.2, which is a 3-dimensional elliptope.

which can be rewritten in the form (2.6) with

$$A_0 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \quad A_1 = \begin{bmatrix} -1 & 1 \\ 1 & 1 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 2 & 0 \\ 0 & -2 \end{bmatrix}.$$

Since a 2×2 symmetric matrix is positive semidefinite if and only if its trace and determinant are nonnegative, the feasible set $\mathcal{S} := \{x \in \mathbb{R}^2 \mid A(x) \succeq 0\}$ can be found by imposing $(1 - x_1 + 2x_2)(x_1 - 2x_2) - x_1^2 = x_1 - 2x_2 - 2x_1^2 + 4x_1x_2 - 4x_2^2 \geq 0$. This quadratic polynomial inequality defines an ellipse shown in Figure 2.1(a).

Example 2.2. Let $x \in \mathbb{R}^3$ and consider a 3×3 LMI

$$A(x) := \begin{bmatrix} 1 & x_1 & x_2 \\ x_1 & 1 & x_3 \\ x_2 & x_3 & 1 \end{bmatrix} \succeq 0,$$

which can be rewritten in the form (2.6) with

$$A_0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad A_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \quad A_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}.$$

The feasible set $\mathcal{S} := \{x \in \mathbb{R}^3 \mid A(x) \succeq 0\}$ is a 3-dimensional elliptope [11], plotted in Figure 2.1(b).

Linear matrix inequalities can represent many common types of constraints. We list three typical examples that will be encountered in the subsequent chapters. In the following, we denote the optimization variable as $y \in \mathbb{R}^m$.

- **Linear inequalities.** Given $A \in \mathbb{R}^{n \times m}$ and $b \in \mathbb{R}^n$, a set of linear inequalities $Ay \leq b$ is equivalent to the diagonal LMI

$$\operatorname{diag}\left(\sum_{i=1}^{m} y_i A_{1i} - b_1, \sum_{i=1}^{m} y_i A_{2i} - b_2, \ldots, \sum_{i=1}^{m} y_i A_{ni} - b_n\right) \succeq 0. \tag{2.7}$$

- **Second-order cone constraints.** Given $A \in \mathbb{R}^{n\times m}, b \in \mathbb{R}^n, c \in \mathbb{R}^m$, and $d \in \mathbb{R}$, by Schur's complement condition [3, Appendix A.5.5], a second-order constraint $\|Ay + b\| \le c^\mathsf{T} y + d$ is equivalent to the LMI constraint

$$\begin{bmatrix} (c^\mathsf{T} y + d)I & Ay + b \\ (Ay + b)^\mathsf{T} & c^\mathsf{T} y + d \end{bmatrix} \succeq 0. \tag{2.8}$$

- **Polynomial sum-of-squares constraints.** Given $x \in \mathbb{R}^n$, we denote $p(x)$ as a polynomial in x of degree $2d$. We say $p(x)$ is a sum-of-squares (SOS) polynomial if it can be written into a sum of squares of other polynomials of degree no greater than d. It is known that $p(x)$ admits an SOS decomposition if and only if there exists a positive semidefinite $Q \in \mathbb{S}^s_+$ with $s := \binom{n+d}{d}$ such that [52]

$$p(x) = v_d(x)^\mathsf{T} Q v_d(x), \tag{2.9}$$

  where $v_d(x)$ is a vector of monomials of degree no greater than d. When the coefficients of $p(x)$ depend affinely on $y \in \mathbb{R}^m$, comparing coefficients on both sides of (2.9) leads to a set of equalities on y and Q. Considering the positive semidefiniteness of Q, an SOS constraint on $p(x)$ is equivalent to an LMI constraint involving y and Q.

Example 2.3. Let $x \in \mathbb{R}^2, y \in \mathbb{R}^3$ and consider a parametric polynomial

$$p(x) := x_1^2 + x_2^2 + 2y_3 x_1 x_2 + 2y_1 x_1 + 2y_2 x_2 + 1.$$

With $v_d(x) = \begin{bmatrix} 1, x_1, x_2 \end{bmatrix}^\mathsf{T}$ and $Q \in \mathbb{S}^3$, we have

$$\begin{aligned} p(x) &= \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}^\mathsf{T} \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{12} & q_{22} & q_{23} \\ q_{13} & q_{23} & q_{33} \end{bmatrix} \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix} \\ &= q_{33} x_2^2 + q_{22} x_1^2 + 2q_{23} x_1 x_2 + 2q_{12} x_1 + 2q_{13} x_2 + q_{11}. \end{aligned}$$

Matching coefficients on both sides lead to

$$q_{11} = 1, \quad q_{22} = 1, \quad q_{33} = 1, \quad q_{12} = y_1, \quad q_{13} = y_2, \quad q_{23} = y_3.$$

Consequently, we have

$$\{y \in \mathbb{R}^3 \mid p(x) \text{ is SOS}\} \equiv \left\{ y \in \mathbb{R}^3 \;\middle|\; \begin{bmatrix} 1 & y_1 & y_2 \\ y_1 & 1 & y_3 \\ y_2 & y_3 & 1 \end{bmatrix} \succeq 0 \right\},$$

which is the same as the 3-dimensional elliptope shown in Figure 2.1(b).

2.1.4 Semidefinite programs

A semidefinite program (SDP) is a convex optimization problem involving a linear cost function subject to linear matrix inequalities [3, 11]. A standard *primal form* SDP is a problem of the form

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & \langle C, X\rangle \\ \text{subject to} \quad & \langle A_i, X\rangle = b_i, i = 1, \ldots, m, \\ & X \succeq 0, \end{aligned} \tag{2.10}$$

where $C, A_i \in \mathbb{S}^n, i = 1, \ldots, m, b \in \mathbb{R}^m$ are given problem data. Note that $X \succeq 0$ is a particular LMI constraint. The Lagrangian dual problem associated with (2.10) is a problem involving a general-form LMI constraint.

To derive the dual problem, following Section 2.1.2, we augment the cost function with the constraints in (2.10) and define the Lagrangian function as

$$\begin{aligned} L(X, y, Z) :=& \langle C, X\rangle + \sum_{i=1}^{m} y_i \left(b_i - \langle A_i, X\rangle\right) - \langle X, Z\rangle \\ =& \left\langle C - \sum_{i=1}^{m} y_i A_i - Z, X \right\rangle + b^{\mathsf{T}} y. \end{aligned}$$

The dual function is

$$g(y, Z) := \inf_X L(X, y, Z) = \begin{cases} b^{\mathsf{T}} y, & \text{if } C - \sum_{i=1}^{m} y_i A_i - Z = 0, \\ -\infty, & \text{otherwise.} \end{cases}$$

As stated in Section 2.1.2, the dual function $g(y, Z)$ provides a lower bound on the optimal value p^* of (2.10): $\forall y \in \mathbb{R}^m$ and $Z \succeq 0$, one has

$$g(y, Z) = \inf_X L(X, y, Z) \leq \inf_{X \succeq 0, \langle A_i, X\rangle = b_i} L(X, y, Z) \leq p^*,$$

where the last inequality holds since $\langle X, Z\rangle \geq 0, \forall X \succeq 0, Z \succeq 0$. Then, the dual problem associated with (2.10) is

$$\begin{aligned} \underset{y,Z}{\text{maximize}} \quad & b^{\mathsf{T}} y \\ \text{subject to} \quad & Z + \sum_{i=1}^{m} y_i A_i = C, \\ & Z \succeq 0, \end{aligned} \tag{2.11}$$

which is equivalent to a problem involving a general-form LMI constraint

$$\begin{aligned} \underset{y}{\text{maximize}} \quad & b^{\mathsf{T}} y \\ \text{subject to} \quad & C - \sum_{i=1}^{m} y_i A_i \succeq 0. \end{aligned} \tag{2.12}$$

In the literature, problem (2.11) is referred to as a standard *dual form* SDP. Following the fact that LMIs can represent linear inequalities and second-order cone constraints (see Section 2.1.3), SDPs include linear programs and second-order cone programs as special cases. Also, linear optimization over SOS polynomials can be equivalently formulated into an SDP problem, which is a crucial fact for many applications [7, 8, 11].

2.2 Chordal graphs

Chordal graphs play an important role in exploiting sparsity in large-scale SDPs and related convex problems involving sparse positive semidefinite matrices [15]. This section introduces the notation in graph theory that will be used throughout this thesis, and reviews some basic definitions and results on chordal graphs [14, 15].

2.2.1 Chordal graph

A graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is defined by a set of vertices $\mathcal{V} = \{1, 2, \ldots, n\}$ and a set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. A graph is called *complete* if any two nodes are connected by an edge. A path in $\mathcal{G}$ is a sequence of edges that connect a sequence of distinct vertices. A graph is called *connected* if there is a path between every pair of vertices. A subset of vertices $\mathcal{C} \subseteq \mathcal{V}$ is called a *clique*, if $(i, j) \in \mathcal{E}$ for any distinct vertices $i, j \in \mathcal{C}$, *i.e.*, the subgraph induced by $\mathcal{C}$ is complete. The number of vertices in $\mathcal{C}$ is denoted by $|\mathcal{C}|$. If $\mathcal{C}$ is not a subset of any other clique, then it is referred to as a *maximal clique*. A *cycle* of length k in a graph $\mathcal{G}$ is a set of pairwise distinct nodes $\{v_1, v_2, \ldots, v_k\} \subset \mathcal{V}$ such that $(v_k, v_1) \in \mathcal{E}$ and $(v_i, v_{i+1}) \in \mathcal{E}$ for $i = 1, \ldots, k-1$. A *chord* is an edge joining two non-adjacent nodes in a cycle. A graph $\mathcal{G}$ is undirected if $(v_i, v_j) \in \mathcal{E} \Leftrightarrow (v_j, v_i) \in \mathcal{E}$.

Definition 2.4. An undirected graph $\mathcal{G}$ is called *chordal* if every cycle of length greater than or equal to four has at least one chord.

Chordal graphs include several other classes of graphs, such as acyclic undirected graphs, undirected graphs with cycles of length no greater three, and complete graphs. Algorithms such as the maximum cardinality search [53] can test chordality and identify the maximal cliques of a chordal graph efficiently, *i.e.*, in linear time in terms of the number of nodes and edges. Nonchordal graphs can always be *chordal extended*, *i.e.*, extended to a chordal graph, by adding additional edges to the original graph.

Definition 2.5. A *chordal extension* (or *chordal embedding*) of a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is a chordal graph $\hat{\mathcal{G}}(\mathcal{V}, \hat{\mathcal{E}})$, where $\mathcal{E} \subseteq \hat{\mathcal{E}}$.

Figure 2.2: (a) Nonchordal graph: the cycle (1-2-3-4) is of length four but has no chords. (b) Chordal graph: all cycles of length no less than four have a chord; the maximal cliques are $\mathcal{C}_1 = \{1, 2, 4\}$ and $\mathcal{C}_2 = \{2, 3, 4\}$.

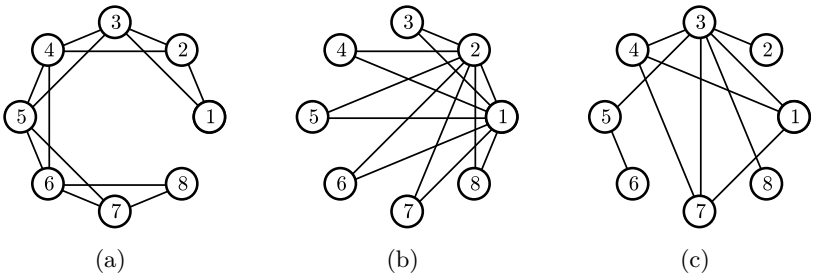


Figure 2.3: Examples of chordal graphs: (a) a banded graph; (b) a block-arrow graph; (c) a generic chordal graph.

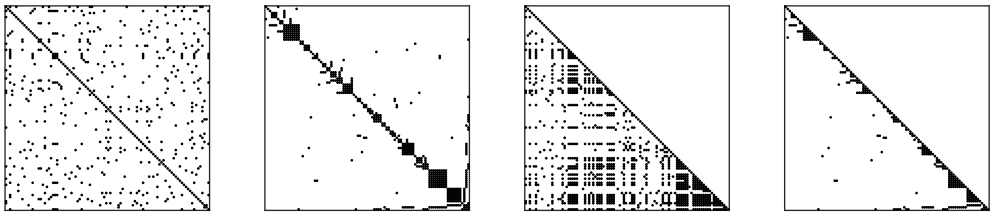
Finding chordal extensions with the minimum number of additional edges corresponds to sparse Cholesky factorization with minimum fill-ins, which is known to be NP-complete [54]. However, several heuristics can be used to find good chordal extensions efficiently, such as the minimum degree ordering followed by a symbolic Cholesky factorization [23]. Figure 2.2 illustrates these concepts. The graph in Figure 2.2(a) is not chordal, but can be chordal extended to the graph in Figure 2.2(b) by adding the edge $(2, 4)$. The chordal graph in Figure 2.2(b) has two maximal cliques, $\mathcal{C}_1 = \{1, 2, 4\}$ and $\mathcal{C}_2 = \{2, 3, 4\}$. Other examples of chordal graphs are given in Figure 2.3.

2.2.2 Perfect elimination orderings

A vertex v of an undirected graph is called *simplicial* if all its neighbors are connected to each other, *i.e.*, the subgraph induced by its neighbors is complete. It is known that every chordal graph has at least one simplicial vertex [15, Section 3.4]. For example, vertex 1 and 3 are both simplicial in Figure 2.2(b).

A bijection $\sigma : \mathcal{V} \mapsto \{1, \ldots, n\}$ is called an ordering of $\mathcal{G}(\mathcal{V}, \mathcal{E})$. For simplicity, one can write an ordering as $\sigma = \langle v_1, \ldots, v_n \rangle$ where $i = \alpha(v_i)$. An ordering σ of an undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is a *perfect elimination ordering* if each $v_i, i = 1, \ldots, n$, is a simplicial vertex in the subgraph induced by the nodes $\{v_i, v_{i+1}, \ldots, v_n\}$. Then, we have an equivalent definition of chordal graphs.

Theorem 2.6 ([15, Theorem 4.1])**.** A graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is chordal if and only if $\mathcal{G}$ has a perfect elimination ordering.



(a) (b) (c) (d)

Figure 2.4: Sparsity pattern of a positive definite matrix $A \in \mathbb{S}^n_{++}(\mathcal{E}, 0)$ with a chordal pattern and its Cholesky factor before/after performing a perfect elimination ordering permutation: (a) pattern of A; (b) pattern of $P_\sigma A P_\sigma^\mathsf{T}$; (c) pattern of the Cholesky factor of A; (d) pattern of the Cholesky factor of $P_\sigma A P_\sigma^\mathsf{T}$.

Given a chordal graph, a perfect elimination ordering can be found efficiently, *e.g.*, using the maximum cardinality search [53] that has an $\mathcal{O}(|\mathcal{V}| + |\mathcal{E}|)$ complexity. Finding perfect elimination orderings for chordal graphs has an important application in numerical algebra. In particular, positive definite matrices with chordal sparsity patterns (a precise definition will be given in Section 2.3) always admit Cholesky factorizations with zero fill-ins [17]. Precisely, given a positive definite matrix A with a chordal sparsity pattern, we have a Cholesky factorization

$$P_\sigma A P_\sigma^\mathsf{T} = L L^\mathsf{T},$$

where P_σ is a permutation matrix according to the perfect elimination ordering σ, and L is a lower-triangular matrix. Then, $P_\sigma^\mathsf{T}(L + L^\mathsf{T})P_\sigma$ has the same sparsity pattern as A.

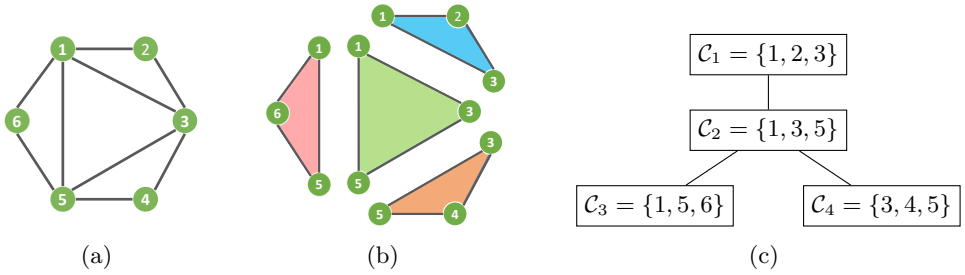
Example 2.7. There exists no perfect elimination ordering for the nonchordal graph in Figure 2.2 (a), while for the chordal graph in Fig 2.2 (b), one can verify that $\langle 1, 2, 3, 4\rangle$ is a perfect elimination ordering. Consequently, the following Cholesky factorization admits zero fill-ins,

$$\underbrace{\begin{bmatrix} 4 & 2 & 0 & 2 \\ 2 & 2 & 1 & 2 \\ 0 & 1 & 5 & 3 \\ 2 & 2 & 3 & 4 \end{bmatrix}}_{A \succ 0} = \underbrace{\begin{bmatrix} 2 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 2 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}}_{L} \cdot \underbrace{\begin{bmatrix} 2 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{L^\mathsf{T}},$$

where A and $L + L^\mathsf{T}$ have the same sparsity pattern corresponding to Figure 2.2 (b) and the $(1, 3)$-th element is zero in both A and $L + L^\mathsf{T}$. Figure 2.4 shows the zero fill-in property for another positive definite matrix with a chordal sparsity pattern.

2.2.3 Maximal cliques and clique trees

Listing all maximal cliques for a general graph is computationally challenging since the number of maximal cliques can increase exponentially in $|\mathcal{V}|$. However, a chordal



(a) (b) (c)

Figure 2.5: Illustration of the chordal graph decomposition: (a) a chordal graph with six nodes; (b) maximal cliques; (c) a clique tree that satisfies the clique intersection property.

graph with n vertices can have at most n maximal cliques [16]. Moreover, given a perfect elimination ordering, the maximal cliques of a chordal graph can be identified in linear time.

Let $\mathcal{G}(\mathcal{V}, \mathcal{E})$ be a connected chordal graph with a set of maximal cliques $\Gamma = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_t\}$. These maximal cliques can be further arranged in a clique tree $\mathcal{T} = (\Gamma, \Xi)$ with the maximal cliques as its vertices and $\Xi \subseteq \Gamma \times \Gamma$, which satisfies the *clique-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 [14]. A related notion is the so-called *running intersection property*: an ordering of the maximal cliques $\mathcal{C}_1, \ldots, \mathcal{C}_t$ satisfies the running intersection property if for every $k = 1, 2, \ldots, t-1$, such that

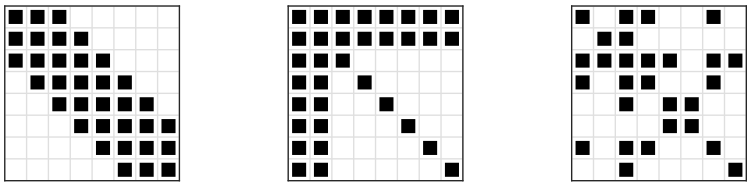
$$\left(\mathcal{C}_{k+1} \cap \bigcup_{j=1}^{k} \mathcal{C}_j\right) \subseteq \mathcal{C}_s, \text{for some } s \leq k.$$

Given a clique tree that satisfies the clique-intersection property, a topological ordering of the nodes in the clique tree naturally satisfies the running intersection property. Recall that a topological ordering is an ordering of the nodes in a rooted tree where each parent node comes before its children. We have another two equivalent characterizations of chordal graphs.

Theorem 2.8 ([14, Theorem 3.1])**.** A connected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is chordal if and only if there exists a clique tree that satisfies the clique-intersection property.

Theorem 2.9 ([14, Theorem 3.4])**.** A connected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is chordal if and only if there exists an ordering of its maximal cliques that satisfies the running-intersection property.

Figure 2.5 illustrates these concepts. A chordal graph with six nodes, shown in Figure 2.5(a), has four maximal cliques listed in Figure 2.5(b), and a clique tree satisfying the clique intersection property is shown in Figure 2.5(c). Consider maximal cliques $\mathcal{C}_1 = \{1, 2, 3\}$ and $\mathcal{C}_3 = \{1, 5, 6\}$. We have $\mathcal{C}_1 \cap \mathcal{C}_3 = \{1\} \subset \mathcal{C}_2$. Also, one can verify that the ordering $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4$ satisfies the running intersection property.



(a) (b) (c)

Figure 2.6: Sparsity patterns of 8×8 matrices corresponding to Figure 2.3(a)-Figure 2.3(c), respectively: (a) banded sparsity pattern; (b) block-arrow sparsity pattern; (c) a generic sparsity pattern.

2.3 Sparse matrix decomposition

In this section, we proceed by discussing two key applications of chordal graphs to sparse positive semidefinite matrices. As we have already seen in Example 2.7, chordal graph theory has an important consequence on sparse Cholesky factorization with zero fill-ins [17]. Indeed, the zero fill-in property of the Cholesky factorization can be used to derive a decomposition of the sparse positive semidefinite matrix cone [20–22] and a dual result on the decomposition of the sparse positive semidefinite completable matrix cone [19].

2.3.1 Sparse symmetric matrices

We first introduce some notation. Given an undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, we say a symmetric matrix $X \in \mathbb{S}^n$ has a sparsity pattern $\mathcal{E}$ if $X_{ij} = X_{ji} = 0$, whenever $i \neq j$ and $(i, j) \notin \mathcal{E}$. For example, Figure 2.2(b) shows a sparsity pattern of the matrix

$$X = \begin{bmatrix} X_{11} & X_{12} & 0 & X_{14} \\ X_{21} & X_{22} & X_{23} & X_{24} \\ 0 & X_{32} & X_{33} & X_{34} \\ X_{41} & X_{42} & X_{43} & X_{44} \end{bmatrix}, \tag{2.13}$$

and the sparsity patterns illustrated in Figure 2.6 correspond to the graphs in Figure 2.3. More precisely, we define the space of sparse symmetric matrices as

$$\mathbb{S}^n(\mathcal{E}, 0) := \{X \in \mathbb{S}^n \mid X_{ij} = X_{ji} = 0, \text{if } i \neq j \text{ and } (i, j) \notin \mathcal{E}\}.$$

Note that given $X \in \mathbb{S}^n(\mathcal{E}, 0)$, the diagonal elements X_{ii} and the off-diagonal elements X_{ij} with $(i, j) \in \mathcal{E}$ may be nonzero or zero. If $X \in \mathbb{S}^n(\mathcal{E}, 0)$ and $\mathcal{E} \subset \hat{\mathcal{E}}$, then X also has sparsity pattern $\hat{\mathcal{E}}$, *i.e.*, $X \in \mathbb{S}^n(\hat{\mathcal{E}}, 0)$. In this case, $\hat{\mathcal{E}}$ is called an *extension* or *embedding* of $\mathcal{E}$. In particular, if $\hat{\mathcal{E}}$ is chordal, this is the notion of chordal extension.

It will be convenient to refer to principle submatrices of a sparsity pattern $\mathcal{E}$. Given a clique $\mathcal{C}_k$ of $\mathcal{G}(\mathcal{V}, \mathcal{E})$, we define a matrix $E_{\mathcal{C}_k} \in \mathbb{R}^{|\mathcal{C}_k| \times n}$ with entries

$$(E_{\mathcal{C}_k})_{ij} = \begin{cases} 1, & \text{if } \mathcal{C}_k(i) = j, \\ 0, & \text{otherwise}, \end{cases} \tag{2.14}$$

where $\mathcal{C}_k(i)$ is the i-th vertex in $\mathcal{C}_k$, sorted in the natural ordering. When the clique size is n, this corresponds to an identity matrix up to a certain permutation. Given $X \in \mathbb{S}^n$, the matrix $E_{\mathcal{C}_k}$ can be used to select a principal submatrix defined by the clique $\mathcal{C}_k$, *i.e.*,

$$E_{\mathcal{C}_k} X E_{\mathcal{C}_k}^{\mathsf{T}} \in \mathbb{S}^{|\mathcal{C}_k|}.$$

In addition, the operation $E_{\mathcal{C}_k}^{\mathsf{T}} Y E_{\mathcal{C}_k}$ creates an $n \times n$ symmetric matrix from a $|\mathcal{C}_k| \times |\mathcal{C}_k|$ matrix. For example, the chordal graph in Figure 2.2(b) has a maximal clique $\mathcal{C}_1 = \{1, 2, 4\}$, and for $X \in \mathbb{S}^4$ in (2.13) and $Y \in \mathbb{S}^3$ we have

$$E_{\mathcal{C}_1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \; E_{\mathcal{C}_1} X E_{\mathcal{C}_1}^{\mathsf{T}} = \begin{bmatrix} X_{11} & X_{12} & X_{14} \\ X_{21} & X_{22} & X_{24} \\ X_{41} & X_{42} & X_{44} \end{bmatrix}, \; E_{\mathcal{C}_1}^{\mathsf{T}} Y E_{\mathcal{C}_1} = \begin{bmatrix} Y_{11} & Y_{12} & 0 & Y_{13} \\ Y_{21} & Y_{22} & 0 & Y_{23} \\ 0 & 0 & 0 & 0 \\ Y_{31} & Y_{32} & 0 & Y_{33} \end{bmatrix}.$$

Finally, we say a sparsity pattern $\mathcal{E}$ is chordal if the corresponding graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is chordal. Without loss of generality, we assume the sparsity graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is connected in our discussion of sparse matrices $\mathbb{S}^n(\mathcal{E}, 0)$. If not, the matrices in $\mathbb{S}^n(\mathcal{E}, 0)$ can be rearranged into a block-diagonal form by reordering the rows and columns. The diagonal blocks correspond to the connected components of $\mathcal{G}$ and can be analyzed individually.

2.3.2 Sparse positive semidefinite matrix cone

We define the set of positive semidefinite matrices with sparsity pattern $\mathcal{E}$ as

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

It is easy to see that $\mathbb{S}_+^n(\mathcal{E}, 0) = \mathbb{S}^n(\mathcal{E}, 0) \cap \mathbb{S}_+^n$, and that $\mathbb{S}_+^n(\mathcal{E}, 0)$ is a convex cone in $\mathbb{S}^n(\mathcal{E}, 0)$ since it is an intersection of a subspace and a convex cone. If $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is chordal, the cone $\mathbb{S}_+^n(\mathcal{E}, 0)$ can be equivalently decomposed into a sum of smaller but coupled convex cones, as stated in the following theorem.

Theorem 2.10 ([20, Theorem 2.3], [21, Theorem 4], [22, Theorem 1])**.** Let $\mathcal{G}(\mathcal{V}, \mathcal{E})$ be a chordal graph and let $\{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_t\}$ be the set of its maximal cliques. Then, $Z \in \mathbb{S}_+^n(\mathcal{E}, 0)$ if and only if there exist matrices $Z_k \in \mathbb{S}_+^{|\mathcal{C}_k|}$ for $k = 1, \ldots, t$ such that

$$Z = \sum_{k=1}^{t} E_{\mathcal{C}_k}^{\mathsf{T}} Z_k E_{\mathcal{C}_k}. \tag{2.15}$$

The "if" part of Theorem 2.10 is immediate since a sum of positive semidefinite matrices is positive semidefinite, and the "only if" part can be proved using the zero fill-in property of sparse Cholesky factorization for $Z \in \mathbb{S}_+^n(\mathcal{E}, 0)$ [15, Section 9.2], which relies on a perfect elimination ordering. A recent proof can be found in [22], which is based on simple linear algebra and the existence of at least one simplicial vertex for chordal graphs.

Note that if $Z \in \mathbb{S}^n_+(\mathcal{E}, 0)$ and $\mathcal{E}$ is non-chordal, one may still find a decomposition (2.15). However, for every non-chordal pattern $\mathcal{E}$, there exist positive semidefinite matrices $Z \in \mathbb{S}^n_+(\mathcal{E}, 0)$ that do not admit the decomposition (2.15) (see [15, Page 342]).

Theorem 2.10 has an important application in the context of sparse SDPs [23]: if the LMI constraint in the dual SDP (2.12) has a chordal sparsity pattern, then it can be equivalently replaced by a set of smaller LMIs and a set of affine equality constraints. This procedure can bring substantial computational improvement for solving large sparse SDPs if the underlying maximal cliques are small, as demonstrated in [23, 55].

Example 2.11. Consider the following positive semidefinite matrix

$$\begin{bmatrix} 2 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix} \succeq 0, \tag{2.16}$$

which has a chordal sparsity pattern corresponding to a chain of three nodes with maximal cliques $\mathcal{C}_1 = \{1, 2\}$ and $\mathcal{C}_2 = \{2, 3\}$. In this case, Theorem 2.10 guarantees a decomposition (2.15). Indeed, we have

$$E_{\mathcal{C}_1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, E_{\mathcal{C}_2} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

and

$$\begin{bmatrix} 2 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix} = E_{\mathcal{C}_1}^{\mathsf{T}} \underbrace{\begin{bmatrix} 2 & 1 \\ 1 & 0.5 \end{bmatrix}}_{\succeq 0} E_{\mathcal{C}_1} + E_{\mathcal{C}_2}^{\mathsf{T}} \underbrace{\begin{bmatrix} 0.5 & 1 \\ 1 & 2 \end{bmatrix}}_{\succeq 0} E_{\mathcal{C}_2}.$$

Example 2.12. Consider the following 3×3 LMI that involves a variable $x \in \mathbb{R}^2$

$$Z(x) := \begin{bmatrix} 2x_1 & x_1 + x_2 & 0 \\ x_1 + x_2 & 5 - x_1 - x_2 & x_1 \\ 0 & x_1 & x_2 + 1 \end{bmatrix} \succeq 0. \tag{2.17}$$

This LMI has the same sparsity pattern as (2.16), which is chordal. Consequently, Theorem 2.10 implies that (2.17) holds if and only if there exist matrices

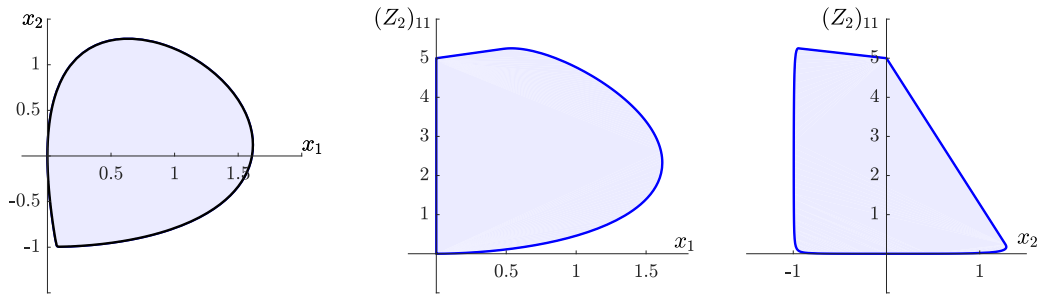
$$Z_1 := \begin{bmatrix} (Z_1)_{11} & (Z_1)_{12} \\ (Z_1)_{21} & (Z_1)_{22} \end{bmatrix} \succeq 0, \qquad Z_2 := \begin{bmatrix} (Z_2)_{11} & (Z_2)_{12} \\ (Z_2)_{21} & (Z_2)_{22} \end{bmatrix} \succeq 0,$$

such that

$$\begin{bmatrix} (Z_1)_{11} & (Z_1)_{12} & 0 \\ (Z_1)_{21} & (Z_1)_{22} + (Z_2)_{11} & (Z_2)_{12} \\ 0 & (Z_2)_{21} & (Z_2)_{22} \end{bmatrix} = Z(x).$$

After eliminating some variables, it is not difficult to see that (2.17) holds if and only if there exists $(Z_2)_{11}$ such that

$$\begin{bmatrix} 2x_1 & x_1 + x_2 \\ x_1 + x_2 & 5 - x_1 - x_2 - (Z_2)_{11} \end{bmatrix} \succeq 0, \qquad \begin{bmatrix} (Z_2)_{11} & x_1 \\ x_1 & x_2 + 1 \end{bmatrix} \succeq 0. \tag{2.18}$$



(a) (b) (c)

Figure 2.7: Joint feasible set of the decomposed LMIs in (2.18): (a) projection onto the (x_1, x_2) plane, (b) projection onto the $(x_1, (Z_2)_{11})$ plane, (c) projection onto the $(x_2, (Z_2)_{11})$ plane. Panel (a) also shows the boundary of the feasible set of the original 3×3 LMI (2.17).

Figure 2.7 shows the joint feasible set of LMIs in (2.18). As expected, the projection on the (x_1, x_2) plane is consistent with the feasible set of LMI (2.17). This confirms that the LMIs in (2.18) are equivalent to LMI (2.17). Therefore, we have equivalently decomposed a 3×3 LMI into two coupled 2×2 LMIs.

2.3.3 Positive semidefinite completable matrix cone

A concept related to the matrix decomposition above is that of positive semidefinite matrix completion. We define the cone

$$\mathbb{S}^n_+(\mathcal{E}, ?) := \mathbb{P}_{\mathbb{S}^n(\mathcal{E},0)}(\mathbb{S}^n_+),$$

given by the projection of the positive semidefinite cone onto the space of sparse matrices $\mathbb{S}^n(\mathcal{E}, 0)$ with respect to the usual Frobenius matrix norm. The question mark "?" means that the off-diagonal elements outside $\mathcal{E}$ are free to find a positive semidefinite completion. Indeed, it is not difficult to see that $X \in \mathbb{S}^n_+(\mathcal{E}, ?)$ if and only if X has a positive semidefinite completion that is consistent with $\mathcal{E}$,*i.e.*, if there exists a positive semidefinite matrix $M \succeq 0$ such that $M_{ij} = X_{ij}$ when $i = j$ and $(i, j) \in \mathcal{E}$. Therefore, we also refer to $\mathbb{S}^n_+(\mathcal{E}, ?)$ as the positive semidefinite completable cone.

For any undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, the cones $\mathbb{S}^n_+(\mathcal{E}, ?)$ and $\mathbb{S}^n_+(\mathcal{E}, 0)$ are dual to each other with respect to the trace inner product in the space $\mathbb{S}^n(\mathcal{E}, 0)$ [15, Chapter 10]. To see this, it is not difficult to verify

$$\begin{aligned}
(\mathbb{S}^n_+(\mathcal{E}, ?))^* &= \{Z \in \mathbb{S}^n(\mathcal{E}, 0) \mid \langle X, Z\rangle \geq 0, \forall X \in \mathbb{S}^n_+(\mathcal{E}, ?)\} \\
&= \{Z \in \mathbb{S}^n(\mathcal{E}, 0) \mid \left\langle \mathbb{P}_{\mathbb{S}^n(\mathcal{E},0)}(X), Z \right\rangle \geq 0, \forall X \succeq 0\} \\
&= \{Z \in \mathbb{S}^n(\mathcal{E}, 0) \mid \langle X, Z\rangle \geq 0, \forall X \succeq 0\} \\
&= \{Z \in \mathbb{S}^n(\mathcal{E}, 0) \mid Z \succeq 0\} \\
&= \mathbb{S}^n_+(\mathcal{E}, 0).
\end{aligned}$$

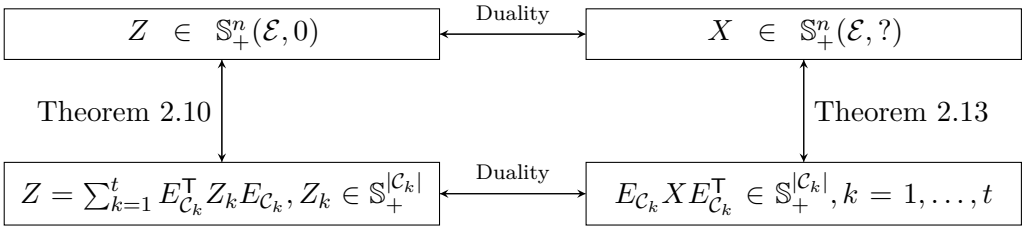


Figure 2.8: Summary of duality between $\mathbb{S}^n_+(\mathcal{E},0)$ and $\mathbb{S}^n_+(\mathcal{E},?)$ and duality between Theorem 2.10 and Theorem 2.13 for a chordal graph $\mathcal{G}(\mathcal{V},\mathcal{E})$ with maximal cliques $\mathcal{C}_1,\ldots,\mathcal{C}_t$.

For a chordal pattern, the decomposition result on $\mathbb{S}^n_+(\mathcal{E},0)$ (see Theorem 2.10) leads to the following characterization of $\mathbb{S}^n_+(\mathcal{E},?)$.

Theorem 2.13 ([19, Theorem 7])**.** Let $\mathcal{G}(\mathcal{V},\mathcal{E})$ be a chordal graph and let $\{\mathcal{C}_1,\mathcal{C}_2,\ldots,\mathcal{C}_t\}$ be the set of its maximal cliques. Then, $X \in \mathbb{S}^n_+(\mathcal{E},?)$ if and only if

$$E_{\mathcal{C}_k} X E_{\mathcal{C}_k}^{\mathsf{T}} \in \mathbb{S}_+^{|\mathcal{C}_k|}, \qquad k = 1, \ldots, t. \tag{2.19}$$

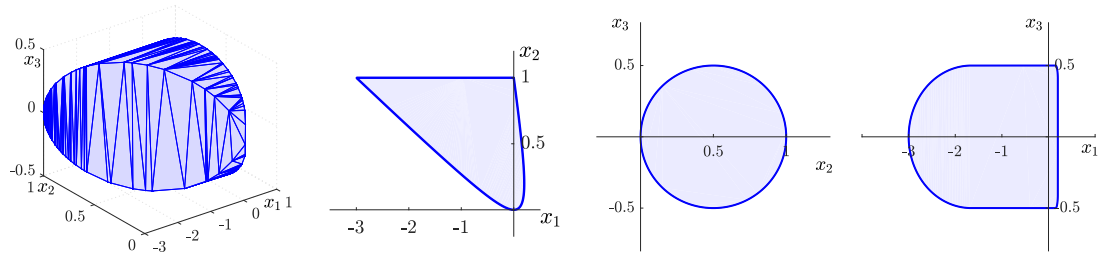
The "only if" part of Theorem 2.13 is immediate since any principal submatrix of a positive semidefinite matrix is positive semidefinite, and the "if" part relies on chordal graph properties. One can prove the "if" part based on the duality between $\mathbb{S}^n_+(\mathcal{E},0)$ and $\mathbb{S}^n_+(\mathcal{E},?)$ and Theorem 2.10 [15, Page 357]. Precisely, we have

$$\begin{aligned} X \in \mathbb{S}^n_+(\mathcal{E},?) &\Leftrightarrow \langle X, Z\rangle \geq 0, \forall Z \in \mathbb{S}^n_+(\mathcal{E},0), \\ &\Leftrightarrow \left\langle X, \sum_{k=1}^t E_{\mathcal{C}_k}^{\mathsf{T}} Z_k E_{\mathcal{C}_k} \right\rangle \geq 0, \forall Z_k \in \mathbb{S}_+^{|\mathcal{C}_k|}, \\ &\Leftrightarrow \sum_{k=1}^t \left\langle E_{\mathcal{C}_k} X E_{\mathcal{C}_k}^{\mathsf{T}}, Z_k \right\rangle \geq 0, \forall Z_k \in \mathbb{S}_+^{|\mathcal{C}_k|}, \\ &\Leftrightarrow E_{\mathcal{C}_k} X E_{\mathcal{C}_k}^{\mathsf{T}} \in \mathbb{S}_+^{|\mathcal{C}_k|}, k = 1, \ldots, t. \end{aligned}$$

In the context of SDPs, similar to the consequence of Theorem 2.10 in the dual SDP (2.12), Theorem 2.13 can be used to decompose the positive semidefinite constraint in the primal sparse SDP (2.10). These decomposition strategies are known as the *conversion* method, originally proposed in [23, 55]. Indeed, this conversion method underpins much of the recent research on sparse SDPs [24–26, 28]. Figure 2.8 shows an elegant picture, where the duality between $\mathbb{S}^n_+(\mathcal{E},0)$ and $\mathbb{S}^n_+(\mathcal{E},?)$ is mirrored in the duality between Theorem 2.10 and Theorem 2.13 for chordal graphs.

Example 2.14. Consider a symmetric matrix with partially specified entries

$$\begin{bmatrix} 2 & 1 & ? \\ 1 & 0.5 & 1 \\ ? & 1 & 2 \end{bmatrix},$$



(a) (b) (c) (d)

Figure 2.9: (a) Feasible set of the positive semidefinite completable condition in (2.20): (b) projection onto the (x_1, x_2) plane, (c) projection onto the (x_2, x_3) plane, (c) projection onto the (x_1, x_3) plane.

where the question mark "?" denotes unspecified entries. This matrix shares the same sparsity pattern with (2.17), which is chordal with maximal cliques $\mathcal{C}_1 = \{1, 2\}$ and $\mathcal{C}_2 = \{2, 3\}$, and its principal submatrices are positive semidefinite. Then, Theorem 2.13 guarantees the existence of a positive semidefinite completion by filling in the unspecified entries, shown as follows

$$\begin{bmatrix} 2 & 1 \\ 1 & 0.5 \end{bmatrix} \succeq 0, \quad \begin{bmatrix} 0.5 & 1 \\ 1 & 2 \end{bmatrix} \succeq 0, \quad X = \begin{bmatrix} 2 & 1 & 2 \\ 1 & 0.5 & 1 \\ 2 & 1 & 2 \end{bmatrix} \succeq 0.$$

Example 2.15. Consider the following 3×3 positive semidefinite completable matrix that involves a variable $x \in \mathbb{R}^3$

$$\begin{bmatrix} 1 - x_1 & x_1 + x_2 & ? \\ x_1 + x_2 & x_2 & x_2 + x_3 \\ ? & x_2 + x_3 & 2x_3 + 1 \end{bmatrix} \in \mathbb{S}^n_+(\mathcal{E}, ?). \tag{2.20}$$

Theorem 2.13 implies that (2.20) is equivalent to the following two LMIs

$$\begin{bmatrix} 1 - x_1 & x_1 + x_2 \\ x_1 + x_2 & x_2 \end{bmatrix} \succeq 0, \quad \begin{bmatrix} x_2 & x_2 + x_3 \\ x_2 + x_3 & 2x_3 + 1 \end{bmatrix} \succeq 0. \tag{2.21}$$

The feasible region (2.21) can be found by imposing $1 - x_1 + x_2 \geq 0, (1 - x_1)x_2 - (x_1 + x_2)^2 = x_2 - 3x_1x_2 - x_1^2 - x_2^2 \geq 0$ and $x_2 + 2x_3 + 1 \geq 0, x_2(2x_3 + 1) - (x_2 + x_3)^2 = x_2 - x_2^2 - x_3^2 \geq 0$. Figure 2.9 shows the feasible region of (2.21). Similar to Example 2.12, we have equivalently decomposed a 3×3 LMI into two coupled 2×2 LMIs.

2.4 Block matrices and chordal decomposition

Block matrices naturally occur both in the modeling of networked systems, where each subsystem corresponds to a block in the matrix description, and in the Gram matrix representation of SOS polynomials, where a lifted Gram matrix is used to characterize an SOS condition [52]. In this section, we extend sparse matrix notation in Section 2.3 to the block partitioned case, and present an extension of Theorem 2.10 and Theorem 2.13 to block symmetric matrices with chordal sparsity patterns.

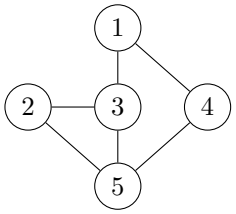


Figure 2.10: A nonchordal graph: the cycle (1-3-5-4) is of length four but with no chords.

2.4.1 Sparse block matrices

Given a vector of integers $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$, we say a *block matrix* $M \in \mathbb{R}^{N\times N}$ has α-*partitioning* with $N = \sum_{i=1}^{n} \alpha_i$ if

$$M = \begin{bmatrix} M_{11} & M_{12} & \ldots & M_{1n} \\ M_{12} & M_{22} & \ldots & M_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ M_{n1} & M_{n2} & \ldots & M_{nn} \end{bmatrix},$$

where each block entry $M_{ij} \in \mathbb{R}^{\alpha_i \times \alpha_j}, i, j = 1, \ldots, n$. We describe the sparsity pattern of α-partitioned matrix M by a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ with a node set $\mathcal{V} = \{1, 2, \ldots, n\}$

$$\mathbb{R}^{N\times N}_{\alpha}(\mathcal{E}, 0) := \{M \in \mathbb{R}^{N\times N} | M_{ij} = 0 \text{ if } i \neq j \text{ and } (j, i) \notin \mathcal{E}\}, \tag{2.22}$$

where M_{ij} represents the (i, j)-th block in M, and 0 denotes a zero block with appropriate size. If $\mathcal{G}$ is undirected, we define the space of sparse symmetric block matrices as

$$\mathbb{S}^{N}_{\alpha}(\mathcal{E}, 0) := \{M \in \mathbb{S}^{N} | M_{ij} = M_{ji}^{\mathsf{T}} = 0 \text{ if } i \neq j \text{ and } (i, j) \notin \mathcal{E}\}, \tag{2.23}$$

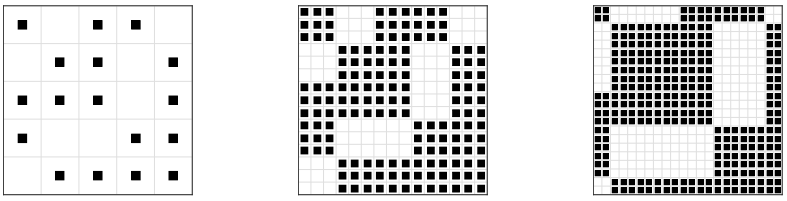
and the cone of sparse block positive semidefinite matrices as

$$\mathbb{S}^{N}_{\alpha,+}(\mathcal{E}, 0) := \{M \in \mathbb{S}^{N}_{\alpha}(\mathcal{E}, 0) \mid M \succeq 0\}. \tag{2.24}$$

Also, we define a cone $\mathbb{S}^{N}_{\alpha,+}(\mathcal{E}, ?)$ as the set of matrices in $\mathbb{S}^{N}_{\alpha}(\mathcal{E}, 0)$ that have a positive semidefinite completion,

$$\mathbb{S}^{N}_{\alpha,+}(\mathcal{E}, ?) = \mathbb{P}_{\mathbb{S}^{N}_{\alpha}(\mathcal{E},0)}(\mathbb{S}^{N}_{+}), \tag{2.25}$$

where $\mathbb{P}$ denotes the projection onto the space of sparse matrices. Figure 2.11 gives an illustration of $\mathbb{S}^{N}_{\alpha}(\mathcal{E}, 0)$ for the nonchordal graph in Figure 2.10. We can see that a graph can represent a class of sparsity patterns with different partition α. This will bring us much convenience in modeling the sparsity of networked systems and the Gram matrix representation of SOS matrices.



(a) (b) (c)

Figure 2.11: Sparsity patterns $\mathbb{S}^N_\alpha(\mathcal{E}, 0)$ with different partitions, where $\mathcal{E}$ denotes the edge set of the graph in Figure 2.10: (a) scalar case $\alpha = \{1, 1, 1, 1, 1\}$; (b) uniform block case $\alpha = \{3, 3, 3, 3, 3\}$; (c) random block case $\alpha = \{2, 8, 4, 6, 2\}$.

Remark 2.16. The notation above is a natural extension of sparse scalar matrices (see Section 2.3) to sparse block matrices with α-partition. If $\alpha = \{1, 1, \ldots, 1\}$, then the notation is reduced to the normal scalar case. Similar to Section 2.3, the definition (2.23) allows the block entry $M_{ij} = 0$ if $(i, j) \in \mathcal{E}$. We have $M \in \mathbb{S}^N_\alpha(\hat{\mathcal{E}}, 0)$ if $M \in \mathbb{S}^N_\alpha(\mathcal{E}, 0)$ and $\hat{\mathcal{E}}$ is a chordal extension of $\mathcal{E}$. Also, one can verify that for any partition α, the cones $\mathbb{S}^N_{\alpha,+}(\mathcal{E}, 0)$ and $\mathbb{S}^N_{\alpha,+}(\mathcal{E}, ?)$ are dual to each other in the space $\mathbb{S}^N_\alpha(\mathcal{E}, 0)$, meaning that

$$\mathbb{S}^N_{\alpha,+}(\mathcal{E}, ?) \equiv \{X \in \mathbb{S}^N_\alpha(\mathcal{E}, 0) \mid \langle X, Z\rangle \geq 0,\ \forall Z \in \mathbb{S}^n_{\alpha,+}(\mathcal{E}, 0)\},$$
$$\mathbb{S}^N_{\alpha,+}(\mathcal{E}, 0) \equiv \{Z \in \mathbb{S}^N_\alpha(\mathcal{E}, 0) \mid \langle Z, X\rangle \geq 0,\ \forall X \in \mathbb{S}^n_{\alpha,+}(\mathcal{E}, ?)\}.$$

2.4.2 Extension of chordal decomposition theorems

As the reader would expect, similar decomposition results may hold for $\mathbb{S}^N_{\alpha,+}(\mathcal{E}, ?)$ and $\mathbb{S}^N_{\alpha,+}(\mathcal{E}, 0)$ for chordal sparsity pattern $\mathcal{E}$. To be precise, we define a block version of index matrices.

Given a partition $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ and a clique $\mathcal{C}_k$ of $\mathcal{G}$, we define a block-wise *index matrix* $E_{\mathcal{C}_k,\alpha} \in \mathbb{R}^{|\mathcal{C}_k|_\alpha \times N}$ with $|\mathcal{C}_k|_\alpha = \sum_{i\in\mathcal{C}_k} \alpha_i$ and $N = \sum_{i=1}^n \alpha_i$ as

$$(E_{\mathcal{C}_k,\alpha})_{ij} = \begin{cases} I_{\alpha_i}, & \text{if } \mathcal{C}_k(i) = j, \\ 0_{\alpha_i\times\alpha_j}, & \text{otherwise,} \end{cases} \tag{2.26}$$

where I_{α_i} is an identity matrix of dimension α_i, $\mathcal{C}_k(i)$ denotes the i-th node in $\mathcal{C}_k$, sorted in the natural ordering, and $0_{\alpha_i\times\alpha_j}$ denotes a zero block of dimension $\alpha_i \times \alpha_j$. When $\alpha = \{1, 1, \ldots, 1\}$ (the scalar case), (2.26) is reduced to (2.14), *i.e.*, $E_{\mathcal{C}_k,\alpha} = E_{\mathcal{C}_k}$. For a uniform partition $\alpha = \{a, a, \ldots, a\}, a \in \mathbb{N}$, we also have

$$E_{\mathcal{C}_k,\alpha} = E_{\mathcal{C}_k} \otimes I_a,$$

where $\otimes$ denotes the Kronecker product. Similar to the scalar case in Section 2.3, given a block matrix $X \in \mathbb{S}^N$ with α-partition, $E_{\mathcal{C}_k,\alpha} X E^\mathsf{T}_{\mathcal{C}_k,\alpha} \in \mathbb{S}^{|\mathcal{C}_k|_\alpha}$ extracts a principal submatrix defined by the clique $\mathcal{C}_k$, and the operation $E^\mathsf{T}_{\mathcal{C}_k,\alpha} Y E_{\mathcal{C}_k,\alpha}$ *inflates* an $|\mathcal{C}_k|_\alpha \times |\mathcal{C}_k|_\alpha$ matrix into a sparse $N \times N$ matrix.

Then, we have the following two theorems, which extend Theorems 2.10 and 2.13 to the case of sparse block matrices, respectively.

Theorem 2.17 (Block-chordal decomposition theorem)**.** Let $\mathcal{G}(\mathcal{V},\mathcal{E})$ be a chordal graph with maximal cliques $\{\mathcal{C}_1,\mathcal{C}_2,\ldots,\mathcal{C}_t\}$. Given a partition $\alpha=\{\alpha_1,\alpha_2,\ldots,\alpha_n\}$ and $N=\sum_{i=1}^n\alpha_i$, then, $Z\in\mathbb{S}^N_{\alpha,+}(\mathcal{E},0)$ if and only if there exist matrices $Z_k\in\mathbb{S}_+^{|\mathcal{C}_k|_\alpha}$ for $k=1,\ldots,p$ such that

$$Z=\sum_{k=1}^{t}E_{\mathcal{C}_k,\alpha}^{\mathsf{T}}Z_kE_{\mathcal{C}_k,\alpha}. \tag{2.27}$$

Theorem 2.18 (Block-chordal completion theorem)**.** Let $\mathcal{G}(\mathcal{V},\mathcal{E})$ be a chordal graph with maximal cliques $\{\mathcal{C}_1,\mathcal{C}_2,\ldots,\mathcal{C}_t\}$. Given a partition $\alpha=\{\alpha_1,\alpha_2,\ldots,\alpha_n\}$ and $N=\sum_{i=1}^n\alpha_i$, then, $X\in\mathbb{S}^N_{\alpha,+}(\mathcal{E},?)$ if and only if

$$E_{\mathcal{C}_k,\alpha}XE_{\mathcal{C}_k,\alpha}^{\mathsf{T}}\in\mathbb{S}_+^{|\mathcal{C}_k|_\alpha},\quad k=1,\ldots,t. \tag{2.28}$$

The following example illustrates Theorem 2.17, and similar examples hold for Theorem 2.18.

Example 2.19. Consider a chain of three nodes shown in Figure 2.12(a), which is chordal with maximal cliques $\mathcal{C}_1=\{1,2\}$ and $\mathcal{C}_2=\{2,3\}$. For the scalar case, *i.e.*, $\alpha=\{1,1,1\}$, Theorem 2.17 or Theorem 2.10 guarantees the following decomposition (throughout this example, $*$ denote a real scalar number)

$$\underbrace{\left[\begin{array}{c|c|c}* & * & 0\\ \hline * & * & *\\ \hline 0 & * & *\end{array}\right]}_{\succeq 0}=\underbrace{\left[\begin{array}{c|c|c}* & * & 0\\ \hline * & * & 0\\ \hline 0 & 0 & 0\end{array}\right]}_{\succeq 0}+\underbrace{\left[\begin{array}{c|c|c}0 & 0 & 0\\ \hline 0 & * & *\\ \hline 0 & * & *\end{array}\right]}_{\succeq 0}. \tag{2.29}$$

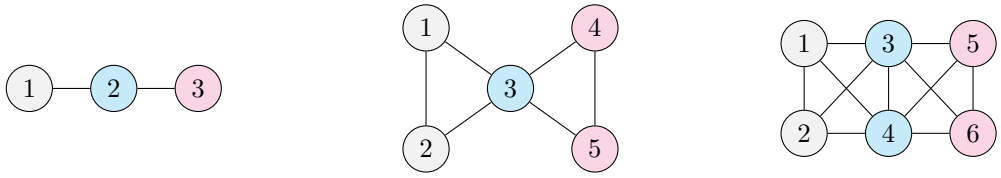
If $\alpha=\{2,1,2\}$, then Theorem 2.17 implies a block-wise decomposition as follows

$$\underbrace{\left[\begin{array}{cc|c|cc}* & * & * & 0 & 0\\ * & * & * & 0 & 0\\ \hline * & * & * & * & *\\ \hline 0 & 0 & * & * & *\\ 0 & 0 & * & * & *\end{array}\right]}_{\succeq 0}=\underbrace{\left[\begin{array}{cc|c|cc}* & * & * & 0 & 0\\ * & * & * & 0 & 0\\ \hline * & * & * & 0 & 0\\ \hline 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0\end{array}\right]}_{\succeq 0}+\underbrace{\left[\begin{array}{cc|c|cc}0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0\\ \hline 0 & 0 & * & * & *\\ \hline 0 & 0 & * & * & *\\ 0 & 0 & * & * & *\end{array}\right]}_{\succeq 0}, \tag{2.30}$$

while for another partition $\alpha=\{2,2,2\}$, Theorem 2.17 guarantees another block-wise decomposition as

$$\underbrace{\left[\begin{array}{cc|cc|cc}* & * & * & * & 0 & 0\\ * & * & * & * & 0 & 0\\ \hline * & * & * & * & * & *\\ * & * & * & * & * & *\\ \hline 0 & 0 & * & * & * & *\\ 0 & 0 & * & * & * & *\end{array}\right]}_{\succeq 0}=\underbrace{\left[\begin{array}{cc|cc|cc}* & * & * & * & 0 & 0\\ * & * & * & * & 0 & 0\\ \hline * & * & * & * & 0 & 0\\ * & * & * & * & 0 & 0\\ \hline 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0\end{array}\right]}_{\succeq 0}+\underbrace{\left[\begin{array}{cc|cc|cc}0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0\\ \hline 0 & 0 & * & * & * & *\\ 0 & 0 & * & * & * & *\\ \hline 0 & 0 & * & * & * & *\\ 0 & 0 & * & * & * & *\end{array}\right]}_{\succeq 0}. \tag{2.31}$$

In fact, one can consider (2.30) and (2.31) as an application of Theorem 2.10 to a hyper-graph shown in Figure 2.12(b) and Figure 2.12(c), respectively. This hyper-graph interpretation will be used to prove Theorems 2.17 and 2.18 in Section 2.4.3.



(a) (b) (c)

Figure 2.12: Hyper-graph interpretations of sparse matrices with different partition α in Example 2.19. (a) a chain of three nodes with $\alpha = \{1, 1, 1\}$, where maximal cliques are $\mathcal{C}_1 = \{1, 2\}$ and $\mathcal{C}_2 = \{2, 3\}$, corresponding to (2.29); (b) a hyper-graph for partition $\alpha = \{2, 1, 2\}$ where maximal cliques are $\mathcal{C}_1 = \{1, 2, 3\}$ and $\mathcal{C}_2 = \{3, 4, 5\}$, corresponding to (2.30); (c) another hyper-graph for partition $\alpha = \{2, 2, 2\}$ where maximal cliques are $\mathcal{C}_1 = \{1, 2, 3, 4\}$ and $\mathcal{C}_2 = \{3, 4, 5, 6\}$, corresponding to (2.31). In the panels (a)-(c), the nodes with the same color can be viewed in the same group.

2.4.3 Proofs of Theorems 2.17 and 2.18

Similar to the duality analysis in Section 2.3 (also, see Figure 2.8), it is not difficult to see that Theorem 2.17 and Theorem 2.18 can be proved from each other. In this section, we first present an induction proof of Theorem 2.17 for the uniform partition case $\alpha = \{a, a, \ldots, a\}, a \in \mathbb{N}$, which is adapted from [22]. Rantzer has used this induction idea to prove chordal decomposition for sparse positive definite rational matrix functions in [56]. Then, we present another proof based on a hyper-graph viewpoint as illustrated in Figure 2.12, which reduces the block-partitioned case into the normal scalar case. Accordingly, Theorems 2.10 and 2.13 can be applied in the block-partitioned case.

Induction for the uniform partition case

Here, we consider the uniform partition case $\alpha = \{a, a, \ldots, a\}, a \in \mathbb{N}$ in Theorems 2.17 and 2.18.

Lemma 2.20. [57] Given a positive semidefinite matrix with compatible blocks

$$X = \begin{bmatrix} X_{11} & X_{12} \\ X_{12}^{\mathsf{T}} & X_{22} \end{bmatrix} \succeq 0,$$

then we have $X_{11} \succeq 0, X_{22} \succeq 0$, and the column space of X_{12} is a subspace of that of X_{11}, *i.e.*, there exists a matrix H such that

$$\begin{bmatrix} I & H \\ 0 & I \end{bmatrix}^{\mathsf{T}} X \begin{bmatrix} I & H \\ 0 & I \end{bmatrix} = \begin{bmatrix} X_{11} & 0 \\ 0 & \hat{X}_{22} \end{bmatrix}.$$

Proof of Theorem 2.17: The "if" part is immediate. We prove the "only if" part by induction. We let $X \in \mathbb{S}^N_{\alpha,+}(\mathcal{E}, 0)$. If $\alpha = \{a\}$ (*i.e.*, there is only one node in $\mathcal{G}$), the result is obvious.

Assume this statement holds for graphs with $n-1\,(n \geq 2)$ nodes, *i.e.*,

$$\alpha = \underbrace{\{a, \ldots, a\}}_{n-1}.$$

Now consider a chordal graph with n nodes. Since $\mathcal{G}$ is chordal, there exists a simplicial vertex v. Without loss of generality, suppose that the maximal clique containing v, which is unique, is $\mathcal{C}_1$. We further assume $v = 1$, and $\mathcal{C}_1 \setminus \{v\} = \{2, \ldots, m\}$. Then, the matrix $X = [X_{ij}]_{n\times n}$ is of the form

$$X = \begin{bmatrix} X_{11} & X_{\{1\},\tilde{\mathcal{C}}_1} & 0 \\ X_{\tilde{\mathcal{C}}_1,\{1\}} & X_{\tilde{\mathcal{C}}_1,\tilde{\mathcal{C}}_1} & X_{\tilde{\mathcal{C}}_1,\hat{\mathcal{C}}_1} \\ 0 & X_{\hat{\mathcal{C}}_1,\tilde{\mathcal{C}}_1} & X_{\hat{\mathcal{C}}_1,\hat{\mathcal{C}}_1} \end{bmatrix},$$

where $\tilde{\mathcal{C}}_1 = \mathcal{C}_1 \setminus \{1\} = \{2, \ldots, m\}$, $\hat{\mathcal{C}}_1 = \mathcal{V} \setminus \mathcal{C}_1$. According to Lemma 2.20, we have $X_{11} \succeq 0$, and X can be transformed into

$$L^{\mathsf{T}} X L = \begin{bmatrix} X_{11} & 0 & 0 \\ 0 & \hat{X}_{\tilde{\mathcal{C}}_1,\tilde{\mathcal{C}}_1} & X_{\tilde{\mathcal{C}}_1,\hat{\mathcal{C}}_1} \\ 0 & X_{\hat{\mathcal{C}}_1,\tilde{\mathcal{C}}_1} & X_{\hat{\mathcal{C}}_1,\hat{\mathcal{C}}_1} \end{bmatrix}, \tag{2.32}$$

where

$$L = \begin{bmatrix} I & H & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix}$$

represents certain column operations. Then, we can rewrite X as

$$X = \hat{X}_1 + \hat{X}_2, \tag{2.33}$$

where

$$\hat{X}_1 = (L^{-1})^{\mathsf{T}} \begin{bmatrix} X_{11} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} L^{-1}, \quad \hat{X}_2 = (L^{-1})^{\mathsf{T}} \begin{bmatrix} 0 & 0 & 0 \\ 0 & \hat{X}_{\tilde{\mathcal{C}}_1,\tilde{\mathcal{C}}_1} & X_{\tilde{\mathcal{C}}_1,\hat{\mathcal{C}}_1} \\ 0 & X_{\hat{\mathcal{C}}_1,\tilde{\mathcal{C}}_1} & X_{\hat{\mathcal{C}}_1,\hat{\mathcal{C}}_1} \end{bmatrix} L^{-1}.$$

Both $\hat{X}_1$ and $\hat{X}_2$ are positive semidefinite since L is non-singular. Moreover, we have

- the nonzero part of $\hat{X}_1$ is $\hat{Z}_1 = E_{\mathcal{C}_1,\alpha} \hat{X}_1 E_{\mathcal{C}_1,\alpha}^{\mathsf{T}} \in \mathbb{S}_+^{|\mathcal{C}_1|_\alpha}$, which satisfies $\hat{X}_1 = E_{\mathcal{C}_1,\alpha}^{\mathsf{T}} \hat{Z}_1 E_{\mathcal{C}_1,\alpha}$;
- the lower-right part of $\hat{X}_2$ is a sparse positive semidefinite matrix corresponding to $\hat{\mathcal{G}}(\mathcal{V} \setminus \{1\}, \hat{\mathcal{E}})$ which is the subgraph induced by $\mathcal{V} \setminus \{1\}$.

Since v is simplicial, the induced subgraph $\hat{\mathcal{G}}$ is also chordal with maximal cliques as $\tilde{\mathcal{C}}_1, \mathcal{C}_2, \ldots, \mathcal{C}_t$. Therefore, applying the induction hypothesis to the lower-right part of $\hat{X}_2$ in (2.33) leads to

$$\hat{X}_2 = E_{\tilde{\mathcal{C}}_1,\alpha}^{\mathsf{T}} \tilde{Z}_1 E_{\tilde{\mathcal{C}}_1,\alpha} + \sum_{k=2}^{t} E_{\mathcal{C}_k,\alpha}^{\mathsf{T}} Z_k E_{\mathcal{C}_k,\alpha} \tag{2.34}$$

for some $\tilde{Z}_1 \in \mathbb{S}_+^{|\tilde{\mathcal{C}}_1|_\alpha}, Z_k \in \mathbb{S}_+^{|\mathcal{C}_k|_\alpha}, k = 2, \ldots, t$. Then, upon defining

$$Z_1 = \hat{Z}_1 + \begin{bmatrix} 0 & 0 \\ 0 & \tilde{Z}_1 \end{bmatrix} \in \mathbb{S}_+^{|\mathcal{C}_1|_\alpha},$$

where 0 denotes a zero block with compatible dimensions, we have found a set of matrices $Z_k \in \mathbb{S}_+^{|\mathcal{C}_k|_\alpha}, k = 1, \ldots, t$ such that

$$\begin{aligned} X &= \hat{X}_1 + \hat{X}_2 \\ &= E_{\mathcal{C}_1,\alpha}^{\mathsf{T}} \hat{Z}_1 E_{\mathcal{C}_1,\alpha} + E_{\tilde{\mathcal{C}}_1,\alpha}^{\mathsf{T}} \tilde{Z}_1 E_{\tilde{\mathcal{C}}_1,\alpha} + \sum_{k=2}^{t} E_{\mathcal{C}_k,\alpha}^{\mathsf{T}} Z_k E_{\mathcal{C}_k,\alpha} \\ &= E_{\mathcal{C}_1,\alpha}^{\mathsf{T}} \left(\hat{Z}_1 + \begin{bmatrix} 0 & 0 \\ 0 & \tilde{Z}_1 \end{bmatrix} \right) E_{\mathcal{C}_1,\alpha} + \sum_{k=2}^{t} E_{\mathcal{C}_k,\alpha}^{\mathsf{T}} Z_k E_{\mathcal{C}_k,\alpha} \\ &= \sum_{k=1}^{t} E_{\mathcal{C}_k,\alpha}^{\mathsf{T}} Z_k E_{\mathcal{C}_k,\alpha}. \end{aligned}$$

This completes the proof. ■

The key in the proof above lies on the facts that chordal graphs have at least one simplicial vertex and that the induced graph by removing the simplicial vertex remains chordal. Then, one can apply the induction hypothesis to finish the proof. This induction idea first appears in [22] which focuses on the scalar case with $\alpha = \{1, \ldots, 1\}$. We note that Rantzer independently used this induction idea to extend Theorem 2.10 to the case of positive definite rational matrix functions. With Theorem 2.17 at hand, the proof of Theorem 2.18 follows from standard duality analysis, as in the proof of Theorem 2.13.

Hyper-graph reduction for arbitrary partitions

Here, we present another proof for Theorems 2.17 and 2.18 that is suitable for an arbitrary partition α. The main strategy is based on a perspective of hyper-graphs that reduces a sparse block partitioned matrix (corresponding to the original graph) to a sparse scalar matrix (corresponding to a new hyper-graph). Consequently, the usual Theorems 2.10 and 2.13 can be applied.

Proof of Theorems 2.17 and 2.18: Let $\mathcal{G}(\mathcal{V}, \mathcal{E})$ be a chordal graph with maximal cliques $\Gamma = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_t\}$. Given a partition $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ and $N = \sum_{i=1}^{n} \alpha_i$, we define a hyper-node set corresponding to the partition α as

$$\mathcal{V}_\alpha = \{1, 2, \ldots, N\},$$

and denote a corresponding set of hyper-maximal cliques $\Lambda_\alpha = \{\mathcal{C}_1^\alpha, \mathcal{C}_2^\alpha, \ldots, \mathcal{C}_p^\alpha\}$ as

$$\mathcal{C}_k^\alpha = \left\{ \sum_{i=1}^{j-1} \alpha_i + 1, \ldots, \sum_{i=1}^{j} \alpha_i \mid j \in \mathcal{C}_k \right\}, k = 1, \ldots, t. \tag{2.35}$$

Further, we define a hyper-edge set $\mathcal{E}_\alpha \subseteq \mathcal{V}_\alpha \times \mathcal{V}_\alpha$ in the following way

$$\mathcal{E}_\alpha = \bigcup_{k=1}^{t} (\mathcal{C}_k^\alpha \times \mathcal{C}_k^\alpha). \tag{2.36}$$

Note that for the scalar case $\alpha = \{1, 1, \ldots, 1\} = \mathbb{1}$, we actually have $\mathcal{C}_k^\alpha = \mathcal{C}_k$ and $\mathcal{E}_\alpha = \mathcal{E}$.

Then, the rest is to prove the following two statements:

- *Statement 1:* the space of sparse block matrices with α-partition and pattern $\mathcal{E}$ is equivalent to the space of sparse scalar matrices with pattern $\mathcal{E}_\alpha$, *i.e.*,

$$\mathbb{S}_\alpha^N(\mathcal{E}, 0) \equiv \mathbb{S}_{\mathbb{1}}^N(\mathcal{E}_\alpha, 0), \tag{2.37}$$

- *Statement 2:* the hyper-graph $\mathcal{G}_\alpha(\mathcal{V}_\alpha, \mathcal{E}_\alpha)$ is chordal with maximal cliques $\{\mathcal{C}_1^\alpha, \mathcal{C}_2^\alpha, \ldots, \mathcal{C}_t^\alpha\}$.

If Statements 1 and 2 hold, then Theorems 2.17 and 2.18 follow immediately by applying the normal Theorems 2.10 and 2.13 to the hyper-graph $\mathcal{G}_\alpha(\mathcal{V}_\alpha, \mathcal{E}_\alpha)$, respectively.

Step 1, proof of Statement 1: $\forall M \in \mathbb{S}_\alpha^N(\mathcal{E}, 0)$, the off-diagonal block satisfies

$$M_{ij} = M_{ji}^\mathsf{T} = 0 \in \mathbb{R}^{\alpha_i \times \alpha_j} \quad \text{if } (i,j) \notin \mathcal{E}, i \neq j. \tag{2.38}$$

Meanwhile, we have

$$\begin{aligned}
&(i,j) \notin \mathcal{E} \\
\Leftrightarrow\ &\text{there exists no } k, \text{such that } i, j \in \mathcal{C}_k \\
\Leftrightarrow\ &\text{there exists no } k, \text{such that } \left\{\sum_{l=1}^{i-1}\alpha_l + 1, \ldots, \sum_{l=1}^{i}\alpha_l\right\}, \left\{\sum_{l=1}^{j-1}\alpha_l + 1, \ldots, \sum_{l=1}^{j}\alpha_l\right\} \in \mathcal{C}_k^\alpha \\
\Leftrightarrow\ &\left\{\sum_{l=1}^{i-1}\alpha_l + 1, \ldots, \sum_{l=1}^{i}\alpha_l\right\} \times \left\{\sum_{l=1}^{j-1}\alpha_l + 1, \ldots, \sum_{l=1}^{j}\alpha_l\right\} \notin \mathcal{E}^\alpha.
\end{aligned} \tag{2.39}$$

Then $M \in \mathbb{S}_{\mathbb{1}}^N(\mathcal{E}_\alpha, 0)$. Similarly, it can be seen that $\forall M \in \mathbb{S}_{\mathbb{1}}^N(\mathcal{E}_\alpha, 0)$, we have $M \in \mathbb{S}_\alpha^N(\mathcal{E}, 0)$. Therefore, Statement 1 is true.

Step 2, proof of Statement 2: Since $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is chordal, it has a clique tree $\mathcal{T} = (\Gamma, \Lambda)$ with $\Lambda \subseteq \Gamma \times \Gamma$ that satisfies the *clique intersection property* (see Theorem 2.8). Now, we consider a clique tree $\mathcal{T}_\alpha = (\Gamma_\alpha, \Lambda)$ with the same edge set as $\mathcal{T}$, where the nodes have been replaced by the hyper-maximal cliques $\Gamma_\alpha = \{\mathcal{C}_1^\alpha, \mathcal{C}_2^\alpha, \ldots, \mathcal{C}_t^\alpha\}$. We shall show that the clique intersection property of $\mathcal{T}$ is invariant for any partition $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$.

Suppose $\mathcal{C}_q^\alpha$ be a hyper-maximal clique on the path between $\mathcal{C}_i^\alpha$ and $\mathcal{C}_j^\alpha$. Then, considering the definition (2.35) and the fact $\mathcal{C}_i \cap \mathcal{C}_j \subset \mathcal{C}_q$, we have

$$\begin{aligned}
\mathcal{C}_i^\alpha \cap \mathcal{C}_j^\alpha &= \left\{ \sum_{l=1}^{t-1} \alpha_l + 1, \ldots, \sum_{l=1}^{t} \alpha_l \;\middle|\; t \in \mathcal{C}_i \text{ and } t \in \mathcal{C}_j \right\} \\
&= \left\{ \sum_{l=1}^{t-1} \alpha_l + 1, \ldots, \sum_{l=1}^{t} \alpha_l \;\middle|\; t \in \mathcal{C}_i \cap \mathcal{C}_j \right\} \\
&\subseteq \left\{ \sum_{l=1}^{t-1} \alpha_l + 1, \ldots, \sum_{l=1}^{t} \alpha_l \;\middle|\; t \in \mathcal{C}_q \right\} \\
&= \mathcal{C}_q^\alpha,
\end{aligned} \tag{2.40}$$

for any partition α. This implies the clique intersection property holds for the clique tree $\mathcal{T}_\alpha = (\Gamma_\alpha, \Lambda)$. Hence, we conclude that the hyper-graph $\mathcal{G}_\alpha = (\mathcal{V}_\alpha, \mathcal{E}_\alpha)$ is chordal with a set of maximal cliques $\{\mathcal{C}_1^\alpha, \mathcal{C}_2^\alpha, \ldots, \mathcal{C}_p^\alpha\}$, according to Theorem 2.8.

The combination of the Statements 1 and 2 completes the proof of Theorems 2.17 and 2.18. ■

We refer the reader to Figure 2.12 to an illustration of constructing hyper-graphs for a chain of three nodes, when $\alpha = \{1, 1, 1\}$, $\alpha = \{2, 1, 2\}$, or $\alpha = \{2, 2, 2\}$.

Part I

Large-scale Sparse Semidefinite Programs (SDPs)

3

Chordal decomposition in sparse semidefinite programs

The first part of this thesis is devoted to show the potential of chordal decomposition (*i.e.*, Theorem 2.10 and 2.13) in general sparse SDPs. In particular, Chapter 3 introduces a new conversion framework for large-scale sparse SDPs that is suitable for first-order algorithms, and presents ADMM algorithms to solve decomposed primal- and dual-standard form SDPs. We also describe an open-source MATLAB solver, CDCS, that implements our algorithms. In Chapter 4, we demonstrate the performance of CDCS in some analysis problems of large-scale linear networked systems, which shows the efficiency and scalability of our algorithms in CDCS.

3.1 Introduction

As stated in Section 2.1.4, semidefinite programs (SDPs) are convex optimization problems over the cone of positive semidefinite (PSD) matrices. For convenience, we restate the standard *primal form* SDP as

$$\begin{array}{ll} \underset{X}{\text{minimize}} & \langle C, X\rangle \\ \text{subject to} & \langle A_i, X\rangle = b_i, \quad i = 1, \ldots, m, \\ & X \in \mathbb{S}^n_+, \end{array} \tag{3.1}$$

and the standard *dual form* as

$$\begin{array}{ll} \underset{y,\, Z}{\text{maximize}} & \langle b, y\rangle \\ \text{subject to} & Z + \sum_{i=1}^{m} A_i\, y_i = C, \\ & Z \in \mathbb{S}^n_+, \end{array} \tag{3.2}$$

where $b \in \mathbb{R}^m$, $C \in \mathbb{S}^n$, and matrices $A_1, \ldots, A_m \in \mathbb{S}^n$ are given problem data.

SDPs have found applications in a wide range of fields, such as control theory, machine learning, combinatorics, and operations research [10]. Semidefinite programming

encompasses other common types of optimization problems, including linear, quadratic, and second-order cone programs [3]. Furthermore, many nonlinear convex constraints admit SDP relaxations that work well in practice [58]. It is well-known that small and medium-sized SDPs can be solved up to any arbitrary precision in polynomial time [58] using efficient second-order interior-point methods (IPMs) [59, 60]. However, many problems of practical interest are too large to be addressed by the current state-of-the-art interior-point algorithms, largely due to the need to compute, store, and factorize an $m \times m$ matrix at each iteration.

A common strategy to address this shortcoming is to abandon IPMs in favour of simpler first-order methods (FOMs), at the expense of reducing the accuracy of the solution. For instance, Malick *et al.* introduced regularization methods to solve SDPs based on a dual augmented Lagrangian [61]. Wen *et al.* proposed an alternating direction augmented Lagrangian method for large-scale SDPs in the dual standard form [62]. Zhao *et al.* presented an augmented Lagrangian dual approach combined with the conjugate gradient method to solve large-scale SDPs [63]. More recently, O'Donoghue *et al.* developed a first-order operator-splitting method to solve the homogeneous self-dual embedding (HSDE) of a primal-dual pair of conic programs [64]. The algorithm, implemented in the C package SCS [65], has the advantage of providing certificates of primal or dual infeasibility.

A second major approach to resolve the aforementioned scalability issues is based on the observation that the large-scale SDPs encountered in applications are often structured and/or sparse [10]. Exploiting sparsity in SDPs is an active and challenging area of research [66], with one main difficulty being that the optimal (primal) solution is typically dense even when the problem data are sparse. Nonetheless, if the *aggregate sparsity pattern* of the data is *chordal* (or has sparse *chordal extensions*), one can replace the original, large PSD constraint with a set of PSD constraints on smaller matrices, coupled by additional equality constraints [19–22]. Having reduced the size of the semidefinite variables, the converted SDP can in some cases be solved more efficiently than the original problem using standard IPMs. These ideas underly the *domain-space* and the *range-space* conversion techniques in [23, 24], implemented in the MATLAB package SparseCoLO [67].

The problem with such decomposition techniques, however, is that the addition of equality constraints to an SDP often offsets the benefit of working with smaller semidefinite cones. One possible solution is to exploit the properties of chordal sparsity patterns directly in the IPMs: Fukuda *et al.* used a positive definite completion theorem [19] to develop a primal-dual path-following method [23]; Burer proposed a nonsymmetric primal-dual IPM using Cholesky factors of the dual variable Z and maximum determinant completion of the primal variable X [68]; and Andersen *et al.* developed fast recursive algorithms to evaluate the function values and derivatives of the barrier functions for

SDPs with chordal sparsity [25]. Another attractive option is to solve the sparse SDP using FOMs: Sun *et al.* proposed a first-order splitting algorithm for partially decomposable conic programs, including SDPs with chordal sparsity [26]; Kalbat & Lavaei applied a first-order operator-splitting method to solve a special class of SDPs with fully decomposable constraints [27]; Madani *et al.* developed a highly-parallelizable first-order algorithm for sparse SDPs with inequality constraints, with applications to optimal power flow problems [28]; Dall'Anese *et al.* exploited chordal sparsity to solve SDPs with separable constraints using a distributed FOM [69].

3.1.1 Statement of results

In this chapter, we embrace the spirit of [26–28, 64, 69] and exploit sparsity in SDPs using a first-order operator-splitting method known as the *alternating direction method of multipliers* (ADMM). In contrast to the approach in [26], which requires the solution of a quadratic SDP at each iteration, our approach relies entirely on first-order methods. Moreover, our ADMM-based algorithm works for generic SDPs with chordal sparsity and has the ability to detect infeasibility, which are key advantages compared to the algorithms in [26–28, 69].

More precisely, our contributions in this chapter are:

1. We apply two chordal decomposition theorems [19, 20] to formulate *domain-space* and *range-space* conversion frameworks for the application of FOMs to standard-form SDPs with chordal sparsity. These are analogous to the conversion methods developed in [23, 24] for IPMs, but we introduce two sets of slack variables that allow for the separation of the conic and the affine constraints when using operator-splitting algorithms. To the best of our knowledge, this extension has never been presented before, and its significant potential is demonstrated in this chapter.
2. We apply ADMM to solve the domain- and range-space converted SDPs, and show that the resulting iterates of the ADMM algorithms are the same up to scaling. The iterations are computationally inexpensive: the positive semidefinite (PSD) constraint is enforced via parallel projections onto small PSD cones—a much more economical strategy than that in [26]—while imposing the affine constraints requires solving a linear system with a constant coefficient matrix, the factorization/inverse of which can be cached before iterating the algorithm.
3. We formulate the HSDE of a converted primal-dual pair of sparse SDPs. In contrast to [26–28, 69], this allows us to compute either primal and dual optimal points, or a certificate of infeasibility. We then extend the algorithm proposed in [64], showing that the structure of our HSDE can be exploited to solve a large linear

system of equations extremely efficiently through a sequence of block eliminations. As a result, we obtain an algorithm that is more efficient than the method of [64], irrespectively of whether this is used on the original primal-dual pair of SDPs (before decomposition) or on the converted problems. In the former case, the advantage comes from the application of chordal decomposition to replace a large PSD cone with a set of smaller ones. In the latter case, efficiency is gained by the proposed sequence of block eliminations.

4. We present the MATLAB solver CDCS (Cone Decomposition Conic Solver), which implements our ADMM algorithms. CDCS is the first open-source first-order solver that exploits chordal decomposition and can detect infeasible problems. We test our implementation on large-scale sparse problems in SDPLIB [70], selected sparse SDPs with nonchordal sparsity pattern [25], and randomly generated SDPs with block-arrow sparsity patterns [26]. The results demonstrate the efficiency of our algorithms compared to the interior-point solvers SeDuMi [71] and the first-order solver SCS [65].

3.1.2 Outline

The rest of this chapter is organized as follows. Section 3.2 introduces our conversion framework for sparse SDPs based on chordal decomposition. We show how to apply the ADMM to exploit domain-space and range-space sparsity in primal and dual SDPs in Section 3.3. Section 3.4 discusses the ADMM algorithm for the HSDE of SDPs with chordal sparsity. The computational complexity of our algorithms in terms of floating-point operations is discussed in Section 3.5. CDCS and our numerical experiments are presented in Section 3.6. Section 3.7 concludes this chapter.

3.2 Chordal decomposition of sparse SDPs

The sparsity pattern of the problem data for the primal-dual pair of standard-form SDPs (3.1)-(3.2) can be described using the so-called *aggregate sparsity pattern*. We say that the pair of SDPs (3.1)-(3.2) has an aggregate sparsity pattern $\mathcal{G}(\mathcal{V}, \mathcal{E})$ if

$$C \in \mathbb{S}^n(\mathcal{E}, 0) \quad \text{and} \quad A_i \in \mathbb{S}^n(\mathcal{E}, 0),\ i = 1, \ldots, m. \tag{3.3}$$

In other words, the aggregate sparsity pattern $\mathcal{G}$ is the union of the individual sparsity patterns of the data matrices C, A_1, $\ldots$, A_m. Throughout the rest of this chapter, we assume that the aggregate sparsity pattern $\mathcal{G}$ is chordal (or that a suitable chordal extension has been found), and that it has p maximal cliques $\mathcal{C}_1$, $\ldots$, $\mathcal{C}_p$. In addition, we assume that the matrices $A_1, \ldots, A_m$ are linearly independent.

It is not difficult to see that the aggregate sparsity pattern determines the sparsity pattern of any feasible dual variable Z in (3.2), *i.e.*, any dual feasible Z must have sparsity pattern $\mathcal{G}$. Similarly, while the primal variable X in (3.1) is usually dense, the value of the cost function and the equality constraints depend only on the entries X_{ij} with $(i,j) \in \mathcal{E}$, and the remaining entries simply guarantee that X is positive semidefinite. Recalling the definition of the sparse matrix cones $\mathbb{S}^n_+(\mathcal{E},?)$ and $\mathbb{S}^n_+(\mathcal{E},0)$, we can therefore recast the primal-form SDP (3.1) as

$$\begin{array}{ll} \underset{X}{\text{minimize}} & \langle C, X\rangle \\ \text{subject to} & \langle A_i, X\rangle = b_i, \quad i = 1, \ldots, m, \\ & X \in \mathbb{S}^n_+(\mathcal{E},?), \end{array} \tag{3.4}$$

and the dual-form SDP (3.2) as

$$\begin{array}{ll} \underset{y,Z}{\text{maximize}} & \langle b, y\rangle \\ \text{subject to} & Z + \sum_{i=1}^m A_i\, y_i = C, \\ & Z \in \mathbb{S}^n_+(\mathcal{E},0). \end{array} \tag{3.5}$$

This formulation was first proposed by Fukuda *et al.* [23], and was later discussed in [24–26]. Note that (3.4) and (3.5) are a primal-dual pair of linear conic problems because the cones $\mathbb{S}^n_+(\mathcal{E},?)$ and $\mathbb{S}^n_+(\mathcal{E},0)$ are dual to each other.

3.2.1 Domain-space decomposition

As we have seen in Section 2.3.2 of Chapter 2, Theorem 2.13 allows us to decompose the sparse matrix cone constraint $X \in \mathbb{S}^n_+(\mathcal{E},?)$ into p standard PSD constraints on the submatrices of X defined by the cliques $\mathcal{C}_1, \ldots, \mathcal{C}_p$. In other words,

$$X \in \mathbb{S}^n_+(\mathcal{E},?) \;\Leftrightarrow\; E_{\mathcal{C}_k} X E_{\mathcal{C}_k}^{\mathsf{T}} \in \mathbb{S}^{|\mathcal{C}_k|}_+, \quad k = 1, \ldots, p.$$

These p constraints are implicitly coupled since $E_{\mathcal{C}_l} X E_{\mathcal{C}_l}^{\mathsf{T}}$ and $E_{\mathcal{C}_q} X E_{\mathcal{C}_q}^{\mathsf{T}}$ have overlapping elements if $\mathcal{C}_l \cap \mathcal{C}_q \neq \emptyset$. Upon introducing slack variables X_k, $k = 1, \ldots, p$, we can rewrite this as

$$X \in \mathbb{S}^n_+(\mathcal{E},?) \;\Leftrightarrow\; \begin{cases} X_k = E_{\mathcal{C}_k} X E_{\mathcal{C}_k}^{\mathsf{T}}, & k = 1, \ldots, p, \\ X_k \in \mathbb{S}^{|\mathcal{C}_k|}_+, & k = 1, \ldots, p. \end{cases} \tag{3.6}$$

The primal optimization problem (3.4) is then equivalent to the SDP

$$\begin{array}{ll} \underset{X, X_1, \ldots, X_p}{\text{minimize}} & \langle C, X\rangle \\ \text{subject to} & \langle A_i, X\rangle = b_i, \qquad i = 1, \ldots, m, \\ & X_k = E_{\mathcal{C}_k} X E_{\mathcal{C}_k}^{\mathsf{T}}, \quad k = 1, \ldots, p, \\ & X_k \in \mathbb{S}^{|\mathcal{C}_k|}_+, \qquad k = 1, \ldots, p. \end{array} \tag{3.7}$$

Adopting the same terminology used in [23], we refer to (3.7) as the *domain-space* decomposition of the primal-standard-form SDP (3.1).

Remark 3.1. The main difference between the conversion method proposed in this section and that in [23, 24] is that the large matrix X is not eliminated. Instead, in the domain-space decomposition of [23, 24], X is eliminated by replacing the constraints

$$X_k = E_{\mathcal{C}_k} X E_{\mathcal{C}_k}^{\mathsf{T}}, \quad k = 1, \ldots, p,$$

with the requirement that the entries of any two different sub-matrices X_j, X_k must match if they map to the same entry in X. Precisely, this condition can be written as

$$E_{\mathcal{C}_j \cap \mathcal{C}_k} \left(E_{\mathcal{C}_k}^{\mathsf{T}} X_k E_{\mathcal{C}_k} - E_{\mathcal{C}_j}^{\mathsf{T}} X_j E_{\mathcal{C}_j} \right) E_{\mathcal{C}_j \cap \mathcal{C}_k}^{\mathsf{T}} = 0, \quad \forall j, k \text{ such that } \mathcal{C}_j \cap \mathcal{C}_k \neq \emptyset. \tag{3.8}$$

Redundant constraints in (3.8) can be eliminated using the *running intersection property* of the cliques [14, 23], and the decomposed SDP can be solved efficiently by IPMs in certain cases [23, 24]. However, applying FOMs to (3.7) effectively after the elimination of X is not straightforward because the PSD matrix variables $X_1, \ldots, X_p$ are coupled via (3.8). In [26], for example, an SDP with a quadratic objective had to be solved at each iteration to impose the PSD constraints, requiring an additional iterative solver. Even when this problem is resolved, *e.g.*, by using the algorithm of [64], the size of the KKT system enforcing the affine constraints is increased dramatically by the consensus conditions (3.8), sometimes so much that memory requirements are prohibitive on desktop computing platforms [23]. In contrast, we show in Section 3.3 that if a set of slack variables X_k are introduced in (3.6) and X is not eliminated from (3.7), then the PSD constraint can be imposed via projections onto small PSD cones. At the same time, the affine constraints require the solution of an $m \times m$ linear system of equations, as if no consensus constraints were introduced. This makes our conversion framework more suitable for FOMs than that of [23, 24], since all steps in many common operator-splitting algorithms have an efficiently computable explicit solution.

3.2.2 Range-space decomposition

A *range-space* decomposition of the dual-standard-form SDP (3.2) can be formulated by applying Theorem 2.10 to the sparse matrix cone constraint $Z \in \mathbb{S}_+^n(\mathcal{E}, 0)$ in (3.5):

$$Z \in \mathbb{S}_+^n(\mathcal{E}, 0) \Leftrightarrow Z = \sum_{k=1}^{p} E_{\mathcal{C}_k}^{\mathsf{T}} Z_k E_{\mathcal{C}_k} \text{ for some } Z_k \in \mathbb{S}_+^{|\mathcal{C}_k|},\ k = 1, \ldots, p.$$

We then introduce slack variables V_k, $k = 1, \ldots, p$ and conclude that $Z \in \mathbb{S}_+^n(\mathcal{E}, 0)$ if and only if there exists matrices $Z_k, V_k \in \mathbb{S}^{|\mathcal{C}_k|}$, $k = 1, \ldots, p$, such that

$$Z = \sum_{k=1}^{p} E_{\mathcal{C}_k}^{\mathsf{T}} V_k E_{\mathcal{C}_k}, \quad Z_k = V_k, \quad Z_k \in \mathbb{S}_+^{|\mathcal{C}_k|}, k = 1, \ldots, p.$$

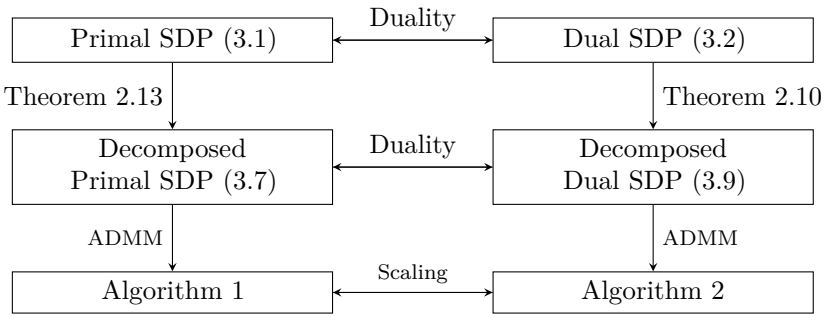


Figure 3.1: Duality between the original primal and dual SDPs, and the decomposed primal and dual SDPs.

The range-space decomposition of (3.2) is then given by

$$\begin{aligned} \underset{y,Z_1,\ldots,Z_p,V_1,\ldots,V_p}{\text{maximize}} \quad & \langle b, y\rangle \\ \text{subject to} \quad & \sum_{i=1}^{m} A_i\, y_i + \sum_{k=1}^{p} E_{\mathcal{C}_k}^{\mathsf{T}} V_k E_{\mathcal{C}_k} = C, \\ & Z_k - V_k = 0,\ k = 1,\ldots,p, \\ & Z_k \in \mathbb{S}_+^{|\mathcal{C}_k|}, \quad k = 1,\ldots,p. \end{aligned} \tag{3.9}$$

Remark 3.2. Similar comments as in Remark 3.1 hold: the slack variables $V_1, \ldots, V_p$ are essential to formulate a decomposition framework suitable for the application of FOMs. Although the domain- and range-space decompositions (3.7) and (3.9) have been derived individually, they are in fact a primal-dual pair of SDPs. The duality between the original SDPs (3.1) and (3.2) is inherited by the decomposed SDPs (3.7) and (3.9) by virtue of the duality between Theorem 2.13 and Theorem 2.10. This elegant picture is illustrated in Figure 3.1.

3.3 ADMM for domain- and range-space decompositions of sparse SDPs

In this section, we demonstrate how ADMM can be applied to solve the domain-space decomposition (3.7) and the range-space decomposition (3.9) efficiently. Furthermore, we show that the resulting domain- and range-space algorithms are equivalent, in the sense that one is just a scaled version of the other (cf. Figure 3.1).

ADMM is an operator-splitting method developed in the 1970s, and it is known to be equivalent to other operator-splitting methods such as Douglas-Rachford splitting and Spingarn's method of partial inverses; see [72] for a review. The ADMM algorithm solves the optimization problem

$$\begin{aligned} \underset{x,y}{\text{minimize}} \quad & f(x) + g(y) \\ \text{subject to} \quad & Ax + By = c, \end{aligned} \tag{3.10}$$

where f and g are convex functions, $x \in \mathbb{R}^{n_x}, y \in \mathbb{R}^{n_y}, A \in \mathbb{R}^{n_c \times n_x}, B \in \mathbb{R}^{n_c \times n_y}$ and $c \in \mathbb{R}^{n_c}$. Given a penalty parameter $\rho > 0$ and a dual multiplier $z \in \mathbb{R}^{n_c}$, the ADMM algorithm finds a saddle point of the augmented Lagrangian

$$\mathcal{L}_\rho(x, y, z) := f(x) + g(y) + z^T (Ax + By - c) + \frac{\rho}{2} \|Ax + By - c\|^2$$

by minimizing $\mathcal{L}$ with respect to the primal variables x and y separately, followed by a dual variable update:

$$x^{(n+1)} = \arg\min_x \ \mathcal{L}_\rho(x, y^{(n)}, z^{(n)}), \tag{3.11a}$$

$$y^{(n+1)} = \arg\min_y \ \mathcal{L}_\rho(x^{(n+1)}, y, z^{(n)}), \tag{3.11b}$$

$$z^{(n+1)} = z^{(n)} + \rho\,(Ax^{(n+1)} + By^{(n+1)} - c). \tag{3.11c}$$

The superscript (n) indicates that a variable is fixed to its value at the n-th iteration. Note that since z is fixed in (3.11a) and (3.11b), one may equivalently minimize the modified Lagrangian

$$\hat{\mathcal{L}}_\rho(x, y, z) := f(x) + g(y) + \frac{\rho}{2} \left\| Ax + By - c + \frac{1}{\rho} z \right\|^2.$$

Under very mild conditions, the ADMM converges to a solution of (3.10) with a rate $\mathcal{O}(\frac{1}{n})$ [72, Section 3.2]. ADMM is particularly suitable when (3.11a) and (3.11b) have closed-form expressions, or can be solved efficiently. Moreover, splitting the minimization over x and y often allows distributed and/or parallel implementations of steps (3.11a)–(3.11c).

3.3.1 Vectorized forms

Throughout this section, $\delta_{\mathcal{K}}(x)$ will denote the indicator function of a set $\mathcal{K}$, *i.e.*,

$$\delta_{\mathcal{K}}(x) := \begin{cases} 0, & \text{if } x \in \mathcal{K}, \\ +\infty, & \text{otherwise.} \end{cases}$$

To simplify notation, we write δ_0 when $\mathcal{K} \equiv \{0\}$.

To ease the exposition further, we consider the usual vectorized forms of (3.7) and (3.9). Specifically, we let $\operatorname{vec} : \mathbb{S}^n \to \mathbb{R}^{n^2}$ be the usual operator mapping a matrix to the stack of its columns and define the vectorized data

$$c := \operatorname{vec}(C), \quad A := \begin{bmatrix} \operatorname{vec}(A_0) & \dots & \operatorname{vec}(A_m) \end{bmatrix}^{\mathsf{T}}.$$

Note that the assumption that $A_1, \dots, A_m$ are linearly independent matrices means that A has full row rank. For all $k = 1, \dots, p$, we also introduce the vectorized variables

$$x := \operatorname{vec}(X), \quad x_k := \operatorname{vec}(X_k), \quad z_k := \operatorname{vec}(Z_k), \quad v_k := \operatorname{vec}(V_k),$$

and define "entry-selector" matrices $H_k := E_{\mathcal{C}_k} \otimes E_{\mathcal{C}_k}$ for $k = 1, \ldots, p$ that project x onto the subvectors $x_1, \ldots, x_p$, *i.e.*, such that

$$x_k = \operatorname{vec}(X_k) = \operatorname{vec}(E_{\mathcal{C}_k} X E_{\mathcal{C}_k}^{\mathsf{T}}) = H_k x.$$

Note that for each $k = 1, \ldots, p$, the rows of H_k are orthonormal, and that the matrix $H_k^{\mathsf{T}} H_k$ is diagonal. Upon defining $\operatorname{mat}(\cdot) := \operatorname{vec}^{-1}$ and

$$\mathcal{S}_k := \left\{ x \in \mathbb{R}^{|\mathcal{C}_k|^2} : \operatorname{mat}(x) \in \mathbb{S}_+^{|\mathcal{C}_k|} \right\},$$

such that $x_k \in \mathcal{S}_k$ if and only if $X_k \in \mathbb{S}_+^{|\mathcal{C}_k|}$, we can rewrite (3.7) as

$$\begin{array}{lll} \underset{x, x_1, \ldots, x_p}{\text{minimize}} & \langle c, x \rangle & \\ \text{subject to} & Ax = b, & \\ & x_k = H_k x, & k = 1, \ldots, p, \\ & x_k \in \mathcal{S}_k, & k = 1, \ldots, p, \end{array} \tag{3.12}$$

while (3.9) becomes

$$\begin{array}{lll} \underset{y, z_1, \ldots, z_p, v_1, \ldots, v_p}{\text{maximize}} & \langle b, y \rangle & \\ \text{subject to} & A^{\mathsf{T}} y + \sum_{k=1}^{p} H_k^{\mathsf{T}} v_k = c, & \\ & z_k - v_k = 0, & k = 1, \ldots, p, \\ & z_k \in \mathcal{S}_k, & k = 1, \ldots, p. \end{array} \tag{3.13}$$

3.3.2 ADMM for the domain-space decomposition

We start by moving the constraints $Ax = b$ and $x_k \in \mathcal{S}_k$ in (3.12) to the objective using the indicator functions $\delta_0(\cdot)$ and $\delta_{\mathcal{S}_k}(\cdot)$, respectively, *i.e.*, we write

$$\begin{array}{ll} \underset{x, x_1, \ldots, x_p}{\text{minimize}} & \langle c, x \rangle + \delta_0 \left(Ax - b \right) + \sum_{k=1}^{p} \delta_{\mathcal{S}_k}(x_k) \\ \text{subject to} & x_k = H_k x, \quad k = 1, \ldots, p. \end{array} \tag{3.14}$$

This problem is in the standard form for the application of ADMM. Given a penalty parameter $\rho > 0$ and a Lagrange multiplier λ_k for each constraint $x_k = H_k x$, $k = 1, \ldots, p$, we consider the (modified) augmented Lagrangian

$$\begin{aligned} \mathcal{L}(x, x_1, \ldots, x_k, \lambda_1, \ldots, \lambda_k) := \langle c, x \rangle + \delta_0 \left(Ax - b \right) \\ + \sum_{k=1}^{p} \left[\delta_{\mathcal{S}_k}(x_k) + \frac{\rho}{2} \left\| x_k - H_k x + \frac{1}{\rho} \lambda_k \right\|^2 \right], \end{aligned} \tag{3.15}$$

and group the variables as $\mathcal{X} := \{x\}$, $\mathcal{Y} := \{x_1, \ldots, x_p\}$, and $\mathcal{Z} := \{\lambda_1, \ldots, \lambda_p\}$. According to (3.11), each iteration of the ADMM requires the minimization of the

Lagrangian in (3.15) with respect to the $\mathcal{X}$- and $\mathcal{Y}$-blocks separately, followed by an update of the multipliers $\mathcal{Z}$. At each step, the variables not being optimized over are fixed to their most current value. Note that splitting the primal variables x, $x_1, \ldots, x_p$ in the two blocks $\mathcal{X}$ and $\mathcal{Y}$ defined above is essential to solving the $\mathcal{X}$ and $\mathcal{Y}$ minimization subproblems (3.11a) and (3.11b); more details will be given in Remark 3.3 after describing the $\mathcal{Y}$-minimization step in Section 3.3.2.

Minimization over $\mathcal{X}$

Minimizing the augmented Lagrangian (3.15) over $\mathcal{X}$ is equivalent to the equality-constrained quadratic program

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \langle c, x\rangle + \dfrac{\rho}{2}\displaystyle\sum_{k=1}^{p}\left\| x_k^{(n)} - H_k x + \frac{1}{\rho}\lambda_k^{(n)}\right\|^2 \\ \text{subject to} & Ax = b. \end{array} \tag{3.16}$$

Letting ρy be the multiplier for the equality constraint (we scale the multiplier by ρ for convenience), and defining

$$D := \sum_{k=1}^{p} H_k^{\mathsf{T}} H_k, \tag{3.17}$$

the optimality conditions for (3.16) can be written as the KKT system

$$\begin{bmatrix} D & A^{\mathsf{T}} \\ A & 0 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \sum_{k=1}^{p} H_k^{\mathsf{T}}\left(x_k^{(n)} + \rho^{-1}\lambda_k^{(n)}\right) - \rho^{-1}c \\ b \end{bmatrix}. \tag{3.18}$$

Recalling that the product $H_k^{\mathsf{T}} H_k$ is a diagonal matrix for all $k = 1, \ldots, p$ we conclude that so is D, and since A has full row rank by assumption (3.18) can be solved efficiently, for instance by block elimination. In particular, eliminating x shows that the only matrix to be inverted/factorized is

$$AD^{-1}A^{\mathsf{T}} \in \mathbb{S}^m. \tag{3.19}$$

Incidentally, we note that the first-order algorithms of [62, 64] require the factorization of a similar matrix with the same dimension. Since this matrix is the same at every iteration, its Cholesky factorization (or any other factorization of choice) can be computed and cached before starting the ADMM iterations. For some families of SDPs, such as the SDP relaxation of MaxCut problems and sum-of-squares (SOS) feasibility problems [48], the matrix $AD^{-1}A^{\mathsf{T}}$ is diagonal, so solving (3.18) is inexpensive even when the SDPs are very large. If factorizing $AD^{-1}A^{\mathsf{T}}$ is too expensive, the linear system (3.18) can alternatively be solved by an iterative method, such as the conjugate gradient method [73].

Minimization over $\mathcal{Y}$

Minimizing the augmented Lagrangian (3.15) over $\mathcal{Y}$ is equivalent to solving p independent conic problems of the form

$$\begin{array}{ll} \underset{x_k}{\text{minimize}} & \left\| x_k - H_k x^{(n+1)} + \rho^{-1}\lambda_k^{(n)} \right\|^2 \\ \text{subject to} & x_k \in \mathcal{S}_k. \end{array} \tag{3.20}$$

In terms of the original matrix variables $X_1, \ldots, X_p$, each of these p sub-problems amounts to a projection onto a PSD cone. More precisely, if $\mathbb{P}_{\mathbb{S}_+^{|\mathcal{C}_k|}}$ denotes the projection onto the PSD cone $\mathbb{S}_+^{|\mathcal{C}_k|}$, we have

$$x_k^{(n+1)} = \operatorname{vec}\left\{ \mathbb{P}_{\mathbb{S}_+^{|\mathcal{C}_k|}} \left[\operatorname{mat}\left(H_k x^{(n+1)} - \rho^{-1}\lambda_k^{(n)} \right) \right] \right\}. \tag{3.21}$$

Since the size of each cone $\mathbb{S}_+^{|\mathcal{C}_k|}$ is small for typical sparse SDPs and the projection onto it can be computed with an eigenvalue decomposition, the variables $x_1, \ldots, x_p$ can be updated efficiently. Moreover, the computation can be carried out in parallel. In contrast, the algorithms for generic SDPs developed in [61, 62, 64] require projections onto the (much larger) original PSD cone $\mathbb{S}_+^n$.

Remark 3.3. As anticipated in Remark 3.1, retaining the global variable x in the domain-space decomposed SDP to enforce the consensus constraints between the entries of the subvectors $x_1, \ldots, x_p$ (*i.e.*, $x_k = H_k x$) is fundamental. In fact, it allowed us to separate the conic constraints from the affine constraints in (3.12) when applying the splitting strategy of ADMM, making the minimization over $\mathcal{Y}$ easy to compute and parallelizable. In contrast, when x is eliminated as in the conversion method of [23, 24], the conic constraints and the affine constraints cannot be easily decoupled when applying the first-order splitting method: in [26] a quadratic SDP had to be solved at each iteration, which limits its scalability.

Updating the multipliers $\mathcal{Z}$

The final step in the n-th ADMM iteration is to update the multipliers $\lambda_1, \ldots, \lambda_p$ with the usual gradient ascent rule: for each $k = 1, \ldots, p$,

$$\lambda_k^{(n+1)} = \lambda_k^{(n)} + \rho\left(x_k^{(n+1)} - H_k x^{(n+1)} \right). \tag{3.22}$$

This computation is inexpensive and easily parallelized.

Algorithm 1 ADMM for the domain-space decomposition of sparse primal-form SDPs

Set $\rho > 0$, $\epsilon_{\text{tol}} > 0$, a maximum number of iterations $n_{\max}$, and initial guesses $x^{(0)}$, $x_1^{(0)}, \ldots, x_p^{(0)}$, $\lambda_1^{(0)}, \ldots, \lambda_p^{(0)}$.
Data preprocessing: chordal extension, chordal decomposition, and factorization of the KKT system (3.18).
for $n = 1, 2, \ldots, n_{\max}$ **do**
    Compute $x^{(n)}$ using (3.18).
    for $k = 1, \ldots, p$ **do**
        Compute $x_k^{(n)}$ using (3.21).
        Compute $\lambda_k^{(n)}$ using (3.22).
    end for
        Update the residuals $\epsilon_{\text{c}}, \epsilon_\lambda$.
        if maximize$(\epsilon_{\text{c}}, \epsilon_\lambda) \leq \epsilon_{\text{tol}}$ **then**
            break
        end if
    end for

Stopping conditions

The ADMM algorithm is stopped after the n-th iteration if the relative primal/dual error measures.

$$\epsilon_{\text{c}} = \frac{\left(\sum_{k=1}^{p} \left\| x_k^{(n)} - H_k x^{(n)} \right\|^2\right)^{1/2}}{\max\left\{ \left(\sum_{k=1}^{p} \left\| x_k^{(n)} \right\|^2\right)^{1/2}, \left(\sum_{k=1}^{p} \left\| H_k x^{(n)} \right\|^2\right)^{1/2} \right\}}, \tag{3.23a}$$

$$\epsilon_\lambda = \rho \left(\sum_{k=1}^{p} \left\| x_k^{(n)} - x_k^{(n-1)} \right\|^2\right)^{1/2} \left(\sum_{k=1}^{p} \left\| \lambda_k^{(n)} \right\|^2\right)^{-1/2}, \tag{3.23b}$$

are smaller than a specified tolerance, ϵ_{tol}. The reader is referred to [72] for a detailed discussion of stopping conditions for ADMM algorithms. In conclusion, a primal-form SDP with domain-space decomposition (3.12) can be solved using the steps summarized in Algorithm 1.

3.3.3 ADMM for the range-space decomposition

An ADMM algorithm similar to Algorithm 1 can be developed for the range-space decomposition (3.13) of a dual-standard-form sparse SDP. As in Section 3.3.2, we start by moving all but the consensus equality constraints $z_k = v_k$, $k = 1, \ldots, p$, to the objective using indicator functions. This leads to

$$\begin{aligned} \text{minimize} \quad & -\langle b, y \rangle + \delta_0\left(c - A^{\mathsf{T}} y - \sum_{k=1}^{p} H_k^{\mathsf{T}} v_k\right) + \sum_{k=1}^{p} \delta_{\mathcal{S}_k}(z_k) \\ \text{subject to} \quad & z_k = v_k, \quad k = 1, \ldots, p. \end{aligned} \tag{3.24}$$

Given a penalty parameter $\rho > 0$ and a Lagrange multiplier λ_k for each of the constraints $z_k = v_k$, $k = 1, \ldots, p$, we consider the (modified) augmented Lagrangian

$$\begin{aligned}\mathcal{L}(y, v_1, \ldots, v_p, z_1, \ldots, z_p, \lambda_1, \ldots, \lambda_p) := &-\langle b, y\rangle \\ &+ \delta_0\left(c - A^\mathsf{T} y - \sum_{k=1}^p H_k^\mathsf{T} v_k\right) + \sum_{k=1}^p \left[\delta_{\mathcal{S}_k}(z_k) + \frac{\rho}{2}\left\|z_k - v_k + \frac{1}{\rho}\lambda_k\right\|^2\right], \quad (3.25)\end{aligned}$$

and consider three groups of variables, $\mathcal{X} := \{y, v_1, \ldots, v_p\}$, $\mathcal{Y} := \{z_1, \ldots, z_p\}$, and $\mathcal{Z} := \{\lambda_1, \ldots, \lambda_p\}$. Similar to Section 3.3.2, each iteration of the ADMM algorithm for (3.13) consists of minimizations over $\mathcal{X}$ and $\mathcal{Y}$, and an update of the multipliers $\mathcal{Z}$. Each of these steps admits an inexpensive closed-form solution, as we demonstrate next.

Minimization over $\mathcal{X}$

Minimizing (3.25) over block $\mathcal{X}$ is equivalent to solving the equality-constrained quadratic program

$$\begin{aligned}\underset{y, v_1, \ldots, v_p}{\text{minimize}} \quad & -\langle b, y\rangle + \frac{\rho}{2}\sum_{k=0}^p \left\|z_k^{(n)} - v_k + \frac{1}{\rho}\lambda_k^{(n)}\right\|^2 \\ \text{subject to} \quad & c - A^\mathsf{T} y - \sum_{k=1}^p H_k^\mathsf{T} v_k = 0. \end{aligned} \quad (3.26)$$

Let ρx be the multiplier for the equality constraint. After some algebra, the optimality conditions for (3.26) can be written as the KKT system

$$\begin{bmatrix} D & A^\mathsf{T} \\ A & 0 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} c - \sum_{k=1}^p H_k^\mathsf{T}\left(z_k^{(n)} + \rho^{-1}\lambda_k^{(n)}\right) \\ -\rho^{-1} b \end{bmatrix}, \quad (3.27)$$

plus a set of p uncoupled equations for the variables v_k,

$$v_k = z_k^{(n)} + \frac{1}{\rho}\lambda_k^{(n)} + H_k x, \quad k = 1, \ldots, p. \quad (3.28)$$

The KKT system (3.27) is the same as (3.18) after rescaling $x \mapsto -x$, $y \mapsto -y$, $c \mapsto \rho^{-1}c$ and $b \mapsto \rho b$. Consequently, the numerical cost of (3.26) is the same as in Section 3.3.2 plus the cost of (3.28), which is inexpensive and can be parallelized. Moreover, as in Section 3.3.2, the factors of the coefficient matrix required to solve the KKT system (3.27) can be pre-computed and cached before iterating the ADMM algorithm.

Minimization over $\mathcal{Y}$

As in Section 3.3.2, the variables $z_1, \ldots, z_p$ are updated with p independent projections,

$$z_k^{(n+1)} = \text{vec}\left\{\mathbb{P}_{\mathbb{S}_+^{|\mathcal{C}_k|}}\left[\text{mat}\left(v_k^{(n+1)} - \rho^{-1}\lambda_k^{(n)}\right)\right]\right\}, \quad (3.29)$$

where $\mathbb{P}_{\mathbb{S}_+^{|\mathcal{C}_k|}}$ denotes projection on the PSD cone $\mathbb{S}_+^{|\mathcal{C}_k|}$. Again, these projections can be computed efficiently and in parallel.

Remark 3.4. As anticipated in Section 3.2.2, introducing the set of slack variables v_k and the consensus constraints $z_k = v_k$, $k = 1, \ldots, p$ is essential to obtain an efficient algorithm for range-space decomposed SDPs. The reason is that the splitting strategy of the ADMM decouples the conic and affine constraints, and the conic variables can be updated using the simple conic projection (3.29).

Updating the multipliers $\mathcal{Z}$

The multipliers λ_k, $k = 1, \ldots, p$, are updated (possibly in parallel) with the computationally inexpensive gradient ascent rule

$$\lambda_k^{(n+1)} = \lambda_k^{(n)} + \rho \left(z_k^{(n+1)} - v_k^{(n+1)} \right). \tag{3.30}$$

Stopping conditions

Similar to Section 3.3.2, we stop our ADMM algorithm after the n-th iteration if the relative primal/dual error measures

$$\epsilon_{\mathrm{c}} = \frac{\left(\sum_{k=1}^{p} \left\| z_k^{(n)} - v_k^{(n)} \right\|^2 \right)^{1/2}}{\max \left\{ \left(\sum_{k=1}^{p} \left\| z_k^{(n)} \right\|^2 \right)^{1/2}, \left(\sum_{k=1}^{p} \left\| v_k^{(n)} \right\|^2 \right)^{1/2} \right\}}, \tag{3.31a}$$

$$\epsilon_{\lambda} = \rho \left(\sum_{k=1}^{p} \left\| z_k^{(n)} - z_k^{(n-1)} \right\|^2 \right)^{1/2} \left(\sum_{k=1}^{p} \left\| \lambda_k^{(n)} \right\|^2 \right)^{-1/2}, \tag{3.31b}$$

are smaller than a specified tolerance, ϵ_{tol}. The ADMM algorithm to solve the range-space decomposition (3.13) of a dual-form sparse SDP is summarized in Algorithm 2.

3.3.4 Equivalence between the primal and dual ADMM algorithms

Since the computational cost of (3.28) is the same as (3.22), all ADMM iterations for the dual-form SDP with range-space decomposition (3.13) have the same cost as those for the primal-form SDP with domain-space decomposition (3.12), plus the cost of (3.30). However, if one minimizes the dual augmented Lagrangian (3.25) over $z_1, \ldots, z_p$ *before* minimizing it over $y, v_1, \ldots, v_p$, then (3.28) can be used to simplify the multiplier update equations to

$$\lambda_k^{(n+1)} = \rho H_k x^{(n+1)}, \quad k = 1, \ldots, p. \tag{3.32}$$

Given that the products $H_1 x, \ldots, H_p x$ have already been computed to update $v_1, \ldots, v_p$ in (3.28), updating the multipliers $\lambda_1, \ldots, \lambda_p$ requires only a scaling operation. Then, after

Algorithm 2 ADMM for the range-space decomposition of sparse dual-form SDPs

Set $\rho > 0$, $\epsilon_{\text{tol}} > 0$, a maximum number of iterations $n_{\max}$ and initial guesses $y^{(0)}$, $v_1^{(0)}$, ..., $v_p^{(0)}$, $\lambda_1^{(0)}$, ..., $\lambda_p^{(0)}$.
Data preprocessing: chordal extension, chordal decomposition, and factorization of the KKT system (3.27).
for $n = 1, 2, \ldots, n_{\max}$ **do**
  for $k = 1, \ldots, p$ **do**
    Compute $z_k^{(n)}$ using (3.29).
  end for
  Compute $y^{(n)}, x$ using (3.27).
  for $k = 1, \ldots, p$ **do**
    Compute $v_k^{(n)}$ using (3.28).
    Compute $\lambda_k^{(n)}$ using (3.32).
  end for
  Update the residuals ϵ_{c} and ϵ_λ.
  if maximize$(\epsilon_{\text{c}}, \epsilon_\lambda) \leq \epsilon_{\text{tol}}$ **then**
    break
  end if
end for

swapping the order of $\mathcal{X}$- and $\mathcal{Y}$-block minimization of (3.25) and recalling that (3.18) and (3.27) are scaled versions of the same KKT system, the ADMM algorithms for the primal and dual standard form SDPs can be considered scaled versions of each other; see Figure 3.1 for an illustration. In fact, the equivalence between ADMM algorithms for the original (*i.e.*, before chordal decomposition) primal and dual SDPs was already noted in [74].

Remark 3.5. Although the iterates of Algorithm 1 and Algorithm 2 are the same up to scaling, the convergence performance of these two algorithms differ in practice because first-order methods are sensitive to the scaling of the problem data and of the iterates.

3.4 Homogeneous self-dual embedding of domain- and range-space decomposed SDPs

Algorithms 1 and 2, as well as other first-order algorithms that exploit chordal sparsity [26–28], can solve feasible problems, but cannot detect infeasibility in their current formulation. Although some recent ADMM methods resolve this issue [75, 76], an elegant way to deal with an infeasible primal-dual pair of SDPs—which we pursue here—is to solve their homogeneous self-dual embedding (HSDE) [77].

The essence of the HSDE method is to search for a non-zero point in the intersection of a convex cone and a linear space; this is non-empty because it always contains the origin, meaning that the problem is always feasible. Given such a non-zero point, one can either recover optimal primal and dual solutions of the original pair of optimization

problems, or construct a certificate of primal or dual infeasibility. HSDEs have been widely used to develop IPMs for SDPs [71, 78], and more recently O'Donoghue *et al.* have proposed an operator-splitting method to solve the HSDE of general conic programs [64].

In this section, we formulate the HSDE of the domain- and range-space decomposed SDPs (3.12) and (3.13), which is a primal-dual pair of SDPs. We also apply ADMM to solve this HSDE; in particular, we extend the algorithm of [64] to exploit chordal sparsity without increasing its computational cost (at least to leading order) compared to Algorithms 1 and 2.

3.4.1 Homogeneous self-dual embedding

To simplify the formulation of the HSDE of the decomposed (vectorized) SDPs (3.12) and (3.13), we let $\mathcal{S} := \mathcal{S}_1 \times \cdots \times \mathcal{S}_p$ be the Cartesian product of PSD cones and define

$$s := \begin{bmatrix} x_1 \\ \vdots \\ x_p \end{bmatrix}, \quad z := \begin{bmatrix} z_1 \\ \vdots \\ z_p \end{bmatrix}, \quad t := \begin{bmatrix} v_1 \\ \vdots \\ v_p \end{bmatrix}, \quad H := \begin{bmatrix} H_1 \\ \vdots \\ H_p \end{bmatrix}.$$

When strong duality holds, the tuple $(x^*, s^*, y^*, t^*, z^*)$ is optimal if and only if all of the following conditions hold:

1. (x^*, s^*) is primal feasible, *i.e.*, $Ax^* = b$, $s^* = Hx^*$, and $s^* \in \mathcal{S}$. For reasons that will become apparent below, we introduce slack variables $r^* = 0$ and $w^* = 0$ of appropriate dimensions and rewrite these conditions as

$$Ax^* - r^* = b, \quad s^* + w^* = Hx^*, \quad s^* \in \mathcal{S}, \quad r^* = 0, \qquad w^* = 0. \tag{3.33}$$

2. (y^*, t^*, z^*) is dual feasible, *i.e.*, $A^\mathsf{T} y^* + H^\mathsf{T} t^* = c$, $z^* = t^*$, and $z^* \in \mathcal{S}$. Again, it is convenient to introduce a slack variable $h^* = 0$ of appropriate size and write

$$A^\mathsf{T} y^* + H^\mathsf{T} t^* + h^* = c, \qquad z^* - t^* = 0, \qquad z^* \in \mathcal{S}, \qquad h^* = 0. \tag{3.34}$$

3. The duality gap is zero, *i.e.*

$$c^\mathsf{T} x^* - b^\mathsf{T} y^* = 0. \tag{3.35}$$

The idea behind the HSDE [77] is to introduce two non-negative and complementary variables τ and κ and embed the optimality conditions (3.33), (3.34) and (3.35) into the linear system $v = Qu$ with u, v and Q defined as

$$u := \begin{bmatrix} x \\ s \\ y \\ t \\ \tau \end{bmatrix}, \quad v := \begin{bmatrix} h \\ z \\ r \\ w \\ \kappa \end{bmatrix}, \quad Q := \begin{bmatrix} 0 & 0 & -A^\mathsf{T} & -H^\mathsf{T} & c \\ 0 & 0 & 0 & I & 0 \\ A & 0 & 0 & 0 & -b \\ H & -I & 0 & 0 & 0 \\ -c^\mathsf{T} & 0 & b^\mathsf{T} & 0 & 0 \end{bmatrix}. \tag{3.36}$$

Any nonzero solution of this embedding can be used to recover an optimal solution for (3.7) and (3.9), or provide a certificate for primal or dual infeasibility, depending on the values of τ and κ; details are omitted for brevity, and the interested reader is referred to [64].

The decomposed primal-dual pair of (vectorized) SDPs (3.12)-(3.13) can therefore be recast as the self-dual conic feasibility problem

$$\begin{aligned} \text{find} \quad & (u, v) \\ \text{subject to} \quad & v = Qu, \\ & (u, v) \in \mathcal{K} \times \mathcal{K}^*, \end{aligned} \tag{3.37}$$

where, writing $n_d = \sum_{k=1}^{p} |\mathcal{C}_k|^2$ for brevity, $\mathcal{K} := \mathbb{R}^{n^2} \times \mathcal{S} \times \mathbb{R}^m \times \mathbb{R}^{n_d} \times \mathbb{R}_+$ is a cone and $\mathcal{K}^* := \{0\}^{n^2} \times \mathcal{S} \times \{0\}^m \times \{0\}^{n_d} \times \mathbb{R}_+$ is its dual.

3.4.2 A simplified ADMM algorithm

The feasibility problem (3.37) is in a form suitable for the application of ADMM, and moreover steps (3.11a)-(3.11c) can be greatly simplified by virtue of its self-dual character [64]. Specifically, the n-th iteration of the simplified ADMM algorithm for (3.37) proposed in [64] consists of the following three steps, where $\mathbb{P}_{\mathcal{K}}$ denotes projection onto the cone $\mathcal{K}$:

$$\hat{u}^{(n+1)} = (I + Q)^{-1} \left(u^{(n)} + v^{(n)} \right), \tag{3.38a}$$

$$u^{(n+1)} = \mathbb{P}_{\mathcal{K}} \left(\hat{u}^{(n+1)} - v^{(n)} \right), \tag{3.38b}$$

$$v^{(n+1)} = v^{(n)} - \hat{u}^{(n+1)} + u^{(n+1)}. \tag{3.38c}$$

Note that (3.38b) is inexpensive, since $\mathcal{K}$ is the cartesian product of simple cones (zero, free and non-negative cones) and small PSD cones, and can be efficiently carried out in parallel. The third step is also computationally inexpensive and parallelizable. On the contrary, even when the preferred factorization of $I + Q$ (or its inverse) is cached before starting the iterations, a direct implementation of (3.38a) may require substantial computational effort because

$$Q \in \mathbb{S}^{n^2+2n_d+m+1}$$

is a very large matrix (*e.g.*, $n^2+2n_d+m+1 = 2\,360\,900$ for problem `rs365` in Section 3.6.3). Yet, it is evident from (3.36) that Q is highly structured and sparse, and these properties can be exploited to speed up step (3.38a) using a series of block-eliminations and the matrix inversion lemma [3, Section C.4.3].

Solving the "outer" linear system

The affine projection step (3.38a) requires the solution of a linear system (which we refer to as the "outer" system for reasons that will become clear below) of the form

$$\begin{bmatrix} M & \zeta \\ -\zeta^{\mathsf{T}} & 1 \end{bmatrix} \begin{bmatrix} \hat{u}_1 \\ \hat{u}_2 \end{bmatrix} = \begin{bmatrix} \omega_1 \\ \omega_2 \end{bmatrix}, \tag{3.39}$$

where

$$M := \begin{bmatrix} I & -\hat{A}^{\mathsf{T}} \\ \hat{A} & I \end{bmatrix}, \quad \zeta := \begin{bmatrix} \hat{c} \\ -\hat{b} \end{bmatrix}, \quad \hat{A} := \begin{bmatrix} A & 0 \\ H & -I \end{bmatrix}, \quad \hat{c} := \begin{bmatrix} c \\ 0 \end{bmatrix}, \quad \hat{b} := \begin{bmatrix} b \\ 0 \end{bmatrix} \tag{3.40}$$

and we have split

$$u^{(n)} + v^{(n)} = \begin{bmatrix} \omega_1 \\ \omega_2 \end{bmatrix}. \tag{3.41}$$

Note that $\hat{u}_2$ and ω_2 are scalars. Eliminating $\hat{u}_2$ from the first block equation in (3.39) yields

$$(M + \zeta\zeta^{\mathsf{T}})\hat{u}_1 = \omega_1 - \omega_2\zeta, \tag{3.42a}$$

$$\hat{u}_2 = \omega_2 + \zeta^{\mathsf{T}}\hat{u}_1. \tag{3.42b}$$

Moreover, applying the matrix inversion lemma [3, Section C.4.3] to (3.42a) shows that

$$\hat{u}_1 = \left[I - \frac{(M^{-1}\zeta)\zeta^{\mathsf{T}}}{1 + \zeta^{\mathsf{T}}(M^{-1}\zeta)} \right] M^{-1}\left(\omega_1 - \omega_2\zeta\right). \tag{3.43}$$

Note that the vector $M^{-1}\zeta$ and the scalar $1 + \zeta^{\mathsf{T}}(M^{-1}\zeta)$ depend only on the problem data, and can be computed before starting the ADMM iterations (since M is quasi-definite it can be inverted, and any symmetric matrix obtained as a permutation of M admits an LDL factorization). Instead, recalling from (3.41) that $\omega_1 - \omega_2\zeta$ changes at each iteration because it depends on the iterates $u^{(n)}$ and $v^{(n)}$, the vector $M^{-1}\left(\omega_1 - \omega_2\zeta\right)$ must be computed at each iteration. Consequently, computing $\hat{u}_1$ and $\hat{u}_2$ requires the solution of an "inner" linear system for the vector $M^{-1}\left(\omega_1 - \omega_2\zeta\right)$, followed by inexpensive vector inner products and scalar-vector operations in (3.43) and (3.42b).

Solving the "inner" linear system

Recalling the definition of M from (3.40), the "inner" linear system to calculate $\hat{u}_1$ in (3.43) has the form

$$\begin{bmatrix} I & -\hat{A}^{\mathsf{T}} \\ \hat{A} & I \end{bmatrix} \begin{bmatrix} \sigma_1 \\ \sigma_2 \end{bmatrix} = \begin{bmatrix} \nu_1 \\ \nu_2 \end{bmatrix}. \tag{3.44}$$

Here, σ_1 and σ_2 are the unknowns and represent suitable partitions of the vector $M^{-1}(\omega_1 - \omega_2\zeta)$ in (3.43), which is to be calculated, and we have split

$$\omega_1 - \omega_2\zeta = \begin{bmatrix} \nu_1 \\ \nu_2 \end{bmatrix}.$$

Applying block elimination to remove σ_1 from the second equation in (3.44), we obtain

$$(I + \hat{A}^\mathsf{T}\hat{A})\sigma_1 = \nu_1 + \hat{A}^\mathsf{T}\nu_2, \tag{3.45a}$$

$$\sigma_2 = -\hat{A}\sigma_1 + \nu_2. \tag{3.45b}$$

Recalling the definition of $\hat{A}$ and recognizing that

$$D = H^\mathsf{T}H = \sum_{k=1}^{p} H_k^\mathsf{T} H_k$$

is a diagonal matrix, as already noted in Section 3.3.2, we also have

$$I + \hat{A}^\mathsf{T}\hat{A} = \begin{bmatrix} (I + D + A^\mathsf{T}A) & -H^\mathsf{T} \\ -H & 2I \end{bmatrix}.$$

Block elimination can therefore be used once again to solve (3.45a), and simple algebraic manipulations show that the only matrix to be factorized (or inverted) is

$$I + \frac{1}{2}D + A^\mathsf{T}A \in \mathbb{S}^{n^2}. \tag{3.46}$$

Note that this matrix depends only on the problem data and the chordal decomposition, so it can be factorized/inverted before starting the ADMM iterations. In addition, it is of the "diagonal plus low rank" form because $A \in \mathbb{R}^{m \times n^2}$ with $m < n^2$ (in fact, often $m \ll n^2$). This means that the matrix inversion lemma can be used to reduce the size of the matrix to factorize/invert even further: letting $P = I + \frac{1}{2}D$ be the diagonal part of (3.46), we have

$$(P + A^\mathsf{T}A)^{-1} = P^{-1} - P^{-1}A^\mathsf{T}(I + AP^{-1}A^\mathsf{T})^{-1}AP^{-1}.$$

In summary, after a series of block eliminations and applications of the matrix inversion lemma, step (3.38a) of the ADMM algorithm for (3.37) only requires the solution of an $m \times m$ linear system of equations with coefficient matrix

$$I + A\left(I + \frac{1}{2}D\right)^{-1} A^\mathsf{T} \in \mathbb{S}^m, \tag{3.47}$$

plus a sequence of matrix-vector, vector-vector, and scalar-vector multiplications. A detailed count of floating-point operations is given in Section 3.5.

Stopping conditions

The ADMM algorithm described in the previous section can be stopped after the n-th iteration if a primal-dual optimal solution or a certificate of primal and/or dual infeasibility is found, up to a specified tolerance ϵ_{tol}. As noted in [64], rather than checking the convergence of the variables u and v, it is desirable to check the convergence of the original primal and dual SDP variables using the primal and dual residual error measures normally considered in interior-point algorithms [71]. For this reason, we employ different stopping conditions than those used in Algorithms 1 and 2, which we define below using the following notational convention: we denote the entries of u and v in (3.36) that correspond to x, y, τ, and z, respectively, by u_x, u_y, u_τ, and v_z.

If $u_\tau^{(n)} > 0$ at the n-th iteration of the ADMM algorithm, we take

$$x^{(n)} = \frac{u_x^{(n)}}{u_\tau^{(n)}}, \qquad y^{(n)} = \frac{u_y^{(n)}}{u_\tau^{(n)}}, \qquad z^{(n)} = \frac{H^{\mathsf{T}} v_z^{(n)}}{u_\tau^{(n)}} \tag{3.48}$$

as the candidate primal-dual solutions, and define the relative primal residual, dual residual, and duality gap as

$$\epsilon_{\text{p}} := \frac{\|Ax^{(n)} - b\|_2}{1 + \|b\|_2}, \tag{3.49a}$$

$$\epsilon_{\text{d}} := \frac{\|A^{\mathsf{T}} y^{(n)} + z^{(n)} - c\|_2}{1 + \|c\|_2}, \tag{3.49b}$$

$$\epsilon_{\text{g}} := \frac{|c^{\mathsf{T}} x^{(n)} - b^{\mathsf{T}} y^{(n)}|}{1 + |c^{\mathsf{T}} x^{(n)}| + |b^{\mathsf{T}} y^{(n)}|}. \tag{3.49c}$$

Also, we define the residual in consensus constraints as

$$\epsilon_{\text{c}} := \max\{(3.23\text{a}), (3.31\text{a})\}. \tag{3.50}$$

We terminate the algorithm if $\max\{\epsilon_{\text{p}}, \epsilon_{\text{d}}, \epsilon_{\text{g}}, \epsilon_{\text{c}}\}$ is smaller than ϵ_{tol}. If $u_\tau^{(n)} = 0$, instead, we terminate the algorithm if

$$\max\left\{\|Au_x^{(n)}\|_2 + \frac{c^{\mathsf{T}} u_x^{(n)}}{\|c\|_2}\,\epsilon_{\text{tol}},\ \|A^{\mathsf{T}} u_y^{(n)} + H^{\mathsf{T}} v_z^{(n)}\|_2 - \frac{b^{\mathsf{T}} u_y^{(n)}}{\|b\|_2}\,\epsilon_{\text{tol}}\right\} \leq 0. \tag{3.51}$$

Certificates of primal or dual infeasibility (with tolerance ϵ_{tol}) are then given, respectively, by the points $u_y^{(n)}/(b^{\mathsf{T}} u_y^{(n)})$ and $-u_x^{(n)}/(c^{\mathsf{T}} u_x^{(n)})$. These stopping criteria are similar to those used by many other conic solvers, and coincide with those used in SCS [65] except for the addition of the residual in the consensus constraints (3.50). The complete ADMM algorithm to solve the HSDE of the primal-dual pair of domain- and range-space decomposed SDPs is summarized in Algorithm 3.

Algorithm 3 ADMM for the HSDE of sparse SDPs with chordal decomposition

Set $\epsilon_{\text{tol}} > 0$, a maximum number of iterations $n_{\max}$ and initial guesses $\hat{u}^{(0)}$, $u^{(0)}$, $v^{(0)}$.
Data preprocessing: chordal extension, chordal decomposition and factorization of the matrix in (3.47).
for $n = 1, \ldots, n_{\max}$ **do**
    Compute $\hat{u}^{(n+1)}$ using the sequence of block eliminations (3.39)-(3.47).
    Compute $u^{(n+1)}$ using (3.38b).
    Compute $v^{(n+1)}$ using (3.38c).
    if $u_\tau^{(n)} > 0$ **then**
        Compute $\epsilon_{\text{p}}, \epsilon_{\text{d}}, \epsilon_{\text{g}}, \epsilon_{\text{c}}$.
        if maximize$\{\epsilon_{\text{p}}, \epsilon_{\text{d}}, \epsilon_{\text{g}}, \epsilon_{\text{c}}\} \leq \epsilon_{\text{tol}}$ **then**
            break
        end if
    else
        if (3.51) holds **then**
            break
        end if
    end if
end for

3.5 Complexity analysis via flop count

The computational complexity of each iteration of Algorithms 1-3 can be assessed by counting the total number of required *floating-point operations* (flops)—that is, the number of additions, subtractions, multiplications, or divisions of two floating-point numbers [3, Appendix C.1.1]—as a function of problem dimensions. For (3.18) and (3.27) we have

$$A \in \mathbb{R}^{m \times n^2}, \quad b \in \mathbb{R}^m, \quad c \in \mathbb{R}^{n^2}, \quad D \in \mathbb{S}^{n^2}, \quad H_k \in \mathbb{R}^{|\mathcal{C}_k|^2 \times n^2} \text{ for } k = 1, \ldots, p,$$

while the dimensions of the variables are

$$x \in \mathbb{R}^{n^2}, \quad y \in \mathbb{R}^m, \quad x_k,\, \lambda_k \in \mathbb{R}^{|\mathcal{C}_k|^2} \text{ for } k = 1, \ldots, p.$$

In this section, we count the flops in Algorithms 1–3 as a function of m, n, p, and $n_d = \sum_{k=1}^{p} |\mathcal{C}_k|^2$. We do not consider the sparsity in the problem data, both for simplicity and because sparsity is problem-dependent. Thus, the matrix-vector product Ax is assumed to cost $(2n^2 - 1)m$ flops (for each row, we need n^2 multiplications and $n^2 - 1$ additions), while $A^{\mathsf{T}}y$ is assumed to cost $(2m - 1)n^2$ flops. In practice, of course, these matrix-vector products may require significantly fewer flops if A is sparse, and sparsity *should* be exploited in any implementation to reduce computational cost. The only exception that we make concerns the matrix-vector products $H_k x$ and $H_k^{\mathsf{T}} x_k$ because each H_k, $k = 1, \ldots, p$, is an "entry-selector" matrix that extracts the subvector $x_k \in \mathbb{R}^{|\mathcal{C}_k|^2}$ from $x \in \mathbb{R}^{n^2}$. Hence, the operations $H_k x$ and $H_k^{\mathsf{T}} x_k$ require no actual matrix multiplications but only indexing operations (plus, possibly, making copies of floating-point numbers depending on the implementation), so they cost no flops according to our definition.

However, we do not take into account that the vectors $H_k^\mathsf{T} x_k \in \mathbb{R}^{n^2}$, $k = 1, \ldots, p$, are often sparse, because their sparsity depends on the particular problem at hand. It follows from these considerations that computing the summation $\sum_{k=1}^p H_k^\mathsf{T} x_k$ costs $(p-1)n^2$ flops.

Using these rules, in Section 3.8, we prove the following results.

Proposition 3.6. Given the Cholesky factorization of $AD^{-1}A^\mathsf{T} = LL^\mathsf{T}$, where L is lower triangular, solving the linear systems (3.18) and (3.27) via block elimination costs $(4m + p + 3)n^2 + 2m^2 + 2n_d$ flops.

Proposition 3.7. Given the constant vector $\hat{\zeta} := (M^{-1}\zeta)/(1 + \zeta^\mathsf{T} M^{-1}\zeta) \in \mathbb{R}^{n^2+2n_d+m}$ and the Cholesky factorization $I + A(I + \frac{1}{2}D)^{-1}A^\mathsf{T} = LL^\mathsf{T}$, where L is lower triangular, solving (3.38a) using the sequence of block eliminations (3.39)–(3.47) requires $(8m + 2p + 11)n^2 + 2m^2 + 7m + 21n_d - 1$ flops.

These propositions reveal that the computational complexity of the affine projections in Algorithms 1 and 2, which amount to solving the linear systems (3.18) and (3.27), is comparable to that of the affine projection (3.38a) in Algorithm 3. In fact, since typically $m \ll n^2$, we expect that the affine projection step of Algorithm 3 will be only approximately twice as expensive as the corresponding step in Algorithms 1 and 2 in terms of the number of flops, and therefore also in terms of CPU time (the numerical results presented in Table 3.9, Section 3.6.4, will confirm this expectation).

Similarly, the following result (also proved in Section 3.8) guarantees that the leading-order costs of the conic projections in Algorithms 1–3 are identical and, importantly, depend only on the size and number of the maximal cliques in the chordal decomposition, *not* on the dimension n of the original PSD cone in (3.1)–(3.1).

Proposition 3.8. The computational costs of the conic projections in Algorithms 1–3 require $\mathcal{O}(\sum_{k=1}^p |\mathcal{C}_k|^3)$ floating-point operations.

In particular, the computational burden grows as a linear function of the number of cliques when their size is fixed, and as a cubic function of the clique size.

Finally, we emphasize that Propositions 3.6–3.8 suggest that Algorithms 1–3 should solve a primal-dual pair of sparse SDPs more efficiently than the general-purpose ADMM method for conic programs of [64], irrespective of whether this is used before or after chordal decomposition. In the former case, the benefit comes from working with smaller PSD cones: one block-elimination in equation (28) of [64] allows solving affine projection step (3.38a) in $\mathcal{O}(mn^2)$ flops, which is typically comparable to the flop count of Propositions 3.6 and 3.7, but the conic projection step costs $\mathcal{O}(n^3)$ flops, which for typical sparse SDPs is significantly larger than $\mathcal{O}(\sum_{k=1}^p |\mathcal{C}_k|^3)$. In the latter case, instead, the conic projection (3.38b) costs the same for all methods, but projecting the iterates onto the affine constraints becomes much more expensive according to our flop count when the sequences of block eliminations described in Section 3.4 is not exploited fully.

3.6 Implementation and numerical experiments

We implemented Algorithms 1–3 in an open-source MATLAB solver which we call CDCS (Cone Decomposition Conic Solver). We refer to our implementation of Algorithms 1–3 as CDCS-primal, CDCS-dual and CDCS-hsde, respectively. This section briefly describes CDCS and presents numerical results on sparse SDPs from SDPLIB [70], large and sparse SDPs with nonchordal sparsity patterns from [25], and randomly generated SDPs with block-arrow sparsity pattern. Such problems have also been used as benchmarks in [25, 26].

In order to highlight the advantages of chordal decomposition, first-order algorithms, and their combination, the three algorithms in CDCS are compared to the interior-point solver SeDuMi [71], and to the single-threaded direct implementation of the first-order algorithm of [64] provided by the conic solver SCS [65]. All experiments were carried out on a PC with a 2.8 GHz Intel Core i7 CPU and 8GB of RAM and the solvers were called with termination tolerance $\epsilon_{\text{tol}} = 10^{-3}$, number of iterations limited to 2 000, and their default remaining parameters. The purpose of comparing CDCS to a low-accuracy IPM is to demonstrate the advantages of combining FOMs with chordal decomposition, while a comparison to the high-performance first-order conic solver SCS highlights the advantages of chordal decomposition alone. When possible, accurate solutions ($\epsilon_{\text{tol}} = 10^{-8}$) were also computed using SeDuMi; these can be considered "exact", and used to assess how far the solution returned by CDCS is from optimality. Note that tighter tolerances could be used with CDCS and SCS to obtain a more accurate solution, at the expense of increasing the number of iterations required to meet the convergence requirements. Finally, SparseCoLO [67] was used as a preprocessor for SeDuMi, which implemented the conversion techniques [23, 24] to exploit chordal sparsity.

3.6.1 CDCS

To the best of our knowledge, CDCS is the first open-source first-order conic solver that exploits chordal decomposition for the PSD cones and is able to handle infeasible problems. Cartesian products of the following cones are supported: the cone of free variables $\mathbb{R}^n$, the non-negative orthant $\mathbb{R}^n_+$, second-order cones, and PSD cones. The current implementation is written in MATLAB and can be downloaded from

https://github.com/oxfordcontrol/cdcs.

Note that although many steps of Algorithms 1–3 can be carried out in parallel, our implementation is sequential. Interfaces with the optimization toolboxes YALMIP [79] and SOSTOOLS [80] are also available.

Implementation details

CDCS applies chordal decomposition to all PSD cones. Following [15], the sparsity pattern of each PSD cone is chordal extended using the MATLAB function `chol` to compute a symbolic Cholesky factorization of the approximate minimum-degree permutation of the cone's adjacency matrix, returned by the MATLAB function `symamd`. The maximal cliques of the chordal extension are then computed using a `.mex` function from SparseCoLO [67].

As far as the steps of our ADMM algorithms are concerned, projections onto the PSD cone are performed using the MATLAB routine `eig`, while projections onto other supported cones only use vector operations. The Cholesky factors of the $m \times m$ linear system coefficient matrix (permuted using `symamd`) are cached before starting the ADMM iterations. The permuted linear system is solved at each iteration using the routines `cs_lsolve` and `cs_ltsolve` from the CSparse library [81]. CDCS solves the decomposed problems (3.12) and/or (3.13) using any of Algorithms 1–3, and then attempts to construct a primal-dual solution of the original SDPs (3.1) and (3.1) with a maximum determinant completion routine (see [23, Section 2], [15, Section 10.2]) adapted from SparseCoLO [67].

Adaptive penalty strategy

While the ADMM algorithms proposed in the previous sections converge independently of the choice of penalty parameter ρ, in practice its value strongly influences the number of iterations required for convergence. Unfortunately, analytic results for the optimal choice of ρ are not available except for very special problems [82]. Consequently, in order to improve the convergence rate and make performance less dependent on the choice of ρ, CDCS employs the dynamic adaptive rule.

$$\rho^{(h+1)} = \begin{cases} \mu\,\rho^{(h)} & \text{if } \|\epsilon_{\mathrm{p}}^{(h)}\|_2 \geq \nu \|\epsilon_{\mathrm{d}}^{(h)}\|_2, \\ \mu^{-1}\rho^{(h)} & \text{if } \|\epsilon_{\mathrm{d}}^{(h)}\|_2 \geq \nu \|\epsilon_{\mathrm{p}}^{(h)}\|_2, \\ \rho^{(k)} & \text{otherwise.} \end{cases}$$

Here, $\epsilon_{\mathrm{p}}^{(h)}$ and $\epsilon_{\mathrm{d}}^{(h)}$ are the primal and dual residuals at the h-th iteration, while μ and ν are parameters no smaller than 1. Note that since ρ does not enter any of the matrices being factorized/inverted, updating its value is computationally inexpensive. The adaptive penalty strategy can bring certain convergence improvements in practice, but in theory it is difficult to prove the convergence of the ADMM algorithm when ρ varies by iteration [72].

The idea of the rule above is to adapt ρ to balance the convergence of the primal and dual residuals to zero; more details can be found in [72, Section 3.4.1]. Typical choices for the parameters (the default in CDCS) are $\mu = 2$ and $\nu = 10$ [72].

Scaling the problem data

The relative scaling of the problem data also affects the convergence rate of ADMM algorithms. CDCS scales the problem data after the chordal decomposition step using a strategy similar to [64]. In particular, the decomposed SDPs (3.12) and (3.13) can be rewritten as:

$$\begin{array}{lll} \underset{\hat{x}}{\text{minimize}} & \hat{c}^{\mathsf{T}}\hat{x} & \underset{\hat{y},\hat{z}}{\text{maximize}} \quad \hat{b}^{\mathsf{T}}\hat{y} \\ \text{subject to} & \hat{A}\hat{x} = \hat{b} & \text{subject to} \quad \hat{A}^{\mathsf{T}}\hat{y} + \hat{z} = \hat{c} \\ & \hat{x} \in \mathbb{R}^{n^2} \times \mathcal{K}, & \hat{z} \in \{0\}^{n^2} \times \hat{\mathcal{K}}^*, \end{array} \tag{3.52a,b}$$

where

$$\hat{x} = \begin{bmatrix} x \\ s \end{bmatrix}, \quad \hat{y} = \begin{bmatrix} y \\ t \end{bmatrix}, \quad \hat{z} = \begin{bmatrix} 0 \\ z \end{bmatrix}, \quad \hat{c} = \begin{bmatrix} c \\ 0 \end{bmatrix}, \quad \hat{b} = \begin{bmatrix} b \\ 0 \end{bmatrix}, \quad \hat{A} = \begin{bmatrix} A & 0 \\ H & -I \end{bmatrix}.$$

CDCS solves the scaled decomposed problems

$$\begin{array}{lll} \underset{\hat{x}}{\text{minimize}} & \sigma(D\hat{c})^{\mathsf{T}}\bar{x} & \underset{\hat{y},\hat{z}}{\text{maximize}} \quad \rho(Eb)^{\mathsf{T}}\bar{y} \\ \text{subject to} & E\hat{A}D\bar{x} = \rho E\hat{b} & \text{subject to} \quad D\hat{A}^{\mathsf{T}}E\bar{y} + \bar{z} = \sigma D\hat{c} \\ & \bar{x} \in \mathbb{R}^{n^2} \times \mathcal{K}, & \bar{z} \in \{0\}^{n^2} \times \mathcal{K}^*, \end{array} \tag{3.53a,b}$$

obtained by scaling vectors $\hat{b}$ and $\hat{c}$ by positive scalars ρ and σ, and the primal and dual equality constraints by positive definite, diagonal matrices D and E. Note that such a rescaling does not change the sparsity pattern of the problem. As already observed in [64], a good choice for E, D, σ and ρ is such that the rows of $\bar{A}$ and $\bar{b}$ have Euclidean norm close to one, and the columns of $\bar{A}$ and $\bar{c}$ have similar norms. If D and D^{-1} are chosen to preserve membership to the cone $\mathbb{R}^{n^2} \times \mathcal{K}$ and its dual, respectively (how this can be done is explained in [64, Section 5]), an optimal point for (3.52) can be recovered from the solution of (3.53):

$$\hat{x}^* = \frac{D\bar{x}^*}{\rho}, \quad \hat{y}^* = \frac{E\bar{y}^*}{\sigma}, \quad \hat{z}^* = \frac{D^{-1}\bar{z}^*}{\sigma}.$$

3.6.2 Sparse SDPs from SDPLIB

Our first experiment is based on large-scale benchmark problems from SDPLIB [70]: two Lovász ϑ number SDPs (`theta1` and `theta2`), two infeasible SDPs (`infd1` and `infd2`), two MaxCut problems (`maxG11` and `maxG32`), and two SDP relaxations of box-constrained quadratic programs (`qpG11` and `qpG51`). Table 3.1 reports the dimensions of these problems, as well as chordal decomposition details. Problems `theta1` and `theta2` are dense, so have only one maximal clique; all other problems are sparse and have many maximal cliques of size much smaller than the original cone.

Table 3.1: Details of the SDPLIB problems considered in this chapter.

	Small		Infeasible		Large and sparse			
	theta1	theta2	infd1	infd2	maxG11	maxG32	qpG11	qpG51
Original cone size, n	50	100	30	30	800	2 000	1 600	2 000
Affine constraints, m	104	498	10	10	800	2 000	800	1 000
Number of cliques, p	1	1	1	1	598	1 499	1 405	1 675
Maximum clique size	50	100	30	30	24	60	24	304
Minimum clique size	50	100	30	30	5	5	1	1

Table 3.2: Results for two small SDPs, theta1 and theta2, in SDPLIB.

	theta1			theta2		
	Time (s)	# Iter.	Objective	Time (s)	# Iter.	Objective
SeDuMi (high)	0.281	14	23.00	1.216	15	32.88
SeDuMi (low)	0.161	8	23.00	0.650	8	32.88
SCS (direct)	0.057	140	22.99	0.244	200	32.89
CDCS-primal	0.297	163	22.92	0.618	188	32.94
CDCS-dual	0.284	154	22.83	0,605	178	32.89
CDCS-hsde	0.230	156	23.03	0.392	118	32.88

The numerical results are summarized in Tables 3.2–3.5. Table 3.2 shows that the small dense SDPs theta1 and theta2, were solved in approximately the same CPU time by all solvers. Note that since these problems only have one maximal clique, SCS and CDCS-hsde use similar algorithms, and performance differences are mainly due to the implementation (most notably, SCS is written in C). Table 3.3 confirms that CDCS-hsde successfully detects infeasible problems, while CDCS-primal and CDCS-dual do not have this ability.

The CPU time, number of iterations and terminal objective value for the four large-scale sparse SDPs maxG11, maxG32, qpG11 and qpG51 are listed in Table 3.4. All algorithms in CDCS were faster than either SeDuMi or SCS, especially for problems with smaller maximum clique size as one would expect in light of the complexity analysis of Section 3.5. Notably, CDCS solved maxG11, maxG32, and qpG11 in less than 100 s, a speedup of approximately 9×, 43×, and 66× over SCS. In addition, even though FOMs are only

Table 3.3: Results for two infeasible SDPs in SDPLIB. An objective value of +Inf denotes infeasiblity. Results for the primal-only and dual-only algorithms in CDCS are not reported since they cannot detect infeasibility.

	infp1			infp2		
	Time (s)	# Iter.	Objective	Time (s)	# Iter.	Objective
SeDuMi (high)	0.127	2	+Inf	0.033	2	+Inf
SeDuMi (low)	0.120	2	+Inf	0.031	2	+Inf
SCS (direct)	0.067	20	+Inf	0.031	20	+Inf
CDCS-hsde	0.109	118	+Inf	0.114	101	+Inf

Table 3.4: Results for four large sparse SDPs in SDPLIB, `maxG11`, `maxG32`, `qpG11` and `qpG51`.

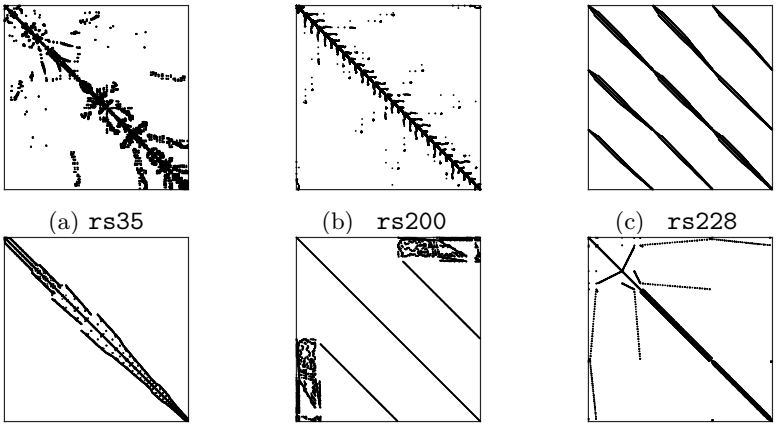
	`maxG11`			`maxG32`		
	Time (s)	# Iter.	Objective	Time (s)	# Iter.	Objective
SeDuMi (high)	88.9	13	629.2	1 266	14	1 568
SeDuMi (low)	48.7	7	628.7	624	7	1 566
SCS (direct)	93.9	1 080	629.1	2 433	2 000	1 568
CDCS-primal	22.2	230	629.5	84	311	1 569
CDCS-dual	16.9	220	629.2	61	205	1 567
CDCS-hsde	10.9	182	629.3	56	291	1 568
	`qpG11`			`qpG51`		
	Time (s)	# Iter.	Objective	Time (s)	# Iter.	Objective
SeDuMi (high)	650	14	2 449	1 895	22	1 182
SeDuMi (low)	357	8	2 448	1 530	18	1 182
SCS (direct)	1 065	2 000	2 449	2 220	2 000	1 288
CDCS-primal	29	249	2 450	482	1 079	1 145
CDCS-dual	21	193	2 448	396	797	1 201
CDCS-hsde	16	219	2 449	865	2 000	1 182

Table 3.5: Average CPU time per iteration (in seconds) for the SDPs from SDPLIB.

	`theta1`	`theta2`	`maxG11`	`maxG32`	`qpG11`	`qpG51`
SCS (direct)	4.0×10^{-4}	1.2×10^{-3}	0.087	1.216	0.532	1.110
CDCS-primal	1.8×10^{-3}	3.3×10^{-3}	0.076	0.188	0.101	0.437
CDCS-dual	1.8×10^{-3}	3.4×10^{-3}	0.064	0.174	0.091	0.484
CDCS-hsde	1.5×10^{-3}	3.3×10^{-3}	0.048	0.140	0.064	0.430

meant to provide moderately accurate solutions, the terminal objective value returned by CDCS-hsde was always within 0.2% of the high-accuracy optimal value computed using SeDuMi. This is an acceptable difference in many practical applications.

Finally, to offer a comparison of the performance of CDCS and SCS that is insensitive both to problem scaling and to differences in the stopping conditions, Table 3.5 reports the average CPU time per iteration required to solve the sparse SDPs `maxG11`, `maxG32`, `qpG11` and `qpG51`, as well as the dense SDPs `theta1` and `theta2`. Evidently, all algorithms in CDCS are faster than SCS for the large-scale sparse SDPs (`maxG11`, `maxG32`, `qpG11` and `qpG51`), and in particular CDCS-hsde improves on SCS by approximately 1.8×, 8.7×, 8.3×, and 2.6× for each problem, respectively. This is to be expected since the conic projection step in CDCS is more efficient due to smaller semidefinite cones, but the results are remarkable considering that CDCS is written in MATLAB, while SCS is implemented in C. Additionally, the performance of CDCS could be improved even further with a parallel implementation of the projections onto small PSD cones.



(a) rs35 (b) rs200 (c) rs228

(d) rs365 (e) rs1555 (f) rs1907

Figure 3.2: Aggregate sparsity patterns of the nonchordal SDPs in [25]; see Table 3.6 for the matrix dimensions.

Table 3.6: Summary of chordal decomposition for the chordal extensions of the nonchordal SDPs form [25].

	rs35	rs200	rs228	rs365	rs1555	rs1907
Original cone size, n	2003	3025	1919	4704	7479	5357
Affine constraints, m	200	200	200	200	200	200
Number of cliques, p	588	1635	783	1244	6912	611
Maximum clique size	418	102	92	322	187	285
Minimum clique size	5	4	3	6	2	7

3.6.3 Nonchordal SDPs

In our second experiment, we solved six large-scale SDPs with nonchordal sparsity patterns form [25]: rs35, rs200, rs228, rs365, rs1555, and rs1907. The aggregate sparsity patterns of these problems, illustrated in Figure 3.2, come from the University of Florida Sparse Matrix Collection [83]. Table 3.6 demonstrates that all six sparsity patterns admit chordal extensions with maximum cliques that are much smaller than the original cone.

Total CPU time, number of iterations, and terminal objective values are presented in Table 3.7. For all problems, the algorithms in CDCS (primal, dual and hsde) are all much faster than either SCS or SeDuMi. In addition, SCS never terminates succesfully, while the objective value returned by CDCS is always within 2% of the high-accuracy solutions returned by SeDuMi (when this could be computed).

The advantages of the algorithms proposed in this work are evident from Table 3.8: the average CPU time per iteration in CDCS-hsde is approximately 22×, 24×, 28×, and 105× faster compared to SCS for problems rs200, rs365, rs1907, and rs1555, respectively. The results for average CPU time per iteration also demonstrate that the computational complexity of all three algorithms in CDCS (primal, dual, and hsde) is independent of the original problem size: problems rs35 and rs228 have similar cone size n and the same

Table 3.7: Results for large-scale SDPs with nonchordal sparsity patterns form [25]. Entries marked *** indicate that the problem could not be solved due to memory limitations.

	rs35			rs200		
	Time (s)	# Iter.	Objective	Time (s)	# Iter.	Objective
SeDuMi (high)	1 391	17	25.33	4 451	17	99.74
SeDuMi (low)	986	11	25.34	2 223	8	99.73
SCS (direct)	2 378	2 000	25.08	9 697	2 000	81.87
CDCS-primal	370	379	25.27	159	577	99.61
CDCS-dual	272	245	25.53	103	353	99.72
CDCS-hsde	2 019	2 000	25.47	254	1 114	99.70
	rs228			rs365		
	Time (s)	# Iter.	Objective	Time (s)	# Iter.	Objective
SeDuMi (high)	1 655	21	64.71	***	***	***
SeDuMi (low)	809	10	64.80	***	***	***
SCS (direct)	2 338	2 000	62.06	34 497	2 000	44.02
CDCS-primal	94	400	64.65	321	401	63.37
CDCS-dual	84	341	64.76	240	265	63.69
CDCS-hsde	79	361	64.87	332	442	63.64
	rs1555			rs1907		
	Time (s)	# Iter.	Objective	Time (s)	# Iter.	Objective
SeDuMi (high)	***	***	***	***	***	***
SeDuMi (low)	***	***	***	***	***	***
SCS (direct)	139 314	2 000	34.20	50 047	2 000	45.89
CDCS-primal	1 721	2 000	61.22	330	349	62.87
CDCS-dual	317	317	69.54	271	252	63.30
CDCS-hsde	1 413	2 000	61.36	393	414	63.14

Table 3.8: Average CPU time per iteration (in seconds) for the nonchordal SDPs form [25].

	rs35	rs200	rs228	rs365	rs1555	rs1907
SCS (direct)	1.188	4.847	1.169	17.250	69.590	25.240
CDCS-primal	0.944	0.258	0.224	0.715	0.828	0.833
CDCS-dual	1.064	0.263	0.232	0.774	0.791	0.920
CDCS-hsde	1.005	0.222	0.212	0.735	0.675	0.893

number of constraints m, yet the average CPU time for the latter is approximately $5\times$ smaller. This can be explained by noticing that for all test problems considered here the number of constraints m is moderate, so the overall complexity of our algorithms is dominated by the conic projection. As stated in Proposition 3.8, this depends only on the size and number of the maximal cliques, not on the size of the original PSD cone. A more detailed investigation of how the number of maximal cliques, their size, and the number of constraints affect the performance of CDCS is presented next.

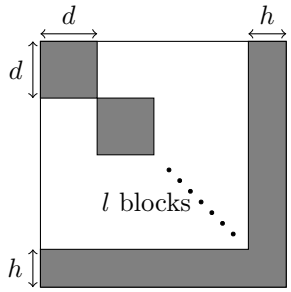


Figure 3.3: Block-arrow sparsity pattern (dots indicate repeating diagonal blocks). The parameters are: the number of blocks, l; block size, d; the width of the arrow head, h.

3.6.4 Random SDPs with block-arrow patterns

To examine the influence of the number of maximal cliques, their size, and the number of constraints on the computational cost of Algorithms 1–3, we considered randomly generated SDPs with a "block-arrow" aggregate sparsity pattern, illustrated in Figure 3.3. Such a sparsity pattern is characterized by: the number of blocks, l; the block size, d; and the size of the arrow head, h. The associated PSD cone has dimension $ld + h$. The block-arrow sparsity pattern is chordal, with l maximal cliques all of the same size $d + h$. The effect of the number of constraints in the SDP, m, is investigated as well, and numerical results are presented below for the following scenarios:

1. Fix $l = 100$, $d = 10$, $h = 20$, and vary the number of constraints, m;
2. Fix $m = 200$, $d = 10$, $h = 20$, and vary l (hence, the number of maximal cliques);
3. Fix $m = 200$, $l = 50$, $h = 10$, and vary d (hence, the size of the maximal cliques).

In our computations, the problem data are generated randomly using the following procedure. First, we generate random symmetric matrices $A_1, \ldots, A_m$ with block-arrow sparsity pattern, whose nonzero entries are drawn from the uniform distribution $U(0, 1)$ on the open interval $(0, 1)$. Second, a strictly primal feasible matrix $X_\mathrm{f} \in \mathbb{S}^n_+(\mathcal{E}, 0)$ is constructed as $X_\mathrm{f} = W + \alpha I$, where $W \in \mathbb{S}^n(\mathcal{E}, 0)$ is randomly generated with entries from $U(0, 1)$ and α is chosen to guarantee $X_\mathrm{f} \succ 0$. The vector b in the primal equality constraints is then computed such that $b_i = \langle A_i, X_\mathrm{f} \rangle$ for all $i = 1, \ldots, m$. Finally, the matrix C in the dual constraint is constructed as $C = Z_\mathrm{f} + \sum_{i=1}^m y_i A_i$, where $y_1, \ldots, y_m$ are drawn from $U(0, 1)$ and $Z_\mathrm{f} \succ 0$ is generated similarly to X_f.

The average CPU time per 100 iterations for the first-order solvers is plotted in Figure 3.4. As already observed in the previous sections, in all three test scenarios the algorithms in CDCS are faster than SCS, when the latter is used to solve the original SDPs (before chordal decomposition). Of course, as one would expect, the computational cost grows when either the number of constraints, the size of the maximal cliques, or their

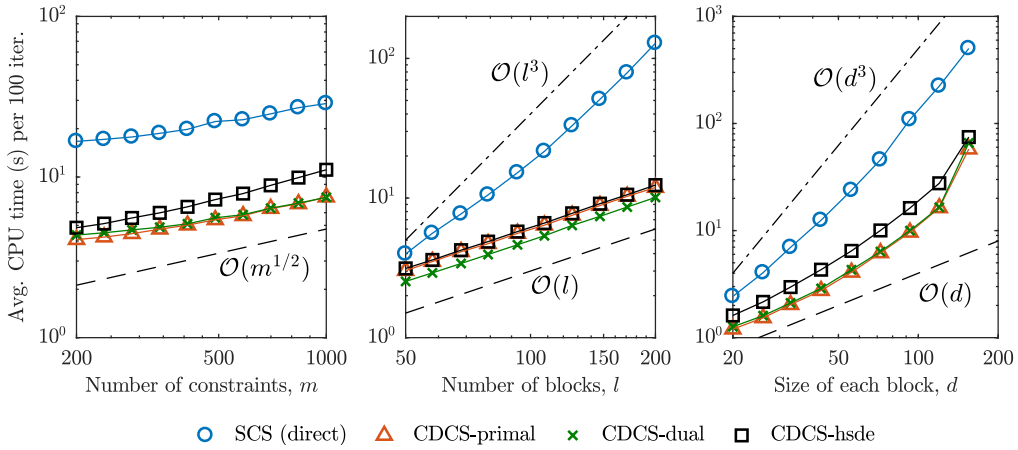


Figure 3.4: Average CPU time (in seconds) per 100 iterations for SDPs with block-arrow patterns. Left to right: varying the number of constraints; varying the number of blocks; varying the block size.

Table 3.9: Average CPU time ($\times 10^{-2}$ s) required by the affine projection steps in CDCS-primal, CDCS-dual, and CDCS-hsde as a function of the number of constraints (m) for $l = 100$, $d = 10$, and $h = 20$.

m	200	239	286	342	409	489	585	699	836	1000
CDCS-primal	1.05	1.21	1.40	1.63	1.90	2.22	2.60	3.12	3.59	4.29
CDCS-dual	1.10	1.26	1.46	1.67	1.94	2.28	2.65	3.16	3.66	4.31
CDCS-hsde	1.84	2.14	2.55	2.95	3.50	4.12	4.85	5.80	6.81	8.04

number is increased. Note, however, that the CPU time per iteration of CDCS grows more slowly than that of SCS as a function of the number of maximal cliques, which is the benefit of considering smaller PSD cones in CDCS. Precisely, the CPU time per iteration of CDCS increases linearly when the number of cliques l is raised, as expected from Proposition 3.8; instead, the CPU time per iteration of SCS grows cubically, since the eigenvalue decomposition on the original cone requires $\mathcal{O}(l^3)$ flops (note that when d and h are fixed, $(ld+h)^3 = \mathcal{O}(l^3)$). Finally, the results in Table 3.9 confirm the analysis in Propositions 3.6 and 3.7, according to which the CPU time required in the affine projection of CDCS-hsde was approximately twice larger than that of CDCS-primal or CDCS-dual. On the other hand, the increase in computational cost with the number of constraints m is slower than predicted by Propositions 3.6 and 3.7 due to the fact that, contrary to the complexity analysis presented in Section 3.5, our implementation of Algorithms 1–3 takes advantage of sparse matrix operations where possible.

3.6.5 Comparison with SparseCoLO

As our final experiment, we solved four large sparse SDPs in SDPLIB (`maxG11`, `maxG32`, `qpG11`, and `qpG51`) using SparseCoLO and SeDuMi. SparseCoLO implements the conversion techniques [23, 24] to exploit chordal decomposition properties. As listed in Table 3.10,

Table 3.10: Results for four large sparse SDPs in SDPLIB using SparseCoLO+SeDuMi. Entries marked $***$ indicate that the problem could not be solved due to memory limitations.

m	maxG11	maxG32	qpG11	qpG51
Time (s)	9.83	577.4	27.3	***
Iter.	15	15	15	***
Objective	629.2	1568	2449	***

the conversion techniques in SparseCoLO can give speedups in some cases (`maxG11` and `qpG11`) when the additional number of equality constraints is moderate. However, the failure to solve the problem `qpG51` — due to memory overflow caused by the large number of consensus constraints in the converted problem — highlights the drawbacks.

3.7 Conclusion

In this chapter, we have presented a conversion framework for large-scale SDPs characterized by chordal sparsity. This framework is analogous to the conversion techniques for IPMs of [23, 24], but is more suitable for the application of FOMs. We have then developed efficient ADMM algorithms for sparse SDPs in either primal or dual standard form, and for their homogeneous self-dual embedding. In all cases, a single iteration of our ADMM algorithms only requires parallel projections onto small PSD cones and a projection onto an affine subspace, both of which can be carried out efficiently. In particular, when the number of constraints m is moderate the complexity of each iteration is determined by the size of the largest maximal clique, not the size of the original problem. This enables us to solve large, sparse conic problems that are beyond the reach of standard interior-point and/or other first-order methods.

All our algorithms have been made available in the open-source MATLAB solver CDCS. Numerical simulations on benchmark problems, including selected sparse problems from SDPLIB, large and sparse SDPs with a nonchordal sparsity pattern, and SDPs with a block-arrow sparsity pattern, demonstrate that our methods can significantly reduce the total CPU time requirement compared to the state-of-the-art interior-point solver SeDuMi [71] and the efficient first-order solver SCS [65].

3.8 Proofs of Chapter 3

3.8.1 Proof of Proposition 3.6

Since (3.18) and (3.27) are the same modulo scaling, we only consider the former. Also, we drop the superscript (n) to lighten the notation. Recall that $H_k^T x_k$ is an indexing

operation and requires no flops, and let

$$\hat{b} := \sum_{k=1}^{p} H_k^{\mathsf{T}} \left(x_k + \rho^{-1} \lambda_k \right) - \rho^{-1} c \in \mathbb{R}^{n^2}. \tag{3.54}$$

After a suitable block elimination and writing $AD^{-1}A^{\mathsf{T}} = LL^{\mathsf{T}}$, the solution of (3.18) is given by

$$LL^{\mathsf{T}} y = AD^{-1}\hat{b} - b, \tag{3.55a}$$

$$x = D^{-1} \left(\hat{b} - A^{\mathsf{T}} y \right). \tag{3.55b}$$

Computing x and y cost $(4m + p + 3)n^2 + 2m^2 + 2n_d$ flops, counted as the sum of:

1. $(p+1)n^2 + 2n_d$ flops to form $\hat{b}$: no flops to multiply by H_k, $2|\mathcal{C}_k|^2$ flops to compute $x_k + \rho^{-1}\lambda_k$, n^2 flops to calculate $\rho^{-1}c$, and $(p-1)n^2 + n^2$ flops to sum all addends in (3.54).
2. $(2m+1)n^2$ flops to compute $AD^{-1}\hat{b} - b$: n^2 flops to compute $D^{-1}\hat{b}$ since D is diagonal, $(2n^2 - 1)m$ flops to multiply by A, and m flops to subtract b.
3. $2m^2$ flops to compute y via forward and backward substitutions using (3.55a).
4. $(2m+1)n^2$ flops to compute x via (3.55b): $(2m-1)n^2$ flops to find $A^{\mathsf{T}}y$, n^2 flops to subtract it from $\hat{b}$, and n^2 flops to multiply by D^{-1}.

3.8.2 Proof of Proposition 3.7

Consider the "inner" system (3.44) first. Partition the vectors σ_1 and σ_2 as

$$\sigma_1 = \begin{bmatrix} \sigma_{11} \\ \sigma_{12} \end{bmatrix}, \qquad \sigma_2 = \begin{bmatrix} \sigma_{21} \\ \sigma_{22} \end{bmatrix},$$

where $\sigma_{11} \in \mathbb{R}^{n^2}$, $\sigma_{12}, \sigma_{22} \in \mathbb{R}^{n_d}$, and $\sigma_{21} \in \mathbb{R}^m$. The vectors ν_1 and ν_2 on the right-hand side of (3.44) can be partitioned in a similar way. Recalling the definition of the matrix $\hat{A}$ from (3.40), (3.45b) becomes

$$\begin{bmatrix} \sigma_{21} \\ \sigma_{22} \end{bmatrix} = \begin{bmatrix} \nu_{21} - A\sigma_{11} \\ \nu_{22} - H\sigma_{11} + \sigma_{12} \end{bmatrix}. \tag{3.56}$$

To calculate σ_{11} and σ_{12} one needs to solve (3.45a), which after partitioning all variables can be rewritten as

$$\begin{bmatrix} \left(I + D + A^{\mathsf{T}}A \right) & -H^{\mathsf{T}} \\ -H & 2I \end{bmatrix} \begin{bmatrix} \sigma_{11} \\ \sigma_{12} \end{bmatrix} = \begin{bmatrix} \nu_{11} + A^{\mathsf{T}}\nu_{21} + H^{\mathsf{T}}\nu_{22} \\ \nu_{12} - \nu_{22} \end{bmatrix}. \tag{3.57}$$

Eliminating σ_{12} from the first block equation results in

$$\left(I + \frac{1}{2}D + A^{\mathsf{T}}A \right) \sigma_{11} = \nu_{11} + A^{\mathsf{T}}\nu_{21} + \frac{1}{2}H^{\mathsf{T}} \left(\nu_{12} + \nu_{22} \right), \tag{3.58a}$$

$$\sigma_{12} = \frac{1}{2}(\nu_{12} - \nu_{22} + H\sigma_{11}). \tag{3.58b}$$

After defining $P := I + \frac{1}{2}D$ and $\eta := \nu_{11} + A^\mathsf{T}\nu_{21} + \frac{1}{2}H^\mathsf{T}(\nu_{12} + \nu_{22})$ to lighten the notation, an application of the matrix inversion lemma to (3.58a) yields

$$\sigma_{11} = P^{-1}\eta - P^{-1}A^\mathsf{T}(I + AP^{-1}A^\mathsf{T})^{-1}AP^{-1}\eta. \tag{3.59}$$

We are now in a position to count the flops required to solve the "inner" linear system. First, computing σ_{11} via (3.59) requires a total $(6m+p+3)n^2+2m^2-m$ flops, counted as follows:

1. $(2m + p + 1)n^2$ flops to form η;
2. n^2 flops to compute $P^{-1}\eta$, since P is an $n^2 \times n^2$ diagonal matrix;
3. $(2n^2 - 1)m$ flops to calculate $AP^{-1}\eta$;
4. $2m^2$ flops to form the vector $(I + AP^{-1}A^\mathsf{T})^{-1}AP^{-1}\eta$ using forward and backward substitutions (we assume that the Cholesky decomposition $I + AP^{-1}A^\mathsf{T} = LL^\mathsf{T}$ has been cached);
5. $(2m - 1)n^2$ flops to find $A^\mathsf{T}(I + AP^{-1}A^\mathsf{T})^{-1}AP^{-1}\eta$;
6. $2n^2$ flops to compute σ_{11} via (3.59) given $P^{-1}\eta$ and $A^\mathsf{T}(I + AP^{-1}A^\mathsf{T})^{-1}AP^{-1}\eta$.

Once σ_{11} is known, σ_{12} is found from (3.58b) with $3n_d$ flops because the product $H\sigma_{11}$ is simply an indexing operation and costs no flops. Given σ_{11} and σ_{12}, computing σ_{21} and σ_{22} from (3.56) requires $2mn^2 + 2n_d$ flops, so the "inner" linear system (3.44) costs a total of $(8m + 2p + 3)n^2 + 2m^2 - m + 5n_d$ flops.

After the inner system has been solved, we see that computing $\hat{u}_1$ from (3.43) requires $(8m + 2p + 9)n^2 + 2m^2 + 5m + 17n_d - 1$ flops in total:

1. $2(n^2 + 2n_d + m)$ flops to compute $\omega_1 - \omega_2\zeta$;
2. $(8m+2p+3)n^2+2m^2-m+5n_d$ flops to solve the "inner" linear system $M^{-1}(\omega_1-\omega_2\zeta)$;
3. $2(n^2 + 2n_d + m) - 1$ flops to compute $\zeta^\mathsf{T}M^{-1}(\omega_1 - \omega_2\zeta) \in \mathbb{R}$;
4. $n^2 + 2n_d + m$ flops to calculate $\hat{\zeta} \cdot \zeta^\mathsf{T}M^{-1}(\omega_1 - \omega_2\zeta)$;
5. $n^2 + 2n_d + m$ flops to compute $\hat{u}_1 = M^{-1}(\omega_1 - \omega_2\zeta) - \hat{\zeta} \cdot \zeta^\mathsf{T}M^{-1}(\omega_1 - \omega_2\zeta)$.

Summing this to the $2(n^2 + 2n_d + m)$ flops required to calculate $\hat{u}_2$ using (3.42b) yields the desired result.

3.8.3 Proof of Proposition 3.8

The conic projection (3.21) in Algorithm 1 amounts to projecting the matrices

$$\operatorname{mat}\left(H_k x^{(n+1)} - \rho^{-1}\lambda_k^{(n)}\right) \in \mathbb{S}^{|\mathcal{C}_k|}, \quad k = 1, \ldots, p$$

onto the PSD cone $\mathbb{S}_+^{|\mathcal{C}_k|}$. Computing $H_k x^{(n+1)} - \rho^{-1}\lambda_k^{(n)}$ requires $2|\mathcal{C}_k|^2$ flops, while a PSD projection using a full eigenvalue decomposition costs $\mathcal{O}(|\mathcal{C}_k|^3)$ flops to leading order, so the overall number of flops is $\mathcal{O}(\sum_{k=1}^p |\mathcal{C}_k|^3)$. The same argument holds for the conic projection (3.29) in Algorithm 2.

In Algorithm 3, instead, the projection is onto the cone $\mathcal{K} := \mathbb{R}^{n^2} \times \mathcal{S} \times \mathbb{R}^m \times \mathbb{R}^{n_d} \times \mathbb{R}_+$. Nothing needs to be done to project onto $\mathbb{R}^{n^2}$, $\mathbb{R}^m$ and $\mathbb{R}^{n_d}$, while the projection of $a \in \mathbb{R}$ onto $\mathbb{R}_+$ is given by maximize$\{0, a\}$ and requires no flops according to our definition. Finally, projecting onto $\mathcal{S}$ requires eigenvalue decompositions of the matrices $\operatorname{mat}(x_k)$, $k = 1, \ldots, p$, with a leading-order cost of $\mathcal{O}(\sum_{k=1}^p |\mathcal{C}_k|^3)$ flops.

4

Scalable systems analysis using CDCS

This chapter demonstrates the performance of CDCS for scalable analysis of linear networked systems, including stability, $\mathcal{H}_2$ and $\mathcal{H}_\infty$ performance. Our main strategy is to exploit any sparsity within these analysis problems and use chordal decomposition. By choosing block-diagonal Lyapunov functions, we decompose large positive semidefinite (PSD) constraints in all of the analysis problems into multiple smaller ones depending on the maximal cliques of the system graph. This makes the solutions more computationally efficient via CDCS.

4.1 Introduction

Large-scale networked systems, consisting of multiple subsystems over a network, have received considerable attention [1]. One of the challenges arising in these systems is to develop scalable methods that are able to solve the associated analysis and synthesis problems efficiently. However, classical methods often suffer from a lack of scalability for large systems, since their computational demand usually grows rapidly as the system's dimension increases.

In the literature, there are two groups of scalable analysis techniques for large-scale networked systems: 1) *compositional analysis* [84–87]; and 2) *positive system theory* [49, 88–91]. The former method is usually carried out in the framework of dissipative systems, while the latter method aims to solve a special type of dynamical systems. The main strategy of compositional analysis is to find individual supply rate for each dissipative subsystem and then to establish a global storage function as a combination of the local storage functions [84, 85]. Recently, Meissen *et al.* employed a first-order method to optimize the local supply rates for certifying stability of an interconnected system [86], which might reduce the conservatism brought by individual storage functions. Anderson and Papachristodoulou proposed a decomposition technique based on graph partition that facilitates the compositional analysis [87]. Another group of scalable strategies focuses on a

particular class of systems, *i.e.*, positive systems [88], where the system matrices only have nonnegative off-diagonal entries. It is well-known that stability and performance of positive systems can be verified using linear Lyapunov functions [89], which can be computed by more scalable linear programs (LPs) instead of traditional semidefinite programs (SDPs). Tanaka and Langbort showed that it is necessary and sufficient to use a diagonal Lyapunov function in the KYP lemma for positive systems [90]. Sootla and Rantzer proposed scalable model reduction techniques for positive systems using linear energy functions [91].

In contrast to the compositional analysis and positive system theory, our approach focuses on the inherent structure and sparsity of networked systems and uses sparse optimization techniques, particularly chordal decomposition, to solve the analysis problems efficiently. This idea is in line with some of the early results in the field [13, 34, 41, 56]. Chordal decomposition is a celebrated result in linear algebra that connects sparse positive semidefinite matrices and chordal graphs. As already discussed in the previous chapters, there is a broad literature regarding the applications of chordal graph properties in combinatorial problems, Cholesky factorization, matrix completion and sparse semidefinite optimization.

In this chapter, we introduce a chordal decomposition approach based on CDCS for scalable analysis of linear networked systems. We focus on the well-known convex formulations of the analysis problems, *i.e.*, stability, $\mathcal{H}_2$ and $\mathcal{H}_\infty$ performance, and show how to decompose large PSD constraints in all of the analysis problems into multiple smaller ones, thus facilitating their solutions. By choosing block-diagonal Lyapunov functions, we show that the graph structure of the networked system can be potentially inherited in the resulting SDPs of the analysis problems. This allows us to decompose a large PSD constraint into multiple smaller ones. Consequently, for sparse networked systems, the decomposed analysis problems can be solved efficiently using CDCS.

The rest of this chapter is organized as follows. In Section 4.2, we present the problem statement. Chordal decomposition in sparse SDPs is reviewed in Section 4.3. Section 4.4 presents the scalable analysis approach for stability $\mathcal{H}_2$ and $\mathcal{H}_\infty$ performance. Numerical results are shown in Section 4.5, and we conclude the chapter in Section 4.6.

4.2 Problem statement

We consider a network of linear heterogeneous subsystems interacting over a directed graph $\mathcal{G}(\mathcal{V},\mathcal{E})$. Each node in $\mathcal{V} = \{1,\dots,N\}$ represents a subsystem, and the edge $(i,j) \in \mathcal{E}$ means that subsystem i exerts dynamical influence on subsystem j. The dynamics of subsystem $i \in \mathcal{V}$ are written as

$$\begin{aligned} \dot{x}_i(t) &= A_{ii}x_i(t) + \sum_{j\in\mathbb{N}_i} A_{ij}x_j(t) + B_i w_i(t), \\ y_i(t) &= C_i x_i(t) + D_i w_i(t), \end{aligned} \tag{4.1}$$

where $x_i \in \mathbb{R}^{\alpha_i}, y_i \in \mathbb{R}^{d_i}, w_i \in \mathbb{R}^{m_i}$ represent the local state, output and disturbance, respectively, and $\mathbb{N}_i$ denotes the neighbours of node i. By collecting the subsystems' states, the overall state-space model is then written concisely as

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bw(t), \\ y(t) &= Cx(t) + Dw(t), \end{aligned} \tag{4.2}$$

where $x = [x_1^\mathsf{T}, x_2^\mathsf{T}, \ldots, x_N^\mathsf{T}]^\mathsf{T}$, and the vectors y, w are defined similarly. The matrix A is composed of blocks A_{ij}, which exhibits a block sparsity pattern $A \in \mathbb{R}_\alpha^{N\times N}(\mathcal{E}, 0)$, where $\alpha = \{\alpha_1, \ldots, \alpha_N\}$ denotes a block partition. The matrices B, C, D have block-diagonal structures with block dimensions matching the diagonal blocks of A.

In this chapter, we consider three analysis problems of the linear networked system (4.2):

1. Verify the asymptotical stability when $w = 0$;
2. Calculate the $\mathcal{H}_2$ performance when $D = 0$;
3. Calculate the $\mathcal{H}_\infty$ performance.

It is well-known that these three problems can be equivalently reformulated as certain optimization problems [10]:

1) Stability: System (4.2) with $w = 0$ is asymptotically stable if and only if the Lyapunov linear matrix inequality (LMI) is feasible

$$\begin{aligned} \text{find} \quad & P \succ 0, \\ \text{subject to} \quad & A^\mathsf{T}P + PA \prec 0. \end{aligned} \tag{4.3}$$

2) $\mathcal{H}_2$ performance: The $\mathcal{H}_2$ performance of system (4.2) with $D = 0$ can be computed as

$$\begin{aligned} \underset{P}{\text{minimize}} \quad & \text{Trace}(B^\mathsf{T}PB) \\ \text{subject to} \quad & A^\mathsf{T}P + PA + C^\mathsf{T}C \preceq 0, \\ & P \succ 0. \end{aligned} \tag{4.4}$$

where $\|C(sI - A)^{-1}B\|_{\mathcal{H}_2} = \sqrt{\text{Trace}(B^\mathsf{T}PB)}$.

3) $\mathcal{H}_\infty$ performance: The $\mathcal{H}_\infty$ performance of system (4.2) can be computed as

$$\begin{aligned} \underset{P}{\text{minimize}} \quad & \gamma \\ \text{subject to} \quad & \begin{bmatrix} A^\mathsf{T}P + PA & PB & C^\mathsf{T} \\ B^\mathsf{T}P & -\gamma I & D^\mathsf{T} \\ C & D & -\gamma I \end{bmatrix} \prec 0, \\ & P \succ 0. \end{aligned} \tag{4.5}$$

where $\|C(sI - A)^{-1}B + D\|_{\mathcal{H}_\infty} = \gamma$.

Problems (4.3)-(4.5) are convex, and ready to solve via existing interior-point solvers, such as SeDuMi [71]. The main difficulty is that standard interior-point solvers do not scale well for large problem instances. One major reason is that the constraints in (4.3)-(4.5) are imposed on the global system and consequently the computational complexity grows very quickly as the number of subsystems increases. Typically, the system graph $\mathcal{G}$ is sparse for practical large-scale systems, meaning that each subsystem only has physical connections with a few other subsystems. In this chapter, we aim to exploit this sparsity via CDCS to solve (4.3)-(4.5) efficiently.

Remark 4.1. Note that there are also other efficient formulations to test stability and to compute $\mathcal{H}_2$ and $\mathcal{H}_\infty$ performance [10]. One additional benefit of problems (4.3)-(4.5) is that we can get a proper Lyapunov function, defined by $V(x) = x^T P x$. Also, problems (4.3)-(4.5) are helpful for some standard static synthesis problems via a standard change of variables. In this chapter, we will focus on (4.3)-(4.5), and use CDCS to solve them efficiently when the system graph $\mathcal{G}$ is sparse.

4.3 Chordal decomposition in sparse SDPs

In this section, we focus on the SDP formulations of the optimization problems (4.3)-(4.5), and briefly review the idea of Chapter 3 to decompose them using chordal graph theory. For convenience, we recall that the standard *primal form* of an SDP is

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & \langle A_0, X\rangle \\ \text{subject to} \quad & \langle A_i, X\rangle = b_i, i = 1, \ldots, m, \\ & X \succeq 0, \end{aligned} \tag{4.6}$$

and its standard *dual form* is

$$\begin{aligned} \underset{y,Z}{\text{maximize}} \quad & \langle b, y\rangle \\ \text{subject to} \quad & Z + \sum_{i=1}^{m} y_i A_i = A_0, \\ & Z \succeq 0, \end{aligned} \tag{4.7}$$

where X is the primal variable, y, Z are the dual variables, and $b \in \mathbb{R}^m, A_i \in \mathbb{S}^N, i = 0, 1, \ldots, m$ are problem data.

Suppose the data matrices in (4.6) and (4.7) have an aggregate sparsity pattern: $A_0, A_1, \ldots, A_m \in \mathbb{S}^N_\alpha(\mathcal{E}, 0)$. It is assumed that the pattern $\mathcal{E}$ is chordal with a set of maximal cliques $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_p$. Note that the cost function and equality constraints in (4.6) only depend on the entries X_{ij} on the diagonal and $(i, j) \in \mathcal{E}$. The remaining elements simply guarantee that the matrix is PSD. Also, in (4.7) any feasible solution

Z satisfies the sparsity pattern $\mathbb{S}^N_\alpha(\mathcal{E}, 0)$. Recalling the definition of $\mathbb{S}^N_{\alpha,+}(\mathcal{E}, ?)$ and according to Theorems 2.17 and 2.18, we can rewrite the primal SDP (4.6) and dual SDP (4.7), respectively, as

$$\begin{array}{ll} \underset{X, X_k}{\text{minimize}} & \langle A_0, X\rangle \\ \text{subject to} & \langle A_i, X\rangle = b_i, \qquad\qquad i = 1, \ldots, m, \\ & X_k = E_{\mathcal{C}_k,\alpha} X E_{\mathcal{C}_k,\alpha}^{\mathsf{T}}, \quad k = 1, \ldots, p, \\ & X_k \in \mathbb{S}_+^{|\mathcal{C}_k|_\alpha}, \qquad\quad k = 1, \ldots, p, \end{array} \tag{4.8}$$

and

$$\begin{array}{ll} \underset{y, Z_k, V_k}{\text{maximize}} & \langle b, y\rangle \\ \text{subject to} & \displaystyle\sum_{k=1}^{p} E_{\mathcal{C}_k,\alpha}^{\mathsf{T}} V_k E_{\mathcal{C}_k,\alpha} + \sum_{i=1}^{m} y_i A_i = A_0, \\ & Z_k = V_k, \quad k = 1, \ldots, p, \\ & Z_k \in \mathbb{S}_+^{|\mathcal{C}_k|_\alpha}, k = 1, \ldots, p. \end{array} \tag{4.9}$$

In (4.8) and (4.9), the original single large PSD cone has been replaced by multiple smaller PSD cones, coupled by a set of consensus variables. Then, first-order methods can be applied to the decomposed formulations (4.8) and (4.9), or their homogeneous self-dual embedding, which leads to algorithms only involving parallel PSD projections onto p smaller cones and a projection onto an affine set at each iteration. If the size of the largest maximal clique is small, then the reduction of cone dimensions enables us to compute PSD projections much more efficiently. Consequently, the application of first-order methods in the decomposed problems (4.8) and (4.9) improves the scalability to solve sparse SDPs when seeking a solution of moderate accuracy. The interested reader is referred to Chapter 3 or [36, 38] for details. The MATLAB package CDCS [35] provides an efficient implementation of the above decomposition method.

4.4 Scalable performance analysis of sparse systems

This section applies the chordal decomposition techniques in stability, $\mathcal{H}_2$ and $\mathcal{H}_\infty$ analyses of linear networked systems. Our strategy is to restrict the sparsity pattern of P, such that the sparsity structure of the dynamical system is preserved in the SDP formulations of (4.3)-(4.5). This allows one to decompose a single large PSD constraint into multiple smaller ones using chordal decomposition, thus facilitating their solutions using CDCS. We note that the proposed method of this section may introduce certain conservatism for general networked systems since we use a structured Lyapunov function.

4.4.1 Stability verification

We first show that the sparsity pattern of $A^\mathsf{T}P + PA$ reflects the aggregate sparsity pattern of the resulting SDP formulation. Let us write the Lyapunov LMI as

$$\begin{bmatrix} -P & 0 \\ 0 & A^\mathsf{T}P + PA \end{bmatrix} \prec 0. \tag{4.10}$$

There are up to $m = \frac{N(N+1)}{2}$ free variables in matrix P. We denote $W_1, W_2, \ldots, W_m$ as the standard basis matrices for $\mathbb{S}^N$, and define the matrices $A_1, A_2, \ldots, A_m \in \mathbb{S}^{2N}$ as

$$A_i = \begin{bmatrix} -W_i & 0 \\ 0 & A^\mathsf{T}W_i + W_iA \end{bmatrix}, i = 1, 2, \ldots, m. \tag{4.11}$$

Then, (4.10) can be reformulated into a standard dual SDP

$$\begin{aligned} \underset{y,Z}{\text{maximize}} \quad & \langle b, y \rangle \\ \text{subject to} \quad & Z + \sum_{i=1}^{m} y_i A_i = A_0, \\ & Z \succeq 0 \end{aligned} \tag{4.12}$$

where $y \in \mathbb{R}^m, Z \in \mathbb{S}_+^{2N}$, $A_0 = -\epsilon I, \epsilon > 0, b = 0$, and A_i is defined in (4.11). At this point, we know that the aggregate sparsity pattern of (4.12) is

$$\mathcal{P}(A_0) \cup \mathcal{P}(A_1) \cup \cdots \cup \mathcal{P}(A_m) = \mathcal{P}\left(\begin{bmatrix} -P & 0 \\ 0 & A^\mathsf{T}P + PA \end{bmatrix}\right),$$

where $\mathcal{P}(\cdot)$ denotes the sparsity pattern of a matrix. The aggregate sparsity pattern of (4.12) directly depends on the sparsity patterns of P and $A^\mathsf{T}P + PA$.

For a networked system (4.2), system matrix A has an inherent structure described by $\mathcal{G}(\mathcal{V}, \mathcal{E})$, *i.e.*, $A \in \mathbb{R}_\alpha^{N\times N}(\mathcal{E}, 0)$, where $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ denotes the dimensions of local states. Apparently, a dense P has no conservatism in certifying stability, but leads to a full pattern of $A^\mathsf{T}P + PA$. To preserve the physical sparsity structure $\mathcal{G}$, we consider the following problem: Given a sparse $A \in \mathbb{R}_\alpha^{N\times N}(\mathcal{E}, 0)$, find a sparsity pattern of P, such that the pattern of $A^\mathsf{T}P + PA$ is still sparse.

We note that a complete answer to this question is difficult for general systems, especially considering the relationship between sparsity (*i.e.*, efficiency) and conservativeness. One trivial choice is a block-diagonal P with block sizes compatible with the subsystem sizes α_i. To handle the symmetry in $A^\mathsf{T}P + PA$, we introduce mirror graphs as follows.

Definition 4.2. (*Mirror Graph*) Let $\mathcal{G}(\mathcal{V}, \mathcal{E})$ be a directed graph. We define $\mathcal{E}_r$ 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 $\mathcal{G}$ is a directed graph in the form $\mathcal{G}_r(\mathcal{V}, \mathcal{E}_r)$ with the same set of nodes $\mathcal{V}$ and the set of reverse edges $\mathcal{E}_r$.

Then, the graph structure in the dynamical system (4.1) is naturally inherited in (4.12), *i.e.*,

$$A^{\mathsf{T}}P + PA \in \mathbb{S}^N_{\alpha}(\mathcal{E} \cup \mathcal{E}_r, 0). \tag{4.13}$$

We note that the existence of block-diagonal P is investigated in [49], and diagonal P is necessary and sufficient to certify stability of positive systems [89]. Other choices of P are available for special graphs $\mathcal{G}$, such as trees and cycles [13].

In this chapter, we assume the pattern of $A^{\mathsf{T}}P + PA$ can be described by a chordal graph $\mathcal{G}_c(\mathcal{V}, \mathcal{E}_c)$ with a set of maximal cliques $\mathcal{C}_1, \ldots, \mathcal{C}_p$. As mentioned above, one basic choice is a block-diagonal P. If $\mathcal{E} \cup \mathcal{E}_r$ is non-chordal, then we can make a chordal extension to get $\mathcal{E}_c$. This chapter aims to highlight the computational benefits brought by subsequent chordal decomposition, and we leave the discussion of non-block diagonal P to our future work. In (4.12), the single large PSD cone $Z \succeq 0$ has two blocks, where the upper-left one corresponds to block-diagonal P and the bottom-right one can be replaced by $\mathbb{S}^N_{\alpha,+}(\mathcal{E}_c, 0)$. Then, $\mathbb{S}^N_{\alpha,+}(\mathcal{E}_c, 0)$ can be subsequently decomposed into multiple smaller cones using chordal decomposition (see the reformulations (4.8) and (4.9)). Consequently, if the largest maximal clique is small, the SDP formulation (4.12) can be expected to solve efficiently for sparse systems using CDCS.

4.4.2 $\mathcal{H}_2$ performance

Similar to the stability analysis, the $\mathcal{H}_2$ optimization problem (4.4) can be reformulated into a standard SDP of primal form (4.6) or dual form (4.7). The aggregate sparsity pattern of the resulting SDP is determined by the pattern of $A^{\mathsf{T}}P + PA + C^{\mathsf{T}}C$. Considering the structure of networked system (4.1), we have

$$\mathcal{P}(A^{\mathsf{T}}P + PA + C^{\mathsf{T}}C) = \mathcal{P}(A^{\mathsf{T}}P + PA). \tag{4.14}$$

Then, the argument for stability analysis can be applied to $\mathcal{H}_2$ analysis for the purpose of scalable computation. We assume the pattern of $A^TP + PA$ can be described by a chordal graph $\mathcal{G}_c(\mathcal{V}, \mathcal{E}_c)$, leading to

$$A^{\mathsf{T}}P + PA + C^{\mathsf{T}}C \in \mathbb{S}^N_{\alpha}(\mathcal{E}_c, 0).$$

Consequently, the sparse optimization technique [35] is ready to solve the decomposed version of (4.4). Note that we can only obtain an approximated (upper bound) $\mathcal{H}_2$ performance in general due to using a sparse P.

4.4.3 $\mathcal{H}_\infty$ performance

When reformulating the $\mathcal{H}_\infty$ analysis problem (4.5) into a standard SDP, the aggregate sparsity pattern depends on the pattern of the following matrix

$$M = \begin{bmatrix} A^\mathsf{T}P + PA & PB & C^\mathsf{T} \\ B^\mathsf{T}P & -\gamma I & D^\mathsf{T} \\ C & D & -\gamma I \end{bmatrix}. \tag{4.15}$$

According to the inherent structure in (4.1), we know $A \in \mathbb{R}_\alpha^{N\times N}(\mathcal{E}, 0)$ with $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ corresponding to the dimensions of local states, and B, C, D are block-diagonal. Consequently, the second main block γI on the diagonal has a partition $\theta = \{m_1, m_2, \ldots, m_n\}$ that corresponds to the dimensions of local disturbances, and the third main block γI has a partition $\delta = \{d_1, d_2, \ldots, d_n\}$ that corresponds to the dimensions of local outputs.

If we restrict P to be block-diagonal with compatible block sizes, then the entry PB is also block-diagonal. The sparsity pattern of the first block on the diagonal is

$$A^\mathsf{T}P + PA \in \mathbb{S}_\alpha^N(\mathcal{E}_c, 0),$$

where $\mathcal{E}_c$ is the chordal extension of $\mathcal{E}\cup\mathcal{E}_r$, defined in Section 4.4.1. Then, we have the following result.

Theorem 4.3. Consider a networked system with dynamics (4.2) and a block-diagonal P. Suppose that the sparsity pattern of $A^\mathsf{T}P + PA$ has p maximal cliques $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_p$, and the cardinality of the largest maximal clique is h. Then,

1. the block matrix M in (4.15) has a partition

$$\hat{\alpha} = \{\alpha_1, \alpha_2, \ldots, \alpha_n, m_1, m_2, \ldots, m_n, d_1, d_2, \ldots, d_n\};$$

2. the sparsity pattern of M, denoted as $M \in \mathbb{S}_{\hat{\alpha}}^{\hat{N}}(\hat{\mathcal{E}}, 0)$, has $p + n$ maximal cliques, and the cardinality of the largest maximal clique is maximize$\{h, 3\}$.

Proof. According to (4.1), we know $A \in \mathbb{R}_\alpha^{N\times N}(\mathcal{E}_c, 0)$, and B, C, D are block-diagonal. Then, it is straightforward to see that the block matrix M in (4.15) has a partition $\hat{\alpha} = \{\alpha_1, \alpha_2, \ldots, \alpha_n, m_1, m_2, \ldots, m_n, d_1, d_2, \ldots, d_n\}$, where $\alpha_i, m_i, d_i (i = 1, \ldots, n)$ are the dimensions of local states, disturbances and outputs, respectively.

Let us first consider the following block

$$M_1 = \begin{bmatrix} A^\mathsf{T}P + PA & PB \\ B^T P & -\gamma I \end{bmatrix} \in \mathbb{S}_{\hat{\alpha}_1}^{\hat{N}_1}(\hat{\mathcal{E}}_1, 0), \tag{4.16}$$

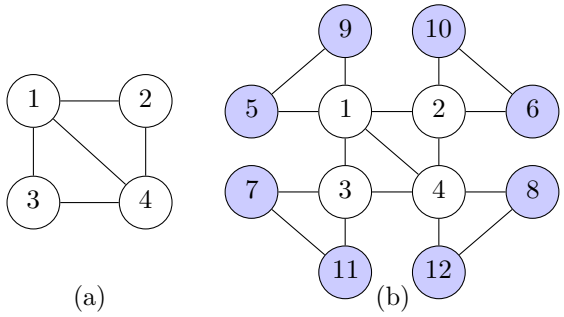


Figure 4.1: (a) Sparsity pattern of $A^\mathsf{T}P + PA$ with maximal cliques $\mathcal{C}_1 = \{1, 2, 4\}$ and $\mathcal{C}_2 = \{1, 3, 4\}$. (b) Corresponding sparsity pattern of the $\mathcal{H}_\infty$ performance matrix (4.15), where the maximal cliques are $\mathcal{C}_1 = \{1, 2, 4\}, \mathcal{C}_2 = \{1, 3, 4\}, \mathcal{C}_3 = \{1, 5, 9\}, \mathcal{C}_4 = \{2, 6, 10\}, \mathcal{C}_5 = \{3, 7, 11\}$ and $\mathcal{C}_6 = \{4, 8, 12\}$.

where the partition $\hat{\alpha}_1 = \{\alpha_1, \alpha_2, \ldots, \alpha_n, m_1, m_2, \ldots, m_n\}$ and $\hat{N}_1 = \sum_{i=1}^{n}(\alpha_i + m_i)$. Since the matrices PB and γI are block diagonal, every node $i \in \{n+1, \ldots, n+n\}$ is only connected to one node $i - n$. Then, the edge set for M_1 is shown as

$$\hat{\mathcal{E}}_1 = \mathcal{E}_c \bigcup \{(i, i+n) \mid i = 1, \ldots, n\}, \tag{4.17}$$

indicating that the maximal cliques of $\hat{\mathcal{E}}_1$ are given by

$$\mathcal{C}_1, \ldots, \mathcal{C}_p, \mathcal{C}_{p+i} = \{i, i+n\}, i = 1, \ldots, n. \tag{4.18}$$

Next, according to (4.15) and (4.16), we know

$$M = \begin{bmatrix} M_1 & H \\ H^\mathsf{T} & -\gamma I \end{bmatrix} \in \mathbb{S}_{\hat{\alpha}}^{\hat{N}}(\hat{\mathcal{E}}, 0), \tag{4.19}$$

where $H^\mathsf{T} = \begin{bmatrix} C & D \end{bmatrix}$ and $\hat{N} = \sum_{i=1}^{n}(\alpha_i + m_i + d_i)$. Since the matrices C, D and γI are block diagonal, every node $i \in \{2n+1, \ldots, 2n+n\}$ is connected to another two nodes $i - n, i - 2n$. Consequently, the edge set for M is given by

$$\hat{\mathcal{E}} = \hat{\mathcal{E}}_1 \bigcup \{(i, i+2n), (i+n, i+2n) \mid i = 1, \ldots, n\}. \tag{4.20}$$

According to the edge set $\hat{\mathcal{E}}_1$ (4.17), we know that in the edge set $\hat{\mathcal{E}}$, $\{i, i+n, i+2n\}$ forms a maximal clique. This implies that the maximal cliques of $\hat{\mathcal{E}}$ are (see Figure 4.1 for illustration)

$$\mathcal{C}_1, \ldots, \mathcal{C}_p, \mathcal{C}_{p+i} = \{i, i+n, i+2n\}, i = 1, \ldots, n. \tag{4.21}$$

Therefore, the sparsity pattern of M, $\mathbb{S}_{\hat{\alpha}}^{\hat{N}}(\hat{\mathcal{E}}, 0)$, has $p + n$ maximal cliques, and the cardinality of the largest maximal clique is maximize$\{h, 3\}$. ■

Although $\mathcal{H}_\infty$ analysis problem (4.5) appears to be more complex than the Lyapunov LMI (4.3), Theorem 4.3 shows that the underlying maximal cliques are similar and that the cardinality of the largest maximal clique for (4.5) and (4.3) is almost identical. Therefore, the strategy for stability analysis can be applied to $\mathcal{H}_\infty$ analysis problem (4.5): the single large PSD cone can be decomposed into $p+n$ smaller ones, and the sparse optimization technique [35] can be used to solve the decomposed problem in a scalable fashion. Figure 4.1 shows an instance of Theorem 4.3: the pattern of the Lyapunov LMI (4.3) in Figure 4.1 (a) with $p=2$ maximal cliques is extended to that in Figure 4.1 (b) with $p+n=6$ maximal cliques for $\mathcal{H}_\infty$ analysis.

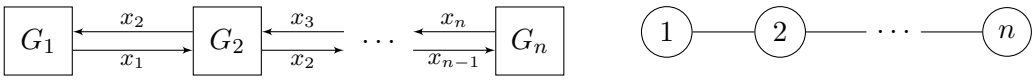
4.5 Numerical simulations

To show the efficiency of the chordal decomposition approach, we consider a chain of n subsystems, and a networked system over a scale-free graph. We solved the SDP formulations of stability analysis (4.3), $\mathcal{H}_2$ performance (4.4), and $\mathcal{H}_\infty$ performance (4.5) using standard dense solvers: SeDuMi [71] and SCS [65], as well as using the sparse conic solver CDCS [35] that exploits chordal sparsity. Block-diagonal P was used in the formulations, which were reformulated into standard SDPs using YALMIP [79]. For the interior-point solver SeDuMi, we used its default parameters, and the first-order solvers SCS and CDCS were called with termination tolerance 10^{-4} and number of iterations limited to 2000. All simulations were run on a PC with a 2.8 GHz Intel Core i7 CPU and 8GB of RAM.

4.5.1 A chain of subsystems

We first consider a chain of n subsystems, where each subsystem has physical interactions with its two neighbouring ones except the first and last subsystem, which only interacts with one neighbouring subsystem; see Figure 4.2 (a) for illustration. The simplification of this chain is shown in Figure 4.2 (b). In this case, the maximal cliques of the graph in Figure 4.2 (b) are $\{i, i+1\}, i=1,\ldots,n-1$, and the cardinality of the largest maximal clique is only 2. We can expect that the chordal decomposition strategy gains significant speed-up for solving the analysis problems in this case.

In the simulations, the state dimension n_i was chosen randomly from 5 to 10, and the dimensions of output and disturbance (d_i, m_i) were chosen randomly from 1 to 5. Then, we generated random matrices A_{ii}, A_{ij}, B_i, D_i and imposed the global state matrix A with negative eigenvalues by setting $A := A - (\lambda_{\max}+5)I$, where $\lambda_{\max}$ denotes the maximum real part of the eigenvalues of A. Figure 4.3 shows the CPU time in seconds required by the solvers for testing stability, and computing approximated $\mathcal{H}_2$ and $\mathcal{H}_\infty$



(a) (b)

Figure 4.2: A chain of n subsystems: (a) each subsystem G_i has physical interactions with its nearest two neighbouring ones, except the first one and the last one which only interacts with one nearest neighbouring subsystem; (b) a simplified line graph illustration.

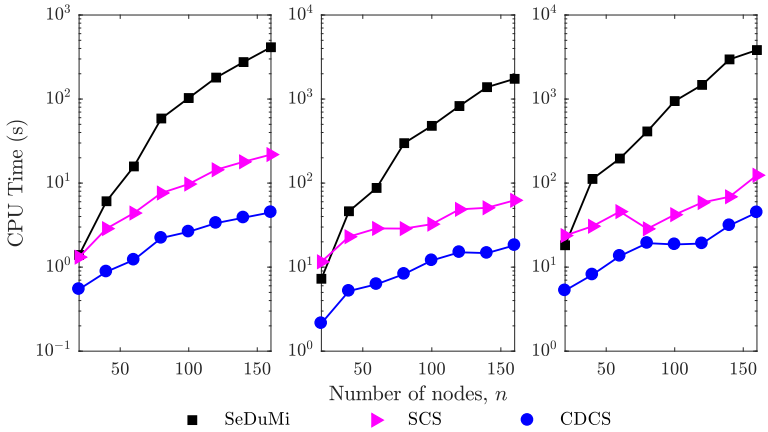


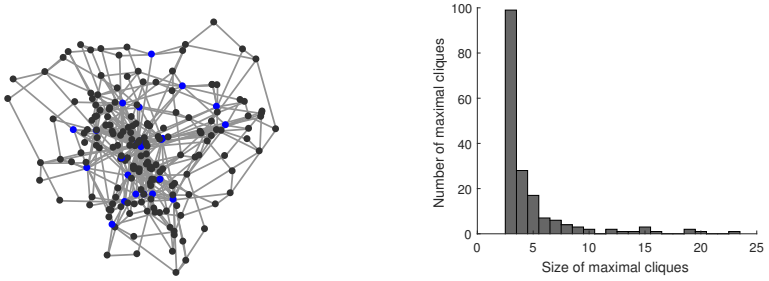
Figure 4.3: CPU time in seconds required by SeDuMi, SCS and CDCS to solve the SDP formulations of the analysis problems of a chain of subsystems. CDCS exploits the chordal decomposition in solving sparse SDPs.

performance. The chordal decomposition approach (via CDCS) took significantly less time than standard dense methods (using either SeDuMi or SCS). Moreover, the CPU time required by CDCS seems to grow linearly as the system size increases. This is expected since the size of the largest maximal clique is fixed for a line graph, indicating that the size of the PSD cones after decomposition is fixed and only the number of PSD cones increases linearly as growth of the graph size. Finally, Table 4.1 lists the $\mathcal{H}_2$ and $\mathcal{H}_\infty$ performance computed by different solvers. We can see that although first-order methods are only meant to provide solutions of moderate accuracy, the value returned CDCS was very close to the one computed by SeDuMi (within 1%). Also, as shown in Table 4.1, the $\mathcal{H}_\infty$ performance computed using a block-diagonal P is close to the one returned by MATLAB routine `norm(sys,inf)` without any assumption on P (we refer to this as the accurate result). In fact, the minor differences in the $\mathcal{H}_\infty$ case is due to numerical errors returned by different solvers. This is because the numerical examples were positive systems (the off-diagonal elements in the matrix A is nonnegative), for which it is sufficient to use diagonal Lyapunov functions to certificate the $\mathcal{H}_\infty$ norm [89, 90]. For general linear systems, using (block)-diagonal Lyapunov functions could bring certain conservatism for both $\mathcal{H}_2$ and $\mathcal{H}_\infty$ cases. A detailed investigation of the conservatism deserves further investigation.

Table 4.1: Approximated $\mathcal{H}_2$ and $\mathcal{H}_\infty$ performance of a chain of subsystems computed by different solvers.

	$\mathcal{H}_2$				$\mathcal{H}_\infty$			
n	†	SeDuMi	SCS	CDCS	‡	SeDuMi	SCS	CDCS
20	9.70	17.73	17.73	17.73	3.65	3.66	3.70	3.66
40	11.66	20.07	20.07	20.07	3.67	3.68	3.74	3.68
60	14.72	25.78	25.79	25.78	3.75	3.77	3.85	3.77
80	16.85	28.70	28.71	28.69	4.32	4.34	4.37	4.34
100	18.08	29.88	29.91	29.88	3.91	3.92	3.96	3.92
120	19.71	32.10	32.12	32.09	4.02	4.03	4.10	4.04
140	21.51	35.59	35.64	35.58	4.09	4.10	4.16	4.11
160	24.64	40.65	40.73	40.65	4.07	4.08	4.18	4.09

†: Accurate $\mathcal{H}_2$ performance returned by the MATLAB routine `norm(sys,2)`.
‡: Accurate $\mathcal{H}_\infty$ performance returned by the MATLAB routine `norm(sys,inf)`.



(a) (b)

Figure 4.4: (a) A scale free graph of 200 nodes in our experiment. The chordal extension has 178 maximal cliques and the size of the largest maximal clique is 23 (highlighted in blue); (b) Distribution of the maximal clique size in a chordal extension of the scale-free graph.

4.5.2 Networked systems over a scale-free graph

In our next experiment, we consider a network of subsystems interacting over a randomly generated scale-free graph. A scale-free graph is a graph that exhibits a power-law degree distribution, defined as $P(k) \sim k^{-\beta}$, where $P(k)$ denotes the probability of a node with k links to other nodes (*i.e.* degree k) and β is a parameter typically varying from 2 to 3.

We conduct experiments on a network of 200 subsystems. The underlying scale-free graph was generated using the B-A algorithm [92], where the initial graph was a line with five nodes and the number of links a new node can connect was two. The resulting scale-free graph is shown in Figure 4.4(a), where $\beta = -2.187$ in its power-law degree distribution. In this graph, the number of nodes with degree < 5, < 10 and ≥ 10 are 163, 23 and 14, respectively, indicating that the connections of this graph are very sparse. Also, we can build a sparse chordal extension for the scale-free graph, and Figure 4.4(b) shows the distribution of the maximal clique size. There are 178 maximal cliques, all of which are much smaller compared to the original graph size. In our experiment, the

Table 4.2: Performance of different solvers to solve the analysis problems for a system of 200 subsystems over a scale-free netowrk.

	Time (s)			Value		
	sedumi	SCS	CDCS	sedumi	SCS	CDCS
Stability	115.0	108.9	40.6	—	—	—
$\mathcal{H}_2$	805.0	556.1	147.4	10.31	10.36	10.30
$\mathcal{H}_\infty$	3 374.8	2 130.2	172.0	1.90	2.72	1.91

—: Stability test is a feasibility problem, where the optimal objective is zero.
**: The accurate $\mathcal{H}_2$ performance returned by `norm(sys,2)` is 5.61, and the accurate $\mathcal{H}_2$ performance returned by `norm(sys,inf)` is 1.89.

dynamical data was generated randomly in a way similar to Section 4.5.1, where the state dimension n_i was chosen as 5, and the dimensions of output and disturbance (d_i, m_i) were set to 2. Table 4.2 lists the average CPU time in seconds required by different solvers over five random instances. It can be seen that the chordal decomposition approach was much faster than the standard dense methods using either SeDuMi or SCS. In particular, for the $\mathcal{H}_\infty$, exploiting chordal decomposition via CDCS was more than 10 times faster than the standard dense methods. In these cases, the optimality loss is very small in computing $\mathcal{H}_\infty$ performance using block-diagonal Lyapunov functions.

4.6 Conclusion

In this chapter, we have shown that by restricting the sparsity pattern of the Lyapunov matrix P, the SDP formulations of stability, $\mathcal{H}_2$ and $\mathcal{H}_\infty$ analysis problems can potentially inherit the graph structure of the networked system. This allows one to decompose the single large PSD cone in all of the analysis problems into multiple smaller ones, which can be solved efficiently using CDCS. When the largest maximal clique is small, the chordal decomposition approach is significantly faster than the standard dense method, as demonstrated by several numerical examples. This makes it a promising approach for large sparse systems analysis.

Note that first-order methods are typically appealing to obtain solutions with moderate accuracy, and this feature has two implications: 1) the objective is moderately accurate, and 2) the constraints are only loosely satisfied up to certain tolerance (*e.g.*, $\epsilon = 10^{-3}$ in the simulations of this chapter). Thus, the solutions from CDCS can be used to estimate $\mathcal{H}_2$ or $\mathcal{H}_\infty$ performance bounds of certain linear networked systems, as we demonstrated in this chapter. When it comes to the problem of synthesizing controllers, however, solutions with moderate accuracy might be not suitable in practice, since recovering the controller normally requires to invert a certain positive definite matrix. The inversion operation may amplify the numerical errors coming from the first-order methods. In Part II of this thesis, we develop customized algorithms to take advantage of chordal structures in the LMIs arising in the distributed control of networked systems.

Part II

Distributed Control of Networked Systems

5

Scalable design using chordal decomposition

Part I of this thesis has demonstrated the potential of chordal decomposition in operator-splitting algorithms for general sparse SDPs. The resulting solutions are typical of moderate accuracy, which might be not suitable for synthesizing controllers directly. This part of the thesis (Chapters 5 and 6) focuses on exploiting chordal decomposition to develop customized algorithms in distributed control of networked systems. The focus is especially on *solution scalability* and *model privacy*, and we take advantage of systems' sparsity and apply Theorem 2.17 to decompose LMIs arising in structured stabilization and decentralized optimal control of networked systems.

The primal objective of Chapter 5 is scalability (although model privacy can be maintained as a byproduct), and a sequential design method is proposed for structured stabilization based on the clique-intersection property of a clique tree. Chapter 6 mainly focuses on model privacy in solving an optimal decentralized control problem, where an ADMM algorithm is introduced to deal with the consensus of overlapping subsystems. One distinct difference between Chapter 5 and Chapter 6 lies in how to handle the overlapping elements among different maximal cliques: Chapter 5 applies an equal splitting strategy, and each maximal clique only needs to solve a subproblem once, then passes information along the clique tree, while Chapter 6 uses a "negotiating" process based on the ADMM framework, and each maximal clique solves subproblems iteratively to reach consensus among overlapping variables.

5.1 Introduction

Controller synthesis for interconnected systems, where multiple subsystems are interacting over a network with limited communication, has received considerable attention in recent years [93–95]. This problem arises in several applications, such as the smart grid [96], unmanned aerial vehicles [97], and automated highways [98]. 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 computationally hard. In particular, it is proved that the problem of finding a decentralized static output feedback with a norm bound is NP-hard [99]. 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, 100, 101]; 2) seeking tractable design approaches, using convex approximations [102, 103]; and 3) obtaining suboptimal solutions using non-convex optimization [104, 105]. 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 [100], Shah and Parrilo derived explicit state-space solutions by solving a number of uncoupled Ricatti equations when the performance metric was $\mathcal{H}_2$ norm. More recently, Kim and Lall [101] 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 program in the discrete-time and linear-quadratic setting: a convex relaxation can be obtained by dropping the rank constraint [102]. An alternative convex optimization objective is formulated in [103], 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 [104] and alternating direction method of multipliers (ADMM) approaches [105]. Additionally, the notions of implementability and realizability over arbitrary graphs have been introduced in [106], 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 networked 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 examples in the literature are relatively small-scale systems. In contrast, some practical networked systems, such as the electrical power grid [107] and mass transportation systems [108], could involve thousands of states and controls or more. 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.

As discussed in Section 2.2, chordal graphs are well studied in graph theory [14, 15]. 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 [16]. 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 [17]. For problems in inference and machine learning, chordal graphs have been applied to maximum likelihood estimation for sparse graphical models [18], and to generalize the message passing algorithms to graphs with cycles [109]. In the context of SDPs, Grone *et al.* [19] and Agler *et al.* [20] proved two important results (*i.e.*, Theorems 2.10 and 2.13 in Chapter 2), which relate chordal graphs to the decomposition of sparse positive semidefinite matrices. Moreover, Fukuda *et al.* [23] and Kim *et al.* [24] showed that the results in [19, 20] 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 [13], obtaining significantly faster solutions than using standard methods. Andersen *et al.* have used ideas from chordal graphs to improve the efficiency of robust stability analysis of large-scale uncertain systems [34].

In this chapter, we develop a scalable sequential design algorithm to synthesize structured feedback gains for large-scale networked systems. We use a plant graph and a communication graph to represent networked systems. This naturally results in block structured constraints in controller synthesis. Our approach exploits block matrices with chordal structure to efficiently compute structured feedback gains for large-scale systems. More specifically, we apply the block-chordal decomposition (*i.e.*, Theorem 2.17) to design structured controllers for large-scale systems in a sequential fashion. This method first uses a block-diagonal Lyapunov matrix assumption [110] to restrict 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. This strategy greatly improves the computational efficiency for large-scale sparse systems, as demonstrated by numerical experiments.

The rest of this chapter is organized as follows. Section 5.2 presents the problem statement. In Section 5.3, we introduce a simple assumption to restrict the design of structured feedback gains into a convex problem. Section 5.4 introduces the scalable sequential design algorithm, and Section 5.5 presents several illustrative examples. We conclude the chapter in Section 5.6.

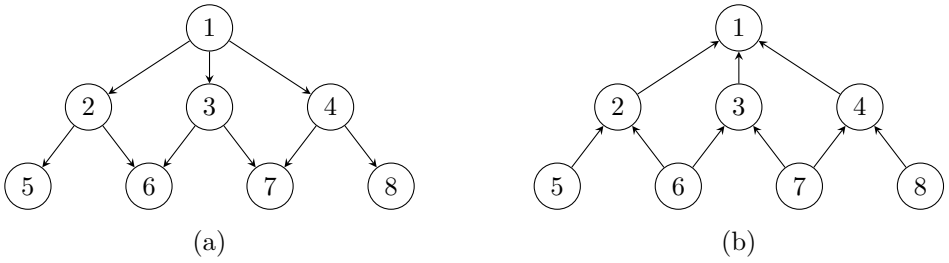


Figure 5.1: Example of hierarchical systems: (a) plant graph $\mathcal{G}^p(\mathcal{V}, \mathcal{E}^p)$ where only the subsystems in upper layer have dynamical influence on those in lower layer; (b) communication graph $\mathcal{G}^c(\mathcal{V}, \mathcal{E}^c)$, where only the nodes in upper layer can use the state information of the nodes in lower layer.

5.2 Problem statement

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 principle, a large-scale system consists of two underlying graph structures (see an example of hierarchical systems in Figure 5.1): 1) a plant graph $\mathcal{G}^p(\mathcal{V}, \mathcal{E}^p)$, determining the dynamic coupling of subsystems; 2) a communication graph $\mathcal{G}^c(\mathcal{V}, \mathcal{E}^c)$, indicating the allowable communication of local controllers between subsystems.

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 [94]; $\mathcal{G}^p$ and $\mathcal{G}^c$ share the same graph structure in [106]. Shah and Parrilo assumed these graphs could be modelled by partial order sets [100]. In particular, there are no edges in $\mathcal{G}^p$ for dynamically decoupled plants, such as in the platoon control problem [111]. Similarly, $\mathcal{G}^c$ has no edges if there exists no communication between subsystems (known as fully decentralized systems).

For each subsystem $i \in \mathcal{V}$, the state $x_i(t) \in \mathbb{R}^{\alpha_i}$ 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 $u_i(t) \in \mathbb{R}^{m_i}$ 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

$$\dot{x}(t) = Ax(t) + Bu(t), \tag{5.1}$$

where $x(t) = [x_1(t)^\mathsf{T}, \ldots, x_N(t)^\mathsf{T}]^\mathsf{T}$ and similarly for $u(t)$. Upon defining a partition $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ according to the state dimension of subsystems, and $N = \sum_i^n \alpha_i$, we have $A \in \mathbb{R}_\alpha^N(\mathcal{E}^p, 0)$. Similarly, the matrix B has a block-diagonal structure, *i.e.*, $B = \text{diag}\{B_1, \ldots, B_N\}$.

Our goal is to stabilize (5.1) by designing $u(t)$ based on local communication defined by $\mathcal{G}^c$. In this chapter, we use a static state feedback, and we assume communication conditions are perfect, *i.e.*, there are no time-delays or bandwidth restrictions. As a result, we are looking for local controllers of the form

$$u_i(t) = k_{ii}x_i(t) + \sum_{j\in\mathbb{N}_i^c} k_{ij}x_j(t), \tag{5.2}$$

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. A compact form of the overall controller is

$$u(t) = Kx(t), \quad K \in \mathcal{K}, \tag{5.3}$$

where $\mathcal{K}$ encodes the block information structure defined by $\mathcal{G}^c(\mathcal{V}, \mathcal{E}^c)$, *i.e.*,

$$\mathcal{K} = \{K \in \mathbb{R}^{\hat{m}\times N} \mid K_{ij} = 0, \text{ if } j \notin \mathbb{N}_i^c\},$$

with $\hat{m} = \sum_{i=1}^n m_i$. Then, the closed-loop system is

$$\dot{x}(t) = (A + BK)x(t), \qquad A \in \mathbb{R}_\alpha^N(\mathcal{E}^p, 0), K \in \mathcal{K}. \tag{5.4}$$

Concisely, this chapter considers the following stabilization problem

$$\begin{gathered}\text{Find } K \in \mathcal{K}, \\ \text{such that } A + BK \text{ is asymptotically stable.}\end{gathered} \tag{5.5}$$

Without the structural constraint $\mathcal{K}$, there exist well-known methods for solving (5.5). However, sparsity constraints arise naturally for synthesizing decentralized control systems. In general, such seemingly mild and natural requirements make the problem challenging [100, 104]. Previous work imposed either special structures or used certain relaxation techniques to solve this problem with a certain cost function (typically $\mathcal{H}_2$ or $\mathcal{H}_\infty$ norm) [5, 100–104].

This chapter shows that the system's sparsity in (5.4) have the potential to bring certain benefits from the perspective of numerical computations. The speed of numerically computing a controller can be improved if this sparsity is taken advantage of. Besides, it is favourable to exploit the sparsity in $\mathcal{G}^p, \mathcal{G}^c$ such that the feedback gains can be computed locally. Specifically, this chapter focuses on the structured stabilization problem (5.5), and propose a scalable sequential algorithm to solve (5.5) by exploiting properties between chordal graphs and sparse positive semidefinite matrices.

5.3 Design of structured feedback gains using convex restriction

In this section, we present a classical restriction technique [110] to convert problem (5.5) into an LMI which inherits the problem's sparsity. Accordingly, the scalable design algorithm that uses chordal decomposition can be applied.

Recall that conditions for stability can be equivalently expressed as

$$\begin{cases} QA^{\mathsf{T}} + AQ + R^{\mathsf{T}}B^{\mathsf{T}} + BR \prec 0, \\ RQ^{-1} \in \mathcal{K}, \quad Q \succ 0. \end{cases} \tag{5.6}$$

The steps to obtain (5.6) are well-known, which use a Lyapunov function $V(x) = x^{\mathsf{T}}Px, P \succeq 0$ and introduce a change of variables $Q = P^{-1}$, and $R = KQ$.

In (5.6), the structural constraint introduced by $\mathcal{G}^c$, which is nonlinear, can be relaxed if we assume that Q (and hence Q^{-1}) is block diagonal with size compatible to each subsystem, resulting in the following equivalence: $RQ^{-1} \in \mathcal{K} \Leftrightarrow R \in \mathcal{K}$. This assumption convexifies the problem (5.6) into

$$\begin{cases} QA^{\mathsf{T}} + AQ + R^{\mathsf{T}}B^{\mathsf{T}} + BR \prec 0, \\ Q \succ 0, Q \text{ is block diagonal}, \\ R \in \mathcal{K}, \end{cases} \tag{5.7}$$

but this is still centralized. The assumption that the closed-loop system admits a block-diagonal Lyapunov function would introduce conservativeness for general systems. Our motivation is that the block-diagonal assumption not only leads to a convex problem (5.7) but also endows that (5.7) has the same sparsity pattern with (5.5), allowing the subsequent chordal decomposition. Note that a notion of strongly decentralized stabilization was introduced in [110], which requires the closed-loop system to admit a block-diagonal Lyapunov function. In Appendix A, we will present some sufficient conditions to guarantee the existence of block-diagonal Lyapunov functions.

Remark 5.1. The convex problem (5.7) can be solved using general conic solvers, such as SeDuMi [71]. However, both the computational efficiency and quality of the solution will degrade for larger systems. Note that (5.7) inherits the original sparsity pattern in (5.5). There are some techniques [24, 25] that exploit chordal sparsity to improve the efficiency of solving general sparse SDPs, such as SparseCoLO [24], SMCP [25], SDPA-C [112]. Using these techniques, the efficiency of solving (5.7) can be improved to some extent. In the next section, we establish a scalable sequential algorithm based on the block-chordal decomposition Theorem 2.17 to solve (5.7) locally for sparse $\mathcal{G}^p$ and $\mathcal{G}^c$.

5.4 Scalable solution via chordal decomposition

In this section, we first present a chordal characterization of system data in (5.7), directly leading to a decomposition of the PSD constraints by applying Theorem 2.17. 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.

5.4.1 Chordal characterization of system data

Matrices A, K in (5.5) have sparsity patterns defined by $\mathcal{G}^p$ and $\mathcal{G}^c$, respectively. However, the sparsity pattern in the Lyapunov condition (5.7) is one of an undirected super-graph covering both $\mathcal{G}^p$ and $\mathcal{G}^c$. Precisely, thanks to the block diagonal assumption on Q, we have

$$AQ \in \mathbb{R}^N_\alpha(\mathcal{E}^p, 0), BR \in \mathbb{R}^N_\alpha(\mathcal{E}^c, 0). \tag{5.8}$$

To handle the symmetry in the Lyapunov condition, we use the Definition 4.2 of mirror graphs $\mathcal{G}_r(\mathcal{V}, \mathcal{E}_r)$. For example, the graphs in Figure 5.1 (a) and (b) are mirror graphs to each other. Then, we have

$$QA^\mathsf{T} \in \mathbb{R}^N_\alpha(\mathcal{E}^p_r, 0), R^\mathsf{T}B^\mathsf{T} \in \mathbb{R}^N_\alpha(\mathcal{E}^c_r, 0), \tag{5.9}$$

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

$$\mathcal{G}_s := \mathcal{G}^p \cup \mathcal{G}^p_r \cup \mathcal{G}^c \cup \mathcal{G}^c_r, \tag{5.10}$$

where $\mathcal{E}_s := \mathcal{E}^p \cup \mathcal{E}^p_r \cup \mathcal{E}^c \cup \mathcal{E}^c_r$. Combining (5.8) and (5.9) leads to

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

Next, we construct a chordal graph $\mathcal{G}_{ex}(\mathcal{V}, \mathcal{E}_{ex})$ by making a chordal extension to $\mathcal{G}_s$. Define a graph $\mathcal{G}_0(\mathcal{V}, \mathcal{E}_0)$ which only contains nodes, but no edges. Then, (5.7) can be rewritten into (5.11),

$$\begin{cases} -(QA^\mathsf{T} + AQ + R^\mathsf{T}B^\mathsf{T} + BR + \varepsilon I) \in \mathbb{S}^N_{\alpha,+}(\mathcal{E}_{ex}, 0), \\ Q - \varepsilon I \in \mathbb{S}^N_{\alpha,+}(\mathcal{E}_0, 0), \\ R \in \mathbb{R}^N_{m,n}(\mathcal{E}^c, 0), \end{cases} \tag{5.11}$$

where ε is a small positive number. Figure 5.2 illustrates the construction steps. For example, Figure 5.3 (a) shows a chordal graph $\mathcal{G}_{ex}$ for the system shown in Figure 5.1.

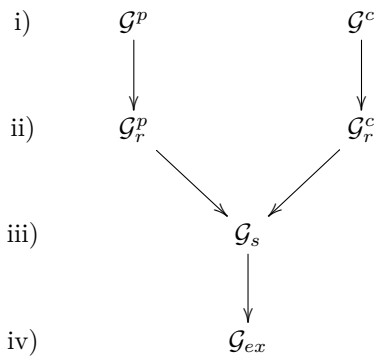


Figure 5.2: Illustrative diagram for the steps of chordal characterization. i) Define $\mathcal{G}^p, \mathcal{G}^c$ for plant and communication structure; ii) Get mirror graphs $\mathcal{G}_r^p, \mathcal{G}_r^c$; iii) Define a super-graph $\mathcal{G}_s$ to characterize the whole structure; iv) Finally, obtain $\mathcal{G}_{ex}$ by making a chordal extension to $\mathcal{G}_s$.

5.4.2 Decomposition of positive semidefinite constraints

Having established the chordal characterization, we now turn to apply Theorem 2.17 to decompose the large PSD constraint in (5.11).

Let $\Gamma = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_p\}$ be the set of maximal cliques in graph $\mathcal{G}_{ex}$, and $\mathcal{T} = (\Gamma, \Xi)$ with $\Xi \subseteq \Gamma \times \Gamma$ be a clique tree that satisfies the clique intersection property. In (5.11), for notational simplicity, define

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

Then, according to Theorem 2.17, we can equivalently reduce (5.11) into

$$\begin{cases} \sum_{k=1}^{p} E_{\mathcal{C}_k,\alpha}^{\mathsf{T}} J_k E_{\mathcal{C}_k,\alpha} = J_{Q,R}, \\ J_k \in \mathbb{S}_+^{|\mathcal{C}_k|\alpha}, k = 1, \ldots, p, \\ Q - \varepsilon I \in \mathbb{S}_{\alpha,+}^{N}(\mathcal{E}_0, 0), \\ R \in \mathcal{K}. \end{cases} \tag{5.12}$$

Note that (5.12) only involves a set of PSD constraints of small size (corresponding to maximal cliques) instead of one large PSD constraint in (5.11). The price is that additional equality constraints are added in (5.12). We further relax these equality constraints, resulting in the sequential design method in the next subsection.

5.4.3 Sequential design 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 clique intersection property.

Note that the additional equality constraints in (5.12) only affect overlapping elements in $\mathcal{G}_{ex}$. If there are no overlapping elements, meaning that the maximal cliques are

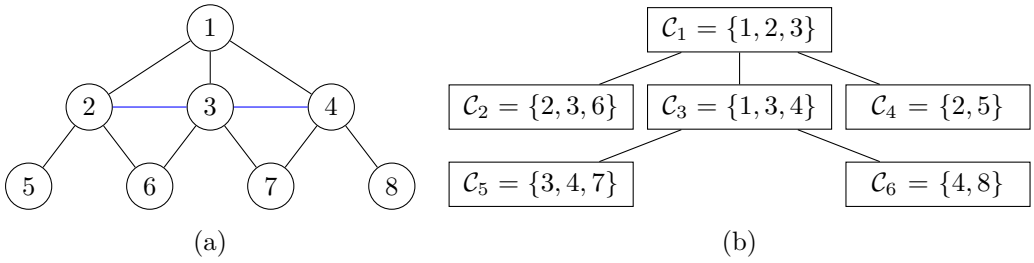


Figure 5.3: Chordal extension and clique tree for the hierarchical system in Figure 5.1. (a) Chordal graph $\mathcal{G}_{ex}$, where two undirected edges (in blue) are added. (b) a clique tree. For the breadth-first tree traversal, we start from the root node $\mathcal{C}_1$, and then explore the neighbouring cliques $\mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4$ in the second layer before moving to the next level neighbours $\mathcal{C}_5, \mathcal{C}_6$.

disjoined, then the design of structured feedback gains can be naturally decomposed into several small sub-problems according to the maximal cliques. In general cases, 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 $\sum_{k=1}^{p} E_{\mathcal{C}_k,\alpha}^{\mathsf{T}} J_k E_{\mathcal{C}_k,\alpha} = J_{Q,R}$ in (5.12) such that this constraint is satisfied.

Here, we introduce a formal description of the aforementioned idea:

Step 1: Obtain an averaging factor for overlapping elements

Given $\Gamma = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_p\}$ as the set of maximal cliques in $\mathcal{G}_{ex}$, we define $\gamma \in \mathbb{S}^n$ as

$$\begin{cases} \gamma_{ii} = \text{the number of times node } i \text{ appears in } \Gamma, \\ \gamma_{ij} = \text{the number of times nodes } i, j \text{ appears in } \Gamma. \end{cases}$$

It is not difficult to see that $\gamma \in \mathbb{S}^n(\mathcal{E}_{ex}, 0)$. Accordingly, we define an averaging factor $\beta \in \mathbb{S}_{\alpha}^{N}(\mathcal{E}_{ex}, 0)$ as

$$\begin{cases} \beta_{ij} = \dfrac{1}{\gamma_{ij}} 1_{\alpha_i \times \alpha_j}, & \text{if } \gamma_{ij} \neq 0, \\ \beta_{ij} = 0_{\alpha_i \times \alpha_j}, & \text{otherwise .} \end{cases}$$

where $1_{\alpha_i \times \alpha_j}$ is an $\alpha_i \times \alpha_j$ 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 (5.12) as

$$J_k := E_{\mathcal{C}_k,\alpha}^{\mathsf{T}} (J_{Q,R} \circ \beta) E_{\mathcal{C}_k,\alpha}, k = 1, \ldots, p, \tag{5.13}$$

where $\circ$ denotes the Hadamard product. Based on this construction, we naturally have $\sum_{k=1}^{p} J_k = J_{Q,R}$. Thus, (5.12) is reduced into a set of small-size LMIs $\mathcal{L}_k, k = 1, \ldots, p$ over maximal cliques, where each $\mathcal{L}_k$ is defined as

$$\mathcal{L}_k : \begin{cases} E_{\mathcal{C}_k,\alpha}^{\mathsf{T}} (J_{Q,R} \circ \beta) E_{\mathcal{C}_k,\alpha} \succeq 0, \\ Q_j - \varepsilon I \succeq 0, j \in \mathcal{C}_k, \end{cases} \tag{5.14}$$

Step 3: Sequential solution over a clique tree

The dimension of $\mathcal{L}_k$ depends on the size of clique $\mathcal{C}_k$. There may exist some common design parameters among different $\mathcal{L}_k$. We can solve them clique-by-clique over a clique tree $\mathcal{T}$ in a sequential way. 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 clique 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: 1) depth-first, which starts at the root, and explores as far as possible along each branch before backtracking; 2) 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. In the simulation, we used the breadth-first strategy. To demonstrate this strategy, consider the example shown in Figure 5.3. 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. 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.

In the sequential design, the size of the resulting SDPs is determined by the size of maximal cliques. When the largest maximal clique size is bounded, the computational complexity of each small-scale problem $\mathcal{L}_k$ in (5.14) is constant. Also, chordal graphs have at most n maximal cliques [15]. Thus, the computational complexity of solving the SDP sequentially scales linearly with the graph size.

Remark 5.2. Our sequential method heavily depends on a clique tree of the chordal graph. 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 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 [113].

5.4.4 Guaranteed minimum decay rate

One straightforward extension of the sequential design method is to add a guaranteed performance of minimum decay rate, using the following result:

Lemma 5.3. 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 (5.1).

$$\begin{cases} (AQ + BR) + (AQ + BR)^{\mathsf{T}} + \alpha Q \prec 0, \\ Q \succ 0, Q \text{ is block diagonal}, \\ R \in \mathcal{K}. \end{cases} \tag{5.15}$$

Also, this formulation has the same sparsity pattern with (5.7), 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, we can add certain bounds on local feedback gains

$$Q_i \succ \kappa_Q I, \begin{bmatrix} -\kappa_R I & R_{ij}^{\mathsf{T}} \\ R_{ij} & -I \end{bmatrix} \prec 0, \tag{5.16}$$

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 5.5.2, a network of coupled inverted pendula is used to demonstrate this extension of guaranteed decay rate and bounded feedback gains.

5.5 Illustrative examples

In this section, we present three illustrative examples to demonstrate the scalability of the proposed sequential design method[1]. 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 then 10^{-8}.

Table 5.1: Computing sequences, structured gains and computing time for the hierarchical system shown in Figure 5.1.

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

5.5.1 Hierarchical systems

Consider the hierarchical system shown in Figure 5.1. As used in [105], we assume each node is a second order unstable system coupled with its neighbouring nodes through an exponentially decaying function of the Euclidean distance $\alpha(i,j)$ between them,

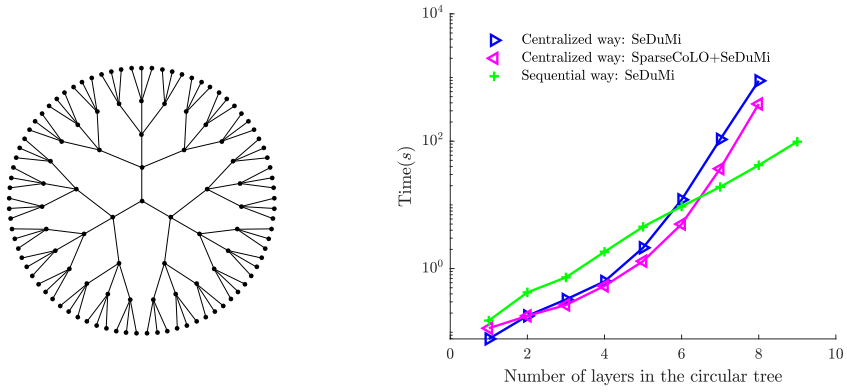
$$\dot{x}_i = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} x_i + \sum_{j \in \mathbb{N}_i^p} e^{-\alpha(i,j)} x_j + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_i, \tag{5.17}$$

where $\alpha(i,j)$ is $\frac{1}{10}(i-j)^2$ in the simulation. We first computed structured feedback gains for this problem in a centralized way (see Section 5.3), which took about 0.087 s to get a stabilizing controller as follows:

- 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: $k_{44} = -[9.64, 4.79], k_{47} = -[0.23, 0.46], k_{48} = -[0.11, 0.23]$,
- 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. The corresponding chordal extension and clique tree are shown in Figure 5.3. Table 5.1 lists the solving sequences, computed feedback gains and time consumption for each clique.

[1] The numerical examples presented in this section and additional examples are available from https://github.com/zhengy09/sdsc.



(a) (b)

Figure 5.4: (a) Hierarchical systems over a circular tree with 4 layers and 3 branches. The information flow is bottom-up but only dynamics of nodes in the upper layer have influence on those in the lower layer. (b) Time consumption (in seconds) comparison for solving the structured feedback gains over circular trees.

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 confirmed in TABLE 5.1. 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 Figure 5.4(a)). The dynamics flow is top-down, while the information flow is bottom-up. Each node is assumed to have dynamics evolving as in (5.17). Figure 5.4(b) shows a comparison between the centralized method (Section 5.3) and the sequential method (Section 5.4) for different number of layers (where each node has two branches). In the simulation, we used two different ways, *i.e.*, SeDuMi and SparseCoLO+SeDuMi, to solve the resulting SDP in the centralized method. SparseCoLO can detect chordal sparsity in general SDPs and then call SeDuMi to solve the problem. As shown in Figure 5.4(b), even though chordal sparsity has been exploited in SparseCoLO, the proposed sequential method is more efficient in computing structured gains for large-scale systems.

5.5.2 A practical example: coupled inverted pendula

Here, we consider a practical example — a network of three coupled inverted pendula (see Figure 5.5) — to demonstrate the extension with minimum decay rate and bounds on the feedback gains. The linearized dynamics around the upright equilibrium point can be described as [114]:

$$A_i = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{M_i+m}{M_i l} g & 0 & \frac{k_i}{M_i l} & \frac{c_i+b_i}{M_i l} \\ 0 & 0 & 0 & 1 \\ -\frac{m}{M_i} g & 0 & -\frac{k_i}{M_i} & -\frac{c_i+b_i}{M_i} \end{bmatrix}, \quad A_{ij} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{k_{ij}}{M_i l} & \frac{b_{ij}}{M_i l} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{k_{ij}}{M_i} & \frac{b_{ij}}{M_i} \end{bmatrix}, \quad B_i = \begin{bmatrix} 0 \\ -\frac{1}{M_i l} \\ 0 \\ \frac{1}{M_i} \end{bmatrix},$$

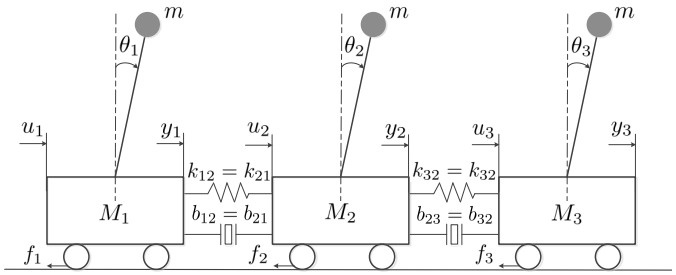


Figure 5.5: A network of three coupled inverted pendula.

(a) (b)

Figure 5.6: Chordal decomposition of the coupled inverted pendula: (a) maximal cliques; (b) clique tree.

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 $x_i = [\theta_i, \dot{\theta}_i, y_i, \dot{y}_i]^\mathsf{T}$, and $c_i, b_{ij} = b_{ji}, k_{ij} = k_{ji}$ are friction, damper and spring coefficients, respectively. We have assumed that the moments of inertia of the pendula are zero.

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. Figure 5.6 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 rete 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 (5.15) centrally. This way requires all of the model information. Alternatively, we can solve (5.15) according to the clique tree shown in Figure 5.6 (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 Figure 5.7 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 $x_3(0) = 0$.

5.5.3 General networked systems

Finally, we present simulation results for networked systems over general directed graphs. It is assumed that each node is an unstable second order system coupled with its neighbouring nodes, as shown in (5.17). In the simulation, we first generated a random chordal graph $\mathcal{G}_1 = (\mathcal{V}, \mathcal{E}_1)$ with a bound on its largest maximal clique size, and then randomly removed

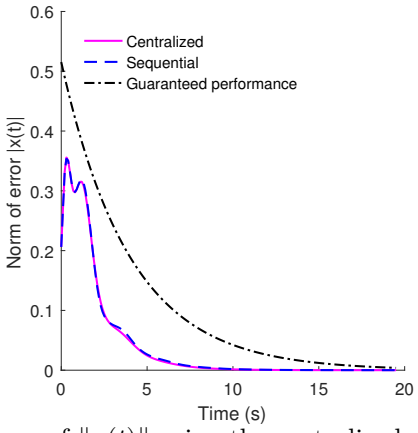
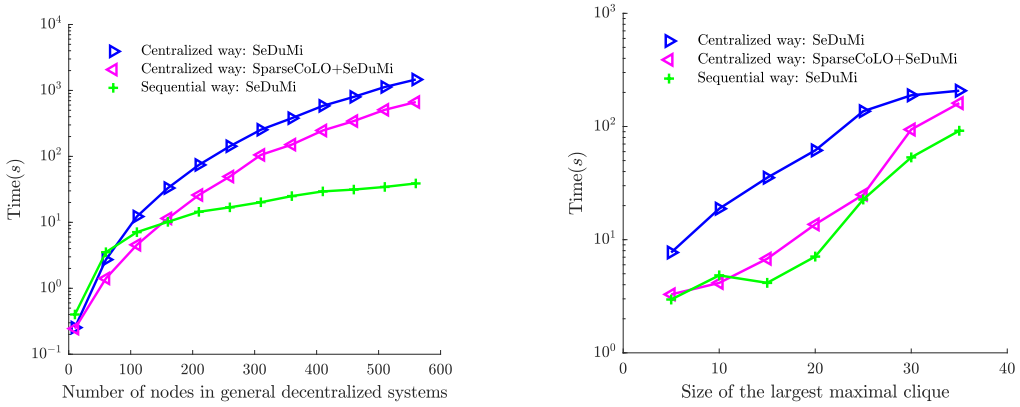


Figure 5.7: Exponential decay of $\|x(t)\|$ using the centralized computation and the sequential computation.



(a) (b)

Figure 5.8: Time consumption (in seconds) comparison for solving structured feedback gains: (a) general systems with bounded maximal clique size (the largest maximal clique size is five); (b) general systems with 100 nodes when varying the largest maximal clique size.

some edges of $\mathcal{G}_1$ to form the plant graph $\mathcal{G}^p$. We ensured 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 a bounded size. When this value is set as five, Figure 5.8(a) shows a comparison between the centralized method and sequential method for different graph sizes. The result 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 Figure 5.8(b), the time consumption increases as the largest maximal clique size increases for the centralized method and sequential design. As expected, the efficiency of the sequential design method becomes worse as the size of largest maximal clique increases. This illustrates that the efficiency of the proposed sequential design method is determined by the largest maximal clique size in the extended graph. If this is small and independent of graph size, the sequential design method scales a lot better.

5.6 Conclusion

This chapter considered the synthesis of structured static feedback gains for large-scale systems over directed graphs. A sequential design method has been proposed by exploiting the chordal decomposition of block PSD matrices, which greatly improves the computational scalability for large-scale applications. However, it should be noted that the sequential design indeed brings some conservatism, particularly in the equal splitting strategy. This strategy may compromise the feasibility of (5.7): the feasibility of the sequential design leads to the feasibility of (5.7), while the converse is not true for general networked systems. This is mainly due to the unidirectional solving process, *i.e.*, the information is non-reversing, passing down along the clique tree.

In the next chapter, we will introduce a negotiating process based on the ADMM framework, which can preserve the feasibility of (5.7) and incorporate certain global performance index, such as $\mathcal{H}_2$ norm.

6

Distributed design of decentralized controllers

Synthesizing decentralized controllers in a distributed fashion is desirable due to privacy concerns of model data in certain complex systems. In this chapter, we propose a distributed design method for optimal decentralized control by exploiting the underlying sparsity properties of the problem. Our method is based on chordal decomposition of sparse block matrices and the alternating direction method of multipliers (ADMM). We first apply a classical parameterization technique to restrict the optimal decentralized control into a convex problem that inherits the sparsity pattern of the original problem. Then, chordal decomposition allows us to decompose the convex restriction into a problem with partially coupled constraints, and the framework of ADMM enables us to solve the decomposed problem in a distributed fashion. Consequently, the subsystems only need to share their model data with their direct neighbours, not centrally or globally.

6.1 Introduction

When regulating a complex system of multiple interconnected subsystems, it is sometimes advantageous to design control laws relying only on the local subsystem's state or output due to implementation and/or operation costs. This approach is known as the *decentralized control* of complex systems [1], which has attracted research attention since the late seventies.

Early efforts have centered on decentralized stabilization and its algebraic characterization of *decentralized fixed modes* [115]. One seminal result is that a system is stabilizable by a decentralized controller if and only if its decentralized fixed modes have negative real parts [115]. Since then, a wide range extensions of decentralized control have been investigated, either by considering various types of performance guarantees [110] or by taking into account neighbouring information for feedback [116]. Several classes of plants have been identified, which allow convex formulations for the design of decentralized $\mathcal{H}_\infty$ and $\mathcal{H}_2$ controllers, such as quadratic invariant systems [5] and poset-casual systems [100].

Also, some numerical approaches have been proposed to find an approximate solution to the optimal decentralized control problem [102, 104]. A common assumption made in these papers is that a central model of the global plant is available, indicating that the design is performed in a centralized fashion even though the implementation of controllers is decentralized (a notable exception is in [100], which obtained independent decoupled problems by utilizing the properties of posets). However, this may be impractical for certain complex systems that are shared between private individuals, such as transportation systems and power-grids. For example, the formation control of autonomous vehicles is to regulate and maintain a desired distance between adjacent vehicles. In this case, a complete platoon model may not be available due to privacy concerns of model information among the vehicles. In addition, it could be desirable to design controllers based on local model information to reduce the effects of merging with other vehicles and splitting platoons on controller synthesis.

In fact, a notion of *distributed design* of controllers relying on limited model information has been introduced in [117], where performance bounds of designing linear quadratic regulators distributedly were discussed for systems with invertible input matrix (early discussions can be traced back to [118]). This framework has been used to discuss the best closed-loop performance achievable by distributed design strategies for a class of fully actuated discrete-time systems [119]. Also, numerical algorithms for distributed design have been proposed using dual decomposition and gradient descent methods in the framework of model predictive control [120, 121]. More recently, Ahmadi *et al.* proposed a distributed synthesis method for linear continuous time systems based on the framework of dissipative systems [122], and Wang *et al.* introduced a separable and localized design method for linear discrete time systems based on the system level approach [123].

In this chapter, we propose a new distributed design method for optimal decentralized control by exploiting the sparsity structure of the system. Our idea originates from the connection between sparse positive semidefinite matrices and chordal graphs [19, 20]. The celebrated chordal decomposition in graph theory [19, 20] allows us to decompose a large sparse positive semidefinite cone into a set of smaller and coupled ones, and has been successfully applied in optimization to decompose sparse semidefinite programs (SDPs) [23, 25, 36]. These results have recently been used for performance analysis of sparse linear systems [13, 34], leading to significantly faster solutions than using standard dense methods. In Chapter 5, a sequential approach has been introduced for scalable design of structured controllers using the properties of clique trees in chordal graphs.

This chapter extends the scope of [13, 34] and Chapter 5 on exploiting chordal decomposition to distributed design of optimal decentralized control. Our main contribution is to combine chordal decomposition with a first-order algorithm to synthesize decentralized

controllers in a distributed fashion. Specifically, by using a classical parameterization technique [110], the optimal decentralized control can be restricted to a convex problem that inherits the original sparsity pattern in the system. Upon assuming that an undirected version of the system graph is chordal, the convex restriction can be equivalently decomposed into a problem with partially coupled constraints. Then, we introduce a distributed algorithm to solve the decomposed problem using ADMM. In our algorithm, no central model of the global plant is required and the subsystems only need to share their model data with their neighbours, which helps preserve the privacy of model data.

The rest of this chapter is organized as follows. In Section 6.2, we present the problem formulation. Chordal decomposition in optimal decentralized control is discussed in Section 6.3. Section 6.4 introduces a distributed algorithm, and numerical examples are given in Section 6.5. We conclude this chapter in Section 6.6.

6.2 Problem statement

We consider a complex system consisting of N subsystems. The interactions between subsystems are modeled by a plant graph $\mathcal{G}_p(\mathcal{V}, \mathcal{E}_p)$, in which each node in $\mathcal{V}$ denotes a subsystem, and the edge $(i, j) \in \mathcal{E}_p$ means that subsystem i has dynamical influence on subsystem j. The dynamics of each subsystem $i \in \mathcal{V}$ are

$$\dot{x}_i(t) = A_{ii}x_i(t) + \sum_{j\in\mathbb{N}_i} A_{ij}x_j(t) + B_i u_i(t) + M_i d_i(t), \tag{6.1}$$

where $x_i \in \mathbb{R}^{\alpha_i}, u_i \in \mathbb{R}^{m_i}, d_i \in \mathbb{R}^{q_i}$ denote the local state, input and disturbance of subsystem i, respectively, and $\mathbb{N}_i$ denotes the set of neighbouring nodes that influence node i. In (6.1), $A_{ii} \in \mathbb{R}^{\alpha_i\times\alpha_i}, B_i \in \mathbb{R}^{\alpha_i\times m_i}, M_i \in \mathbb{R}^{\alpha_i\times q_i}$ represent local dynamics, and $A_{ij} \in \mathbb{R}^{\alpha_i\times\alpha_j}$ represents the interaction with neighbors. In this chapter, we refer to A_{ii}, B_i, M_i, A_{ij} as *model data* of the system.

By collecting the subsystems' states, the overall system can be described compactly as

$$\dot{x}(t) = Ax(t) + Bu(t) + Md(t), \tag{6.2}$$

where $x := [x_1^\mathsf{T}, x_2^\mathsf{T}, \ldots, x_N^\mathsf{T}]^\mathsf{T}$, and the vectors u, d are defined similarly. The matrix A is composed of blocks A_{ij}, which has a block sparsity pattern, *i.e.*, $A \in \mathbb{R}^{n\times n}(\mathcal{E}_p, 0)$. The matrices B, M enjoy block-diagonal structures. Our goal is to design a decentralized static state feedback

$$u_i(t) = -K_i x_i(t), i = 1, \ldots, N \tag{6.3}$$

such that the $\mathcal{H}_2$ norm of the transfer function T_{dz} from disturbance d to a certain performance output z is minimized. In (6.3), the global K has an information structure $\mathcal{K}$, defined as

$$K \in \mathcal{K} = \begin{bmatrix} \star & & \\ & \ddots & \\ & & \star \end{bmatrix}, \tag{6.4}$$

where $\star$ denotes a nonzero block of compatible dimensions.

Then, the problem considered in this chapter is

$$\begin{array}{ll} \underset{K}{\text{minimize}} & \|T_{dz}\|^2 \\ \text{subject to} & K \in \mathcal{K}, \end{array} \tag{6.5}$$

where $\|\cdot\|$ is the $\mathcal{H}_2$ norm of a transfer function. In this chapter, the performance output z is chosen as

$$z = \begin{bmatrix} Q^{\frac{1}{2}} \\ 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ R^{\frac{1}{2}} \end{bmatrix} u,$$

where $Q := \mathrm{diag}(Q_1, \ldots, Q_N)$ and $R := \mathrm{diag}(R_1, \ldots, R_N)$ denote the state and control performance weights, respectively, and diagonal block $Q_i \succeq 0, R_i \succ 0$ correspond to the subsystem i. Adopting the same terminology in [1, 110], we refer to (6.5) as the *optimal decentralized control* problem.

The constraint $\mathcal{K}$ in fact makes the optimal decentralized problem (6.5) non-convex, which is challenging to solve exactly in general. Some previous work either imposed special structures on the system [5, 100], or used certain relaxation techniques [102, 110], or applied non-convex optimization directly [104] to address this problem. These techniques, however, are essentially performing a *centralized design* of decentralized controllers, since they typically require a central unit to collect the global model data, *i.e.*, A, B, M in (6.2). We note that a few recent papers were focused on distributed synthesis using limited model information; *e.g.*, [117, 119, 122, 123].

In this chapter, we introduce a new algorithm that realizes *distributed design* of decentralized controllers by combining chordal decomposition with ADMM. In our algorithm, no centralized model data are required; instead, the knowledge about global model (6.2) can be distributed among subsystems depending on the maximal cliques of an undirected version of plant graph $\mathcal{G}_p(\mathcal{V}, \mathcal{E}_p)$.

6.3 Chordal decomposition in optimal decentralized control

In this section, similar to Chapter 5, we first employ a classical parameterization trick [110] to derive a convex restriction of (6.5). Then, we apply the block-chordal decomposition (*i.e.*, Theorem 2.17) in the relaxed problem, leading to a problem with partially coupled constraints.

6.3.1 Convex restriction of the optimal decentralized control problem

Lemma 6.1 ([10])**.** Consider a stable linear system $\dot{x}(t) = Ax(t) + Md(t), z(t) = Cx(t)$. The $\mathcal{H}_2$ norm of the transfer function from d to z can be computed by

$$\|G_{dz}\|^2 = \inf_{X \succ 0} \{\text{Trace}(CXC^\mathsf{T}) \mid AX + XA^\mathsf{T} + MM^\mathsf{T} \preceq 0\}.$$

When applying the decentralized controller (6.3), the closed-loop system of (6.2) becomes

$$\begin{aligned} \dot{x}(t) &= (A - BK)x(t) + Md(t), \\ z(t) &= \begin{bmatrix} Q^{\frac{1}{2}} \\ -R^{\frac{1}{2}}K \end{bmatrix} x(t), K \in \mathcal{K}. \end{aligned}$$

According to Lemma 6.1, the optimal decentralized control problem (6.5) can be equivalently reformulated as

$$\begin{aligned} \underset{X,K}{\text{minimize}} \quad & \text{Trace}\big((Q + K^\mathsf{T}RK)X\big) \\ \text{subject to} \quad & (A - BK)X + X(A - BK)^\mathsf{T} + MM^\mathsf{T} \preceq 0, \\ & X \succ 0, K \in \mathcal{K}. \end{aligned} \tag{6.6}$$

The first inequality in (6.6) does not linearly depend on X and K. A standard change of variables $Z = KX$ leads to

$$\begin{aligned} \underset{X,Z}{\text{minimize}} \quad & \text{Trace}\left(QX + RZX^{-1}Z^\mathsf{T}\right) \\ \text{subject to} \quad & (AX - BZ) + (AX - BZ)^\mathsf{T} + MM^\mathsf{T} \preceq 0, \\ & X \succ 0, ZX^{-1} \in \mathcal{K}. \end{aligned} \tag{6.7}$$

To handle the nonlinear constraint $ZX^{-1} \in \mathcal{K}$, a classical parameterization idea [110] is to assume a *block diagonal* $X = \text{diag}(X_1, \ldots, X_N)$ with block size compatible to the subsystem's dimensions, resulting in the following equivalence:

$$ZX^{-1} \in \mathcal{K} \Leftrightarrow Z \in \mathcal{K}.$$

Considering the block-diagonal structures of Q, R, we have

$$\text{Trace}\left(QX + RZX^{-1}Z^\mathsf{T}\right) = \sum_{i=1}^{N} \text{Trace}\left(Q_iX_i + R_iZ_iX_i^{-1}Z_i^\mathsf{T}\right).$$

By introducing $Y_i \succeq Z_iX_i^{-1}Z_i^\mathsf{T}$ and using the Schur complement [10], a convex restriction to (6.5) is derived:

$$\begin{aligned} \underset{X_i,Y_i,Z_i}{\text{minimize}} \quad & \sum_{i=1}^{N} \text{Trace}\left(Q_iX_i + R_iY_i\right) \\ \text{subject to} \quad & (AX - BZ) + (AX - BZ)^\mathsf{T} + MM^\mathsf{T} \preceq 0, && \text{(6.8a)} \\ & \begin{bmatrix} Y_i & Z_i \\ Z_i^\mathsf{T} & X_i \end{bmatrix} \succeq 0, X_i \succ 0, i = 1, \ldots, N. && \text{(6.8b)} \end{aligned}$$

Problem (6.8) is convex and ready to be solved using general conic solvers, and the decentralized controller is recovered as $K_i = Z_i X_i^{-1}, i = 1, \ldots, N$. However, solving (6.8) directly requires the global model knowledge, implicitly assuming the existence of a central entity to collect the complete model data. In this chapter, we make the following assumption.

Assumption 6.2. Problem (6.8) is feasible, or equivalently system (6.2) is strongly decentralized stabilizable [110].

Remark 6.3. The block-diagonal strategy was formally discussed in early 1990s [110], which was later implicitly or explicitly used in the field of decentralized stabilization [1, 41, 122]. This strategy essentially requires the closed-loop system to admit a block-diagonal Lyapunov function $V(x) = x^\mathsf{T} X x = \sum_{i=1}^N x_i^\mathsf{T} X_i x_i$. As in Chapter 5, our motivation here is to obtain a convex restriction of the non-convex problem (6.5) that inherits the original sparsity pattern, facilitating a distributed algorithm for the solution. In Appendix A, we present more discussions on block-diagonal Lyapunov functions.

6.3.2 Chordal decomposition of the restriction problem

In (6.8), the variables X_i, Y_i, Z_i are coupled by the inequality (6.8a) only, while the rest of constraints and the objective function are naturally separable due to the separable performance weights Q, R. Meanwhile, thanks to the block-diagonal assumption on X, the coupled linear matrix inequality has a structured sparsity pattern characterized by an undirected version of graph $\mathcal{G}_p(\mathcal{V}, \mathcal{E}_p)$. Precisely, we define an undirected graph $\mathcal{G}_u(\mathcal{V}, \mathcal{E}_u)$ with $\mathcal{E}_u = \mathcal{E}_p \cup \mathcal{E}_r$, where $\mathcal{E}_r$ denotes the edge set of the mirror graph of $\mathcal{G}_p$ (see Definition 4.2).

Here, we make the following assumption.

Assumption 6.4. Graph $\mathcal{G}_u$ is chordal.

Remark 6.5. Similar to Chapter 5, if $\mathcal{G}_u$ is not chordal, we can add suitable edges to $\mathcal{E}_u$ to obtain a chordal graph [15]. In this case, sharing model data with directed neighbours in $\mathcal{G}_p$ is not sufficient for the proposed distributed solution. Still, privacy of model data is maintained within each maximal clique $\mathcal{C}_k$.

We denote by $\mathcal{C}_1, \ldots, \mathcal{C}_p$ the maximal cliques of $\mathcal{G}_u$. Considering the inherent structure of system (6.1), it is straightforward to see that

$$(AX - BZ) + (AX - BZ)^\mathsf{T} + MM^\mathsf{T} \in \mathbb{S}^N(\mathcal{E}_u, 0).$$

To ease the exposition, we define

$$F(X, Z) := -(AX - BZ) - (AX - BZ)^\mathsf{T} - MM^\mathsf{T}. \tag{6.9}$$

Then, according to the block-chordal Theorem 2.17, $F(X,Z) \succeq 0$ is equivalent to the condition that there exist $J_k \in \mathbb{S}_+^{|\mathcal{C}_k|_\alpha}, k = 1,\dots,p$, such that

$$F(X,Z) = \sum_{k=1}^{p} E_{\mathcal{C}_k,\alpha}^{\mathsf{T}} J_k E_{\mathcal{C}_k,\alpha}. \tag{6.10}$$

Therefore, (6.8) can be equivalently decomposed into

$$\begin{aligned} \underset{X_i,Y_i,Z_i,J_k}{\text{minimize}} \quad & \sum_{i=1}^{N} \text{Trace}\,(Q_i X_i + R_i Y_i) \\ \text{subject to} \quad & \sum_{k=1}^{p} E_{\mathcal{C}_k,\alpha}^{\mathsf{T}} J_k E_{\mathcal{C}_k,\alpha} = F(X,Z), \\ & \begin{bmatrix} Y_i & Z_i \\ Z_i^{\mathsf{T}} & X_i \end{bmatrix} \succeq 0, X_i \succ 0, i = 1,\dots,N, \\ & J_k \succeq 0, k = 1,\dots,p. \end{aligned} \tag{6.11}$$

One notable feature of (6.11) is that the global constraint (6.8a) is replaced by a set of small coupled constraints (6.10). In other words, (6.11) has partially coupled constraints, which can be solved in a distributed way by introducing appropriate consensus variables.

6.4 A distributed solution via ADMM

This section introduces an ADMM algorithm to solve (6.11), which leads to local subproblems for maximal cliques and overlapping elements in $\mathcal{G}_u$. ADMM is a first-order operator-splitting method that solves an optimization problem of the form [72]

$$\begin{aligned} \underset{x,y}{\text{minimize}} \quad & f(x) + g(y) \\ \text{subject to} \quad & Ex + Fy = c, \end{aligned} \tag{6.12}$$

where $x \in \mathbb{R}^{n_x}, y \in \mathbb{R}^{n_y}$ are decision variables, f and g are real valued convex functions, and $E \in \mathbb{R}^{n_c \times n_x}, F \in \mathbb{R}^{n_c \times n_y}$ and $c \in \mathbb{R}^{n_c}$ are problem data. Given a penalty parameter $\rho > 0$, the scaled ADMM algorithm solves (6.12) using the following iterations

$$\begin{aligned} x^{(h+1)} &= \arg\min_x \; f(x) + \frac{\rho}{2}\|Ex + Fy^{(h)} - c + \lambda^{(h)}\|^2, \\ y^{(h+1)} &= \arg\min_y \; g(y) + \frac{\rho}{2}\|Ex^{(h+1)} + Fy - c + \lambda^{(h)}\|^2, \\ \lambda^{(h+1)} &= \lambda^h + Ex^{(h+1)} + Fy^{(h+1)} - c, \end{aligned}$$

where $\lambda \in \mathbb{R}^{n_c}$ is a scaled dual variable, and h denotes the iteration index.

If there is no overlapping among the cliques $\mathcal{C}_1,\dots,\mathcal{C}_p$ (*i.e.*, the system (6.2) is composed by dynamically disjoint components), then (6.11) is trivially decomposed into p decoupled subproblems of decentralized optimal control, which can be solved by cliques $\mathcal{C}_1,\dots,\mathcal{C}_p$ independently. In the case where different cliques share some common nodes with each other, we can introduce appropriate auxiliary variables to achieve a distributed solution using ADMM.

6.4.1 A simple example

For the convenience of illustration, we first consider an interconnected system characterized by a chain of three nodes. In this case, the model data are $B = \mathrm{diag}\{B_1, B_2, B_3\}, M = \mathrm{diag}\{M_1, M_2, M_3\}$ and

$$A = \begin{bmatrix} A_{11} & A_{12} & 0 \\ A_{21} & A_{22} & A_{23} \\ 0 & A_{32} & A_{33} \end{bmatrix}.$$

There are two cliques in this case, and $J_k, k = 1, 2$ in (6.10) are in the following form

$$\begin{aligned} J_1(X_1, X_2, Z_1, J_{22,1}) &:= -\begin{bmatrix} J_{11} & A_{12}X_2 + X_1 A_{21}^{\mathsf{T}} \\ * & J_{22,1} \end{bmatrix}, \\ J_2(X_2, X_3, Z_3, J_{22,2}) &:= -\begin{bmatrix} J_{22,2} & A_{23}X_3 + X_2 A_{32}^{\mathsf{T}} \\ * & J_{33} \end{bmatrix}, \end{aligned}$$

where $*$ denotes the corresponding symmetric part and

$$\begin{aligned} J_{11} &:= A_{11}X_1 - B_1 Z_1 + (A_{11}X_1 - B_1 Z_1)^{\mathsf{T}} + M_1 M_1^{\mathsf{T}}, \\ J_{33} &:= A_{33}X_3 - B_3 Z_3 + (A_{33}X_3 - B_3 Z_3)^{\mathsf{T}} + M_3 M_3^{\mathsf{T}}. \end{aligned}$$

The coupling effect is imposed on the overlapping node 2:

$$J_{22,1} + J_{22,2} = A_{22}X_2 - B_2 Z_2 + (A_{22}X_2 - B_2 Z_2)^{\mathsf{T}} + M_2 M_2^{\mathsf{T}}. \tag{6.13}$$

For any coupling variables that appear in two cliques, we introduce auxiliary variables. For this case, we introduce auxiliary variables for node 2

$$J_{22,1} = \hat{J}_{22,1},\ J_{22,2} = \hat{J}_{22,2},\ X_2 = X_{2,1},\ X_2 = X_{2,2}. \tag{6.14}$$

Also, we split the variables according to the cliques and the overlapping node

$$\begin{aligned} \text{Node 2:} \quad & y := \{X_2, Y_2, Z_2, \hat{J}_{22,1}, \hat{J}_{22,2}\}, \\ \text{Clique } \mathcal{C}_1\text{:} \quad & x_{\mathcal{C}_1} := \{X_1, Y_1, Z_1, X_{2,1}, J_{22,1}\}, \\ \text{Clique } \mathcal{C}_2\text{:} \quad & x_{\mathcal{C}_2} := \{X_3, Y_3, Z_3, X_{2,2}, J_{22,2}\}. \end{aligned}$$

Next, we show that (6.11) can be rewritten into the standard ADMM form (6.12) by defining indicator functions as

$$\begin{aligned} \mathbb{I}_{\mathcal{S}_k}(x_{\mathcal{C}_k}) &:= \begin{cases} 0, & x_{\mathcal{C}_k} \in \mathcal{S}_k, \\ +\infty, & \text{otherwise}, \end{cases} \\ \mathbb{I}_{\mathcal{L}}(y) &:= \begin{cases} 0, & y_l \in \mathcal{L}, \\ +\infty, & \text{otherwise}, \end{cases} \end{aligned} \tag{6.15}$$

for $k = 1, 2$, where sets $\mathcal{S}_1, \mathcal{S}_2$ are defined as

$$\mathcal{S}_1 := \left\{ x_{\mathcal{C}_1} \middle| J_1(X_1, X_{2,1}, Z_1, J_{22,1}) \succeq 0, X_1 \succ 0, \begin{bmatrix} Y_1 & Z_1 \\ Z_1^\mathsf{T} & X_1 \end{bmatrix} \succeq 0 \text{ are feasible} \right\},$$

$$\mathcal{S}_2 := \left\{ x_{\mathcal{C}_2} \middle| J_2(X_{2,2}, X_3, Z_3, J_{22,2}) \succeq 0, X_3 \succ 0, \begin{bmatrix} Y_3 & Z_3 \\ Z_3^\mathsf{T} & X_3 \end{bmatrix} \succeq 0 \text{ are feasible} \right\},$$

and $\mathcal{L}$ is defined by

$$\mathcal{L} := \left\{ y \middle| \hat{J}_{22,1} + \hat{J}_{22,2} = A_{22}X_2 - B_2Z_2 + (A_{22}X_2 - B_2Z_2)^\mathsf{T} + M_2M_2^\mathsf{T}, \right.$$
$$\left. X_2 \succ 0, \begin{bmatrix} Y_2 & Z_2 \\ Z_2^\mathsf{T} & X_2 \end{bmatrix} \succeq 0 \text{ are feasible} \right\}.$$

This allows us to rewrite (6.11) as an optimization problem in the form of (6.12):

$$\begin{aligned} \text{minimize} \quad & \sum_{k=1}^{2} f_k(x_{\mathcal{C}_k}) + g(y) \\ \text{subject to} \quad & \text{(6.14) holds,} \end{aligned} \tag{6.16}$$

where functions $f_1(x_{\mathcal{C}_1}), f_2(x_{\mathcal{C}_2})$ based on each clique are defined as

$$f_1(x_{\mathcal{C}_1}) := \text{Trace}\,(Q_1X_1 + R_1Y_1) + \mathbb{I}_{\mathcal{S}_1}(x_{\mathcal{C}_1}), \tag{6.17a}$$

$$f_2(x_{\mathcal{C}_2}) := \text{Trace}\,(Q_3X_3 + R_3Y_3) + \mathbb{I}_{\mathcal{S}_2}(x_{\mathcal{C}_2}), \tag{6.17b}$$

and $g(y)$ based on the overlapping node 2 is defined as

$$g(y) := \text{Trace}\,(Q_2X_2 + R_2Y_2) + \mathbb{I}_{\mathcal{L}}(y). \tag{6.18}$$

Upon denoting $\hat{x}_{\mathcal{C}_k}$ as the variables in $x_{\mathcal{C}_k}$ that appears in the consensus constraint (6.14), and $y_l(\mathcal{C}_k)$ as the corresponding local copies, *e.g.*, $\hat{x}_{\mathcal{C}_1} = \{X_{2,1}, J_{22,1}\}, y(\mathcal{C}_1) = \{X_2, \hat{J}_{22,1}\}$, the ADMM algorithm for (6.16) takes the following distributed form:

ADMM algorithm for the distributed design

1. x-update: for each clique k, solve the local problem:

$$x_{\mathcal{C}_k}^{(h+1)} = \arg\min_{x_{\mathcal{C}_k}} f_k(x_{\mathcal{C}_k}) + \frac{\rho}{2}\|\hat{x}_{\mathcal{C}_k} - y^{(h)}(\mathcal{C}_k) + \lambda_{\mathcal{C}_k}^{(h)}\|^2. \tag{6.19}$$

2. y-update: solve the following problem to update local variables

$$y^{(h+1)} = \arg\min_{y} g(y) + \frac{\rho}{2}\sum_{k=1}^{2}\|\hat{x}_{\mathcal{C}_k}^{(h+1)} - y(\mathcal{C}_k) + \lambda_{\mathcal{C}_k}^{(h)}\|^2. \tag{6.20}$$

3. λ-update: Update the dual variable

$$\lambda_{\mathcal{C}_k}^{(h+1)} = \lambda_{\mathcal{C}_k}^{(h)} + \hat{x}_{\mathcal{C}_k}^{(h+1)} - y^{(h+1)}(\mathcal{C}_k), k = 1, 2. \tag{6.21}$$

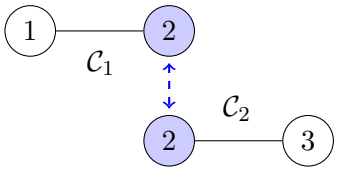


Figure 6.1: Illustration of the ADMM algorithm for solving (6.16): cliques $\mathcal{C}_1 = \{1, 2\}$ and $\mathcal{C}_2 = \{2, 3\}$ can serve as two computing agents and the overlapping node 2 plays a role of coordination by updating the axillary variables.

At each iteration h, subproblem (6.19) only depends on each clique $\mathcal{C}_k$. Consequently, the cliques can serve as computing agents to solve subproblem (6.19) in parallel. For example, clique $\mathcal{C}_1$ needs to solve the following convex problem

$$\begin{aligned} \underset{x_{\mathcal{C}_1}}{\text{minimize}} \;\; & \text{Trace}\,(Q_1 X_1 + R_1 Y_1) + \frac{\rho}{2}\|\hat{x}_{\mathcal{C}_1} - y^{(h)}(\mathcal{C}_1) + \lambda_{\mathcal{C}_1}^{(h)}\|^2 \\ \text{subject to} \;\; & \begin{bmatrix} J_{11} & A_{12}X_{2,1} + X_1 A_{21}^{\mathsf{T}} \\ * & J_{22,1} \end{bmatrix} \succeq 0, \\ & \begin{bmatrix} Y_1 & Z_1 \\ Z_1^{\mathsf{T}} & X_1 \end{bmatrix} \succeq 0, X_1 \succ 0. \end{aligned}$$

The subproblems (6.20) and (6.21) can be solved by node 2. Figure 6.1 illustrates the distributed nature of this algorithm.

Remark 6.6 (Privacy of model data)**.** At each iteration, the coordinator (*i.e.*, node 2) only requires model data of itself A_{22}, B_2, M_2 and the local copies $X_{2,k}^{(h+1)}, J_{22,k}^{(h+1)}$ from clique $\mathcal{C}_k, k = 1, 2$ to solve (6.20) and (6.21). Therefore, the proposed ADMM algorithm for solving (6.11) has a distributed nature (see Figure 6.1 for illustration): cliques $\mathcal{C}_1$ and $\mathcal{C}_2$ can solve (6.19) based on *the model data within each clique* in parallel, and node 2 plays a role of coordination by updating the auxiliary variables and dual variables. Consequently, the model data of node 1 (*i.e.*, $A_{11}, B_1, M_1, A_{12}, A_{21}$) are *private* to clique $\mathcal{C}_1$ only, while clique $\mathcal{C}_2$ holds the model data of node 3 (*i.e.*, $A_{33}, B_3, M_3, A_{32}, A_{23}$), *exclusively*.

Remark 6.7 (Privacy and maximal cliques)**.** In our ADMM algorithm, the privacy of model data are maintained within each maximal clique of $\mathcal{G}_u$. Therefore, the level of privacy depends on the sparsity of $\mathcal{G}_u$. For highly interconnected systems with only one maximal clique, the decomposition (6.10) brings no benefit for privacy, and a global model is still required. In practice, if the plant graph $\mathcal{G}_p$ is a chain or star graph, and the model data privacy can be therefore maintained to a large extent.

Remark 6.8 (Convergence of the ADMM algorithm)**.** Note that the general ADMM algorithm is guaranteed to converge for convex problems under very mild conditions [72, Section 3.2]. For our application, under the feasibility assumption of (6.8), the proposed

ADMM algorithm (6.19)-(6.21) is guaranteed to find a solution as $h \to \infty$. In practical examples, ADMM typically found a solution with moderate accuracy (in the sense of normal stopping criteria [72, Section 3.3]) within a few hundred iterations (see Section 6.5).

6.4.2 The general case

The idea above can be extended to solve (6.11) with a general chordal graph pattern. First, we define a set $\mathcal{N}_0 := \{i \in \mathcal{V} \mid \exists q, k = 1, \ldots, p, \text{such that } i \in \mathcal{C}_q \cap \mathcal{C}_k\}$ that contains the overlapping nodes, and a set $\mathcal{E}_0 := \{(i,j) \in \mathcal{E}_u \mid \exists q, k = 1, \ldots, p, \text{such that } (i,j) \in (\mathcal{C}_q \times \mathcal{C}_q) \cap (\mathcal{C}_k \times \mathcal{C}_k)\}$ that contains the overlapping edges. For the example in Figure 6.1, we have $\mathcal{N}_0 = \{2\}$ and $\mathcal{E}_0 = \emptyset$. Also, we define $\mathcal{N}_i := \{k \mid i \in \mathcal{C}_k, k = 1, \ldots, p\}$, and $\mathcal{E}_{ij} := \{k \mid (i,j) \in \mathcal{C}_k \times \mathcal{C}_k, k = 1, \ldots, p\}$.

In fact, the elements in $\mathcal{N}_0$ and $\mathcal{E}_0$ make the constraint (6.10) coupled among different maximal cliques. Similar to (6.14), for each overlapping node $i \in \mathcal{N}_0$, we introduce local consensus constraints

$$X_i = X_{i,k}, \hat{J}_{ii,k} = J_{ii,k}, \forall k \in \mathcal{N}_i. \tag{6.22}$$

For each overlapping edge $(i,j) \in \mathcal{E}_0$, we introduce local consensus constraints

$$X_{ij} = X_{i,k}, \hat{J}_{ij,k} = J_{ij,k}, \forall (i,j) \in \mathcal{E}_{ij}. \tag{6.23}$$

Then, the variable $x_{\mathcal{C}_k}$ for each maximal clique $k = 1, \ldots, p$ includes

- $X_i, Y_i, Z_i, i \in \mathcal{C}_k \setminus \mathcal{N}_0$ that belongs to clique $\mathcal{C}_k$ exclusively;
- $X_{i,k}, J_{ii,k}, i \in \mathcal{C}_k \cap \mathcal{N}_0$ that corresponds to overlapping nodes in $\mathcal{C}_k$;
- $J_{ij,k}, (i,j) \in (\mathcal{C}_k \times \mathcal{C}_k) \cap \mathcal{E}_0$ that corresponds to overlapping edges in $\mathcal{C}_k$;

We also collect the local copies $X_i, Y_i, Z_i, \hat{J}_{ii,k}, i \in \mathcal{N}_0$ and $\hat{J}_{ij,k}, X_{ij,k}, (i,j) \in \mathcal{E}_0$ as the consensus variable y.

Then, (6.11) can be written into the canonical ADMM form:

$$\begin{aligned} \text{minimize} \quad & \sum_{k=1}^{p} f_k(x_{\mathcal{C}_k}) + g(y) \\ \text{subject to} \quad & (6.22) \text{ and } (6.23) \text{ hold}, \end{aligned} \tag{6.24}$$

where the clique function $f_k(x_{\mathcal{C}_k})$ is defined as

$$f_k(x_{\mathcal{C}_k}) := \sum_{i \in \mathcal{C}_k \setminus \mathcal{N}_0} \text{Trace}\,(Q_i X_i + R_i Y_i) + \mathbb{I}_{\mathcal{S}_k}(x_{\mathcal{C}_k}), \tag{6.25}$$

and $g(y)$ is defined as

$$g(y) := \sum_{i \in \mathcal{N}_0} \text{Trace}\,(Q_i X_i + R_i Y_i) + \mathbb{I}_{\mathcal{L}}(y). \tag{6.26}$$

In (6.25), set $\mathcal{S}_k$ is defined as

$$\mathcal{S}_k := \left\{ x_{\mathcal{C}_k} \middle| J_k(x_{\mathcal{C}_k}) \succeq 0, X_i \succ 0, \begin{bmatrix} Y_i & Z_i \\ Z_i^{\mathsf{T}} & X_i \end{bmatrix} \succeq 0, i \in \mathcal{C}_k \setminus \mathcal{N}_0 \text{ are feasible} \right\},$$

and in (6.26), set $\mathcal{L}$ is defined as

$$\begin{aligned} \mathcal{L} := \Big\{ y \Big| \sum_{k \in \mathcal{N}_i} \hat{J}_{ii,k} = A_{ii}X_i - B_iZ_i + (A_{ii}X_i - B_iZ_i)^{\mathsf{T}} + M_iM_i^{\mathsf{T}}, X_i \succ 0, \\ \begin{bmatrix} Y_i & Z_i \\ Z_i^{\mathsf{T}} & X_i \end{bmatrix} \succeq 0, i \in \mathcal{N}_0, \sum_{k \in \mathcal{E}_{ij}} \hat{J}_{ij,k} = A_{ij}X_{ij} + X_{ji}A_{ji}^{\mathsf{T}}, \\ (i,j) \in \mathcal{E}_0 \text{ are feasible} \Big\}. \end{aligned}$$

By applying the ADMM to (6.24), we obtain iterations that are identical to (6.19)-(6.21). Note that the set $\mathcal{L}$ can be equivalently rewritten as a product of sets defined by $X_i, Y_i, Z_i, \hat{J}_{ii,k}, i \in \mathcal{N}_0$ and $\hat{J}_{ij,k}, X_{ij}, (i,j) \in \mathcal{E}_0$. For each $i \in \mathcal{N}_0$, the set for $X_i, Y_i, Z_i, \hat{J}_{ii,k}$ is defined as

$$\begin{aligned} \mathcal{L}_i := \Big\{ (X_i, Y_i, Z_i, \hat{J}_{ii,k}) \Big| \sum_{k \in \mathcal{N}_i} \hat{J}_{ii,k} = A_{ii}X_i - B_iZ_i + \\ (A_{ii}X_i - B_iZ_i)^{\mathsf{T}} + M_iM_i^{\mathsf{T}}, X_i \succ 0, \begin{bmatrix} Y_i & Z_i \\ Z_i^{\mathsf{T}} & X_i \end{bmatrix} \succeq 0 \Big\}. \end{aligned}$$

This means that y-update (6.20) can be distributed among the overlapping nodes $\mathcal{N}_0$ and overlapping edges $\mathcal{E}_0$. Therefore, similar to the example in Section 6.4.1, variables $x_{\mathcal{C}_k}^h$ can be updated on each clique in parallel, and the overlapping elements in $\mathcal{N}_0$ and $\mathcal{E}_0$ can update $y_{\mathcal{C}_k}^h, \lambda_{\mathcal{C}_k}^h$ individually until convergence.

Here, as stated in Remark 6.6, we emphasize that the main interest of our algorithm is the ability of distributing the computation to cliques and overlapping elements, thus preserving the privacy of model data in the problem.

6.5 Numerical examples

Three numerical examples are used to demonstrate the effectiveness of the proposed distributed design method[1]. We ran the ADMM algorithm with termination tolerance 10^{-3}. In our simulations, SeDuMi [71] was used to solve the subproblems within each clique and overlapping elements.

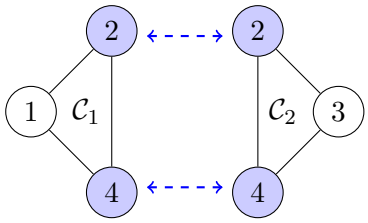


Figure 6.2: Illustration of the ADMM algorithm for solving (6.8) corresponding to the example (6.27): the cliques $\mathcal{C}_1 = \{1, 2, 4\}$ and $\mathcal{C}_2 = \{2, 3, 4\}$ can serve as two computing agents and the overlapping nodes play a role of coordination by updating the axillary variables.

6.5.1 First-order systems with acyclic directed graphs

As our first numerical example, we consider a network of four coupled unstable first-order subsystems, where the plant graph $\mathcal{G}_p$ is a directed acyclic graph, and the global dynamics are

$$\dot{x}(t) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 0 & 2 & 3 & 4 \\ 1 & 2 & 0 & 4 \end{bmatrix} x(t) + u(t) + d(t). \tag{6.27}$$

We chose $Q_i = 1$ and $R_i = 1, i \in \mathcal{V}$ in our simulation. When the global dynamics are available, solving (6.8) directly returned a decentralized controller $K_{11} = 7.34; K_{22} = 11.38; K_{33} = 6.16, K_{44} = 13.48$ with an $\mathcal{H}_2$ performance of 5.36.

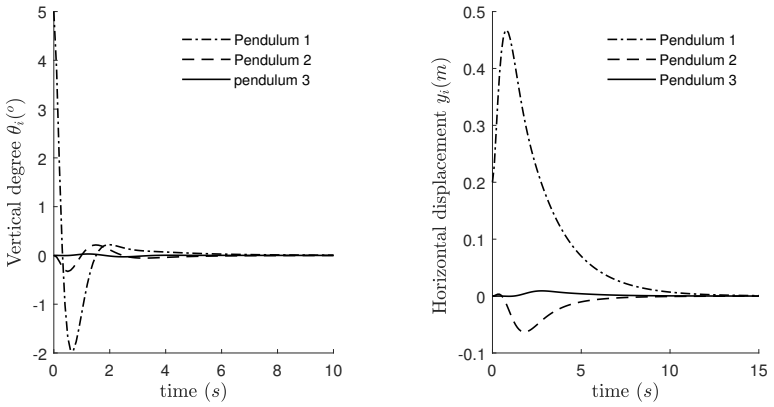
Instead, when the privacy of model data is concerned, the proposed ADMM algorithm can solve (6.8) in a distributed fashion. As shown in Figure 6.2, for clique 1, only the model data of nodes 1, 2, 4 are required, while clique 2 only needs the model data of nodes 2, 3, 4, and the overlapping nodes 2 and 4 play a role of coordinations in the algorithm. In this way, the model of node 1 can be kept private within clique 1 and the model of node 3 is known within clique 2 exclusively. For this instance, after 54 iterations, the ADMM algorithm returned the following decentralized controller $K_{11} = 7.35; K_{22} = 11.41; K_{33} = 6.16, K_{44} = 13.49$ with an $\mathcal{H}_2$ performance of 5.37. This controller is almost identical to that obtained by solving (6.8) directly in a centralized fashion, and the minor difference is due to the stopping criterion of moderate accuracy in the ADMM algorithm.

6.5.2 Coupled inverted pendula

We first consider a network of three coupled inverted pendula, as in Chapter 5 (see Figure 5.5). The linearized dynamics around the upright equilibrium point of each pendulum are

$$A_i = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{m_i+m}{m_i l} g & 0 & \frac{k_i}{m_i l} & \frac{c_i+b_i}{m_i l} \\ 0 & 0 & 0 & 1 \\ -\frac{m}{m_i} g & 0 & -\frac{k_i}{m_i} & -\frac{c_i+b_i}{m_i} \end{bmatrix}, \; A_{ij} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{k_{ij}}{m_i l} & \frac{b_{ij}}{m_i l} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{k_{ij}}{m_i} & \frac{b_{ij}}{m_i} \end{bmatrix}, \; B_i = \begin{bmatrix} 0 \\ -\frac{1}{m_i l} \\ 0 \\ \frac{1}{m_i} \end{bmatrix},$$

[1] Code is available via https://github.com/zhengy09/distributed_design_methods



(a) (b)

Figure 6.3: Response of the closed-loop inverted pendula using the decentralized controller computed by the ADMM algorithm: (a) vertical angle θ_i of each pendulum; (b) horizontal displacement y_i of each pendulum.

for $i = 1, 2, 3; (i, j) = (1, 2), (2, 1), (2, 3), (3, 2)$, where $k_i = \sum_{j \in \mathbb{N}_i} k_{ij}, b_i = \sum_{j \in \mathbb{N}_i} b_{ij}$. The local state variable is $x_i = [\theta_i, \dot{\theta}_i, y_i, \dot{y}_i]^\mathsf{T}$, and $c_i, b_{ij} = b_{ji}, k_{ij} = k_{ji}$ are friction, damper and spring coefficients, respectively. The plant graph for this example is a chain of three nodes, and its chordal decomposition is shown in Figure 6.1. The parameters in the simulation were $m_1 = 0.6, m_2 = 0.8, m_3 = 1, m = 0.1, g = 10, l = 0.5, k_{12} = k_{21} = 0.2, k_{23} = k_{32} = 0.4, b_{12} = b_{21} = 0.4, b_{23} = b_{32} = 0.2, c_1 = 1, c_2 = 0.2$ and $c_3 = 1$. We aim to compute a decentralized controller (6.3) by solving (6.5) with $Q_i = I, R_i = I, i = 1, 2, 3$.

Using the proposed ADMM algorithm, we solved this problem in a distributed fashion: for clique 1, only the models of pendulum 1 and pendulum 2 are required, while clique 2 only needs the model data of pendulum 2 and pendulum 3. In the simulation, the algorithm returned a decentralized controller after 418 iterations with an $\mathcal{H}_2$ performance of 6.65 for the closed-loop system. Figure 6.3 shows the closed-loop response for the initial condition $x_1(0) = [0.172, 0, 0.2, 0]$, $x_2(0) = 0$ and $x_3(0) = 0$.

6.5.3 A chain of unstable second-order coupled systems

Next, we use a chain of five nodes (see Figure 6.4) to provide a comparison between the proposed ADMM algorithm and the following three approaches:

1. *A sequential approach* (see Chapter 5), which exploits the properties of clique trees in chordal graphs;
2. *Localized LQR design* [118, Section 7.3], which computes a local LQR controller for each subsystem independently by ignoring the coupling terms A_{ij};
3. *Truncated LQR design*, which computes a centralized LQR controller using the global model data and only keeps the diagonal blocks for decentralized feedback.

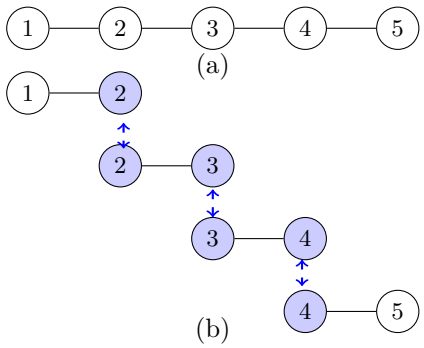


Figure 6.4: (a) A chain of five nodes, where each subsystem is a second-order unstable subsystem coupled with its neighbours, as shown in (6.28); (b) Four maximal cliques in this system $\mathcal{C}_i = \{i, i+1\}, i = 1, 2, 3, 4$, which serve as four computing agents relying only on the model data within each clique; the overlapping nodes 2, 3, 4 play a role of coordination.

Table 6.1: Comparison of the proposed ADMM algorithm, sequential approach [41], localized LQR and truncated LQR design for system (6.28).

	ADMM	Sequential$^\diamond$	Localized LQR	Truncated LQR
pct.‡	100 %	72 %	54 %	62 %
$\mathcal{H}_2{}^\dagger$	6.06	6.36	6.50	6.49

‡: Successful percentage of returning a stabilizing decentralized controller.
†: Average $\mathcal{H}_2$ performance of the closed-loop system based on common successful instances.
$^\diamond$: This refers to the sequential approach in Section 5.4.3.

It is assumed that each node is an unstable second order system coupled with its neighbouring nodes,

$$\dot{x}_i = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} x_i + \sum_{j \in \mathbb{N}_i} A_{ij} x_j + \begin{bmatrix} 0 \\ 1 \end{bmatrix} (u_i + d_i), \tag{6.28}$$

where the entries of coupling term A_{ij} were generated randomly from -0.5 to 0.5. In this example, there are four maximal cliques $\mathcal{C}_i = \{i, i+1\}, i = 1, 2, 3, 4$. The model data can be kept private within each clique, and the overlapping nodes (*i.e.*, $2, 3, 4$) coordinate the consensus variables among maximal cliques. In the simulation, the state and control weights were $Q_i = I$ and $R_i = I$ for each subsystem.

We generated 100 random instances of this interconnected system (6.28). The performance comparison between the four methods is listed in Table 6.1. The proposed ADMM algorithm was able to return stabilizing decentralized controllers for all 100 tests, while the sequential approach, localized LQR and truncated LQR design only succeeded for 72 %, 54 %, 62 % of the tests, respectively. This is expected since the conservatism of the ADMM algorithm only comes from the block-diagonal assumption that leads to a convex problem. The sequential approach requires an additional equal-splitting assumption among maximal cliques, and the applicability of localized LQR and truncated LQR design depends on the coupling strength among subsystems. Also, the average $\mathcal{H}_2$

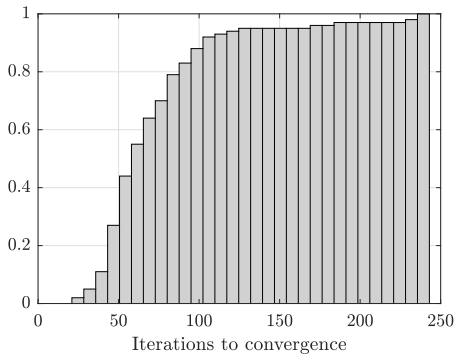


Figure 6.5: Cumulative plot of the fraction of 100 random trails of (6.28) that required a given number of iterations to converge.

performance for the common succeeded instances by the ADMM algorithm is the lowest. Finally, Figure 6.5 shows the cumulative plot of convergence performance of the ADMM algorithm, where 90 % of the tests required less than 150 iterations.

6.6 Conclusion

In this chapter, we have introduced a distributed design method for optimal decentralized control that relies on local model information only. Our main strategy is consistent with the recent general idea of exploiting sparsity in systems theory via chordal decomposition [13, 34, 39, 41]. In this chapter, we have further demonstrated the potential of chordal decomposition in the distributed design of decentralized controllers, by combining it with the ADMM algorithm.

Part III

Large-scale Sum-of-squares (SOS) Programs

7

Partial orthogonality in general SOS programs

Part I and Part II of this thesis have demonstrated the high potential of chordal sparsity in improving the scalability of solving SDPs, including some LMIs arising in analysis and synthesis of large-scale networked systems. As shown in Example 2.3, polynomial sum-of-squares (SOS) constraints can be reformulated into certain LMIs as well.

This part of the thesis, Chapters 7 — 9, focuses on exploiting sparsity in optimization problems with SOS constraints, called *SOS programs*, to facilitate the solution scalability. In particular, Chapter 7 reveals an inherent structural property, called *partial orthogonality*, that naturally arises in the SDPs when recasting general SOS programs using the standard monomial basis, and presents a fast ADMM algorithm that exploits partial orthogonality. In Chapter 8, we introduce an extension of the chordal decomposition Theorems 2.10 and 2.13 to sparse polynomial matrices. Finally, Chapter 9 reveals the relationship between chordal decomposition in SOS programs with two recent scalable techniques – DSOS/SDSOS [124].

7.1 Introduction

Optimizing the coefficients of a polynomial in n variables, subject to a nonnegativity constraint on the entire space $\mathbb{R}^n$ or on a semialgebraic set $\mathcal{S} \subseteq \mathbb{R}^n$ (*i.e.*, a set defined by a finite number of polynomial equations and inequalities), is a fundamental problem in many fields. For instance, linear, quadratic and mixed-integer optimization problems can be recast as polynomial optimization problems (POPs) of the form [125]

$$\underset{x \in \mathcal{S}}{\text{minimize}}\ p(x), \tag{7.1}$$

where $p(x)$ is a multivariate polynomial and $\mathcal{S} \subseteq \mathbb{R}^n$ is a semialgebraic set. Problem (7.1) is clearly equivalent to

$$\begin{aligned} &\text{maximize} && \gamma \\ &\text{subject to} && p(x) - \gamma \geq 0 \quad \forall x \in \mathcal{S}, \end{aligned} \tag{7.2}$$

so POPs of the form (7.1) can be solved globally if a linear cost function can be optimized subject to polynomial nonnegativity constraints on a semialgebraic set.

Another important example is the construction of a Lyapunov function $V(x)$ to certify that an equilibrium point x^* of a dynamical system $\frac{\mathrm{d}x(t)}{\mathrm{d}t} = f(x(t))$ is locally stable. Taking $x^* = 0$ without loss of generality, given a neighbourhood $\mathcal{D}$ of the origin, local stability follows if $V(0) = 0$ and

$$V(x) > 0, \quad \forall x \in \mathcal{D} \setminus \{0\}, \tag{7.3a}$$

$$-f(x)^{\mathsf{T}} \nabla V(x) \geq 0, \quad \forall x \in \mathcal{D}. \tag{7.3b}$$

Often, the vector field $f(x)$ is polynomial [126] and, if one restricts the search to polynomial Lyapunov functions $V(x)$, conditions (7.3a)-(7.3b) amount to a feasibility problem over nonnegative polynomials.

Testing for nonnegativity, however, is NP-hard for polynomials of degree as low as four [52]. This difficulty is often resolved by requiring that the polynomials under consideration are a sum of squares (SOS) of polynomials of lower degree. In fact, checking for the existence (or lack) of an SOS representation amounts to solving a semidefinite program (SDP) [52]. In particular, consider a polynomial of degree $2d$ in n variables,

$$p(x) = \sum_{\alpha \in \mathbb{N}^n, |\alpha| \leq 2d} p_\alpha x_1^{\alpha_1} \dots x_n^{\alpha_n}.$$

The key observation in [52] is that an SOS representation of $p(x)$ exists if and only if there exists a positive semidefinite matrix X such that

$$p(x) = v_d(x)^{\mathsf{T}} X v_d(x), \tag{7.4}$$

where

$$v_d(x) = [1, x_1, x_2, \dots, x_n, x_1^2, x_1 x_2, \dots, x_n^d]^{\mathsf{T}} \tag{7.5}$$

is the vector of monomials of degree no larger than d. Upon equating coefficients on both sides of (7.4), testing if $p(x)$ is an SOS reduces to a feasibility SDP of the form

$$\begin{aligned} \text{find} \quad & X \\ \text{subject to} \quad & \langle B_\alpha, X \rangle = p_\alpha, \quad \alpha \in \mathbb{N}^n_{2d}, \\ & X \succeq 0, \end{aligned} \tag{7.6}$$

where $\mathbb{N}^n_{2d}$ is the set of n-dimensional multi-indices with length at most $2d$, B_α are known symmetric matrices indexed by such multi-indices (see Section 7.2 for more details), and $\langle A, B \rangle = \operatorname{trace}(AB)$ is the standard Frobenius inner product of two symmetric matrices A and B.

Despite the tremendous impact of SOS techniques in the fields of polynomial optimization [127] and systems analysis [9], the current poor scalability of second-order interior-point algorithms for semidefinite programming prevents the use of SOS methods to solve POPs with many variables, or to analyse dynamical systems with many states. The main issue is that, when the full monomial basis (7.5) is used, the linear dimension of the matrix X and the number of constraints in (7.6) are $N = \binom{n+d}{d}$ and $m = \binom{n+2d}{2d}$, respectively, both of which grow quickly as a function of n and d.

7.1.1 Related work

One strategy to mitigate the computational cost of optimization problems with SOS constraints (hereafter called *SOS programs*) is to replace the SDP obtained from the basic formulation outlined above with one that is less expensive to solve using second-order interior-point algorithms. Facial reduction techniques [128], including the Newton polytope [129] and diagonal inconsistency [130], and symmetry reduction strategies [131] can be utilised to eliminate unnecessary monomials in the basis $v_d(x)$, thereby reducing the size of the positive semidefinite (PSD) matrix variable X. Correlative sparsity [31] can also be exploited to construct sparse SOS representations, wherein a polynomial $p(x)$ is written as a sum of SOS polynomials, each of which depends only on a subset of the entries of x. This enables one to replace the large PSD matrix variable X with a set of smaller PSD matrices, which can be handled more efficiently. Further computational gains are achievable if one replaces any PSD constraints—either the original condition $X \succeq 0$ in (7.6) or the PSD constraints obtained after applying the aforemention techniques—with the stronger constraints the PSD matrices are diagonally or scaled-diagonally dominant [124]. These conditions can be imposed with linear and second-order cone programming, respectively, and are therefore less computationally expensive. However, while the conservativeness introduced by the requirement of diagonal dominance can be reduced with a basis pursuit algorithm [132], it cannot generally be removed.

Another strategy to enable the solution of large SOS programs is to replace the computationally demanding interior-point algorithms with first-order methods, at the expense of reducing the accuracy of the solution. The design of efficient first-order algorithms for large-scale SDPs has recently received increasing attention: Wen *et al.* proposed an alternating-direction augmented-Lagrangian method for large-scale dual SDPs [62]; O'Donoghue *et al.* developed an operator-splitting method to solve the homogeneous self-dual embedding of conic programs [64], which has recently been extended by the authors to exploit aggregate sparsity via chordal decomposition [36–38]. Algorithms that specialize in SDPs from SOS programming exist [133, 134], but can be applied only to unconstrained POPs—not to constrained POPs of the form (7.2), nor to the Lyapunov

conditions (7.3a)-(7.3b). First-order regularization methods have also been applied to large-scale constrained POPs, but without taking into account any problem structure [135]. Finally, the sparsity of the matrices B_α in (7.6) was exploited in [48] to design an operator-splitting algorithm that can solve general large-scale SOS programs, but fails to detect infeasibility (however, recent developments [75, 76] may offer a solution for this issue).

7.1.2 Main contributions

One major shortcoming of all but the last of these recent approaches is that they can only be applied to particular classes of SOS programs. For this reason, in this chapter we develop a fast first-order algorithm, based on the alternating direction method of multipliers, for the solution of generic large-scale SOS programs. Our algorithm exploits a particular structural property of SOS programs and can also detect infeasibility. Precisely, our contributions are:

1. At the modeling level, we highlight a structural property of SDPs derived from SOS programs using the standard monomial basis: the equality constraints are *partially orthogonal*. Notably, the SDPs formulated by common SOS modeling toolboxes [79, 80, 136] possess this property.
2. At the computational level, we show how partial orthogonality leads to a "diagonal plus low rank" matrix structure in the ADMM algorithm of [64], so the matrix inversion lemma can be applied to reduce its computational cost. Precisely, a system of $m \times m$ linear equations to be solved at each iteration can be replaced with a $t \times t$ system, often with $t \ll m$.
3. We demonstrate the efficiency of our method—available as a new package in the MATLAB solver CDCS [35]— compared to many common interior-point solvers (SeDuMi [71], SDPT3 [137], SDPA [112], CSDP [138], Mosek [139]) and to the first-order solver SCS [65]. Our results on large-scale SOS programs from constrained POPs and Lyapunov stability analysis of nonlinear polynomial systems suggest that the proposed algorithm will enlarge the scale of practical problems that can be handled via SOS techniques.

7.1.3 Outline

The rest of this chapter is organized as follows. Section 7.2 briefly reviews SOS programs and their reduction to SDPs. Section 7.3 discusses partial orthogonality in the equality constraints of SDPs arising from SOS programs, while Section 7.4 shows how to exploit it to facilitate the solution of large-scale SDPs using ADMM. Sections 7.5 and 7.6 extend our results to matrix-valued SOS programs and weighted SOS constraints. Numerical experiments are presented in Section 7.7, and Section 7.8 concludes the chapter.

7.2 Preliminaries

The sets of nonnegative integers and real numbers are, respectively, $\mathbb{N}$ and $\mathbb{R}$. For $x \in \mathbb{R}^n$ and $\alpha \in \mathbb{N}^n$, the monomial $x^\alpha = x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}$ has degree $|\alpha| := \sum_{i=1}^n \alpha_i$. Given $d \in \mathbb{N}$, we let $\mathbb{N}^n_d = \{\alpha \in \mathbb{N}^n : |\alpha| \leq d\}$ and $\mathbb{R}[x]_{n,2d}$ be the set of polynomials in n variables with real coefficients of degree $2d$ or less. A polynomial $p(x) \in \mathbb{R}[x]_{n,2d}$ is a sum-of-squares (SOS) if $p(x) = \sum_{i=1}^q [f_i(x)]^2$, for some polynomials $f_i \in \mathbb{R}[x]_{n,d}$, $i = 1, \ldots, q$. We denote by $\Sigma[x]_{n,2d}$ the set of SOS polynomials in $\mathbb{R}[x]_{n,2d}$. Finally, $\mathbb{S}^n_+$ is the cone of $n \times n$ PSD matrices and $I_{r\times r}$ is the $r \times r$ identity matrix.

7.2.1 General SOS programs

Consider a vector of optimization variables $u \in \mathbb{R}^t$, a cost vector $w \in \mathbb{R}^t$, and note that any polynomial $p_j(x) \in \mathbb{R}[x]_{n,2d_j}$ whose coefficients depend affinely on u can be written as $p_j(x) = g_0^j(x) - \sum_{i=1}^t u_i g_i^j(x)$ for a suitable choice of polynomials or monomials $g_0^j, \ldots, g_t^j \in \mathbb{R}[x]_{n,2d_j}$. We consider SOS programs written in the standard form

$$\begin{aligned} \underset{u,\, s_1,\ldots,s_k}{\text{minimize}} \quad & w^\mathsf{T} u \\ \text{subject to} \quad & s_j(x) = g_0^j(x) - \sum_{i=1}^t u_i g_i^j(x) \;\forall j = 1, \ldots, k, \\ & s_j \in \Sigma[x]_{n,2d_j}, \quad j = 1, \ldots, k. \end{aligned} \tag{7.7}$$

Note that any linear optimization problem with polynomial nonnegativity constraints on fixed semialgebraic sets can be relaxed into an SOS program of the form (7.7). For instance, when $\mathcal{S} \equiv \mathbb{R}^n$ problem (7.2) can be relaxed as [52]

$$\begin{aligned} \underset{\gamma, s}{\text{minimize}} \quad & -\gamma \\ \text{subject to} \quad & s(x) = p(x) - \gamma, \\ & s \in \Sigma[x]_{n,2d}. \end{aligned} \tag{7.8}$$

Similarly, the global stability of the origin for a polynomial dynamical system such that $f(0) = 0$ may be established by looking for a polynomial Lyapunov function of the form $V(x) = -\sum_{i=1}^t u_i g_i(x)$, where $g_1(0) = \cdots = g_t(0) = 0$. With $\mathcal{D} \equiv \mathbb{R}^n$, and after subtracting $x^\mathsf{T} x$ from the left-hand side of (7.3a) to ensure strict positivity away from the origin [9], suitable values u_i can be found via the SOS feasibility program

$$\begin{aligned} \text{find} \quad & u,\, s_1,\, s_2 \\ \text{subject to} \quad & s_1(x) = -x^\mathsf{T} x - \sum_{i=1}^t u_i g_i(x), \\ & s_2(x) = \sum_{i=1}^t u_i f(x)^\mathsf{T} \nabla g_i(x), \\ & s_1,\, s_2 \in \Sigma[x]_{n,2d}. \end{aligned} \tag{7.9}$$

It can be checked that SOS programs arising from polynomial nonnegativity constraints over fixed semialgebraic sets, such as Lasserre's relaxations of constrained POPs [127] and SOS relaxations of local Lyapunov inequalities [126, 140], can also be recast as in (7.7) by adding extra polynomials to represent the SOS multipliers introduced after applying the *Positivstellensatz* [126].

To simplify the exposition in the rest of this work, instead of (7.7) we will consider the basic problem

$$\begin{aligned} \underset{u,\, s}{\text{minimize}} \quad & w^{\mathsf{T}} u \\ \text{subject to} \quad & s(x) = g_0(x) - \sum_{i=1}^{t} u_i g_i(x), \\ & s \in \Sigma[x]_{n,2d}. \end{aligned} \tag{7.10}$$

All of our results extend to (7.7) when $k > 1$, as well as to SOS programs with additional linear constraints on u, because each of $s_1, \ldots, s_k$ enters one and only one equality constraint.

7.2.2 SDP formulation

The SOS program (7.10) can be converted into an SDP upon fixing a basis to represent the SOS polynomial variables. The simplest and most common choice to represent a degree-$2d$ SOS polynomial is the basis $v_d(x)$ of monomials of degree no greater than d, defined in (7.5). As discussed in [52] and [141], the polynomial $s(x)$ in (7.10) is SOS if and only if

$$s(x) = v_d(x)^{\mathsf{T}} X v_d(x) = \left\langle X, v_d(x) v_d(x)^{\mathsf{T}} \right\rangle, \; X \succeq 0. \tag{7.11}$$

Let B_α be the 0/1 indicator matrix for the monomial x^α in the outer product matrix $v_d(x) v_d(x)^{\mathsf{T}}$, *i.e.*,

$$(B_\alpha)_{\beta,\gamma} = \begin{cases} 1 & \text{if } \beta + \gamma = \alpha, \\ 0 & \text{otherwise,} \end{cases} \tag{7.12}$$

where the natural ordering of multi-indices $\beta, \gamma \in \mathbb{N}^n_d$ is used to index the entries of B_α. Then,

$$v_d(x) v_d(x)^{\mathsf{T}} = \sum_{\alpha \in \mathbb{N}^n_{2d}} B_\alpha x^\alpha. \tag{7.13}$$

Upon writing $g_i(x) = \sum_{\alpha \in \mathbb{N}^n_{2d}} g_{i,\alpha} x^\alpha$ for each $i = 0, 1, \ldots, t$, and representing $s(x)$ as in (7.11), the equality constraint in (7.10) becomes

$$\begin{aligned} \sum_{\alpha \in \mathbb{N}^n_{2d}} \left(g_{0,\alpha} - \sum_{i=1}^{t} u_i g_{i,\alpha} \right) x^\alpha &= \left\langle X, v_d(x) v_d(x)^{\mathsf{T}} \right\rangle \\ &= \sum_{\alpha \in \mathbb{N}^n_{2d}} \langle B_\alpha, X \rangle \, x^\alpha. \end{aligned} \tag{7.14}$$

Matching the coefficients on both sides yields

$$g_{0,\alpha} - \sum_{i=1}^{t} u_i g_{i,\alpha} = \langle B_\alpha, X \rangle, \quad \forall \alpha \in \mathbb{N}^n_{2d}. \tag{7.15}$$

We refer to (7.15) as the *coefficient matching conditions* [48]. The SOS program (7.10) is then equivalent to the SDP

$$\begin{aligned} \underset{u;X}{\text{minimize}} \;\; & w^\mathsf{T} u \\ \text{subject to} \;\; & \langle B_\alpha, X \rangle + \sum_{i=1}^{t} u_i g_{i,\alpha} = g_{0,\alpha} \;\; \forall \alpha \in \mathbb{N}^n_{2d}, \\ & X \succeq 0. \end{aligned} \tag{7.16}$$

As already mentioned in Section 7.1, when the full monomial basis $v_d(x)$ is used to formulate the SDP (7.16), the size of X and the number of constraints are, respectively, $N = \binom{n+d}{d}$ and $m = \binom{n+2d}{2d}$. The size of SDP (7.16) may be reduced (often significantly) by eliminating redundant monomials in $v_d(x)$ based on the structure of the polynomials $g_0(x), \ldots, g_t(x)$; the interested reader is referred to Refs. [128–131].

7.3 Partial orthogonality in SOS programs

For simplicity, we re-index the coefficient matching conditions (7.15) using integers $i = 1, \ldots, m$ instead of the multi-indices α. Let $\mathrm{vec} : \mathbb{S}^N \to \mathbb{R}^{N^2}$ map a matrix to the stack of its columns and define $A_1 \in \mathbb{R}^{m \times t}$ and $A_2 \in \mathbb{R}^{m \times N^2}$ as

$$A_1 := \begin{bmatrix} g_{1,1} & \cdots & g_{t,1} \\ \vdots & \ddots & \vdots \\ g_{1,m} & \cdots & g_{t,m} \end{bmatrix}, \qquad A_2 := \begin{bmatrix} \mathrm{vec}(B_1)^\mathsf{T} \\ \vdots \\ \mathrm{vec}(B_m)^\mathsf{T} \end{bmatrix}. \tag{7.17}$$

In other words, A_1 collects the coefficients of polynomials $g_i(x)$ column-wise, and A_2 lists the vectorized matrices B_α (after re-indexing) in a row-wise fashion. Finally, let $\mathcal{S}_+$ be the vectorized positive semidefinite cone, such that $\mathrm{vec}(X) \in \mathcal{S}_+$ if and only if $X \succeq 0$, and define

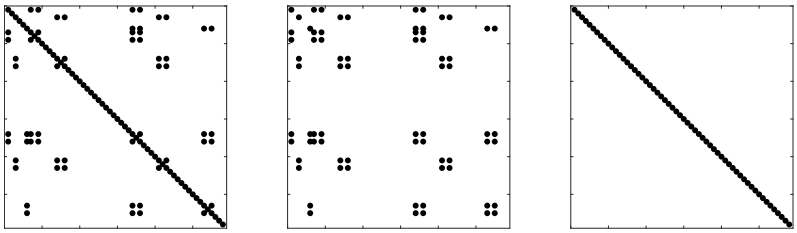
$$A := [A_1,\ A_2] \in \mathbb{R}^{m \times (t+N^2)}, \tag{7.18a}$$

$$b := [g_{0,1},\ \ldots,\ g_{0,m}]^\mathsf{T} \in \mathbb{R}^m, \tag{7.18b}$$

$$c := \left[w^\mathsf{T},\ 0,\ \ldots,\ 0\right]^\mathsf{T} \in \mathbb{R}^{t+N^2}, \tag{7.18c}$$

$$\xi := \left[u^\mathsf{T},\ \mathrm{vec}(X)^\mathsf{T}\right]^\mathsf{T} \in \mathbb{R}^{t+N^2}, \tag{7.18d}$$

$$\mathcal{K} := \mathbb{R}^t \times \mathcal{S}_+ \,. \tag{7.18e}$$



(a) (b) (c)

Figure 7.1: Sparsity patterns for (a) AA^T, (b) $A_1A_1^\mathsf{T}$, and (c) $A_2A_2^\mathsf{T}$ for problem `sosdemo2` in SOSTOOLS [80].

Then, noticing from the definition of the trace inner product of matrices that $\langle B_m, X\rangle = \mathrm{vec}(B_m)^\mathsf{T}\mathrm{vec}(X)$, we can rewrite (7.16) as the primal-form conic program

$$\begin{aligned} \underset{\xi}{\text{minimize}} \quad & c^\mathsf{T}\xi \\ \text{subject to} \quad & A\xi = b, \\ & \xi \in \mathcal{K}. \end{aligned} \tag{7.19}$$

The key observation at this stage is that the rows of the constraint matrix A are *partially orthogonal*. We show this next, assuming without loss of generality that $t < m$. In fact, we often have $t \ll m$ in practice (cf. Tables 7.1 and 7.3 in Section 7.7).

Proposition 7.1. Let $A = [A_1,\ A_2]$ be the constraint matrix in the conic formulation (7.17) of a SOS program modeled using the monomial basis. The $m \times m$ matrix AA^T is of the "diagonal plus low rank" form. Precisely, $D := A_2A_2^\mathsf{T}$ is diagonal and $AA^\mathsf{T} = D + A_1A_1^\mathsf{T}$.

Proof. The definition of A implies $AA^\mathsf{T} = A_1A_1^\mathsf{T} + A_2A_2^\mathsf{T}$, so we need to show that $A_2A_2^\mathsf{T}$ is diagonal. This follows from the definition (7.12) of the matrices B_α: if an entry of B_α is nonzero, the same entry in B_β, $\alpha \neq \beta$, must be zero. Upon re-indexing the matrices using integers $i = 1, \ldots, m$ as explained above and letting n_i be the number of nonzero entries in B_i, it is clear that $\mathrm{vec}(B_i)^\mathsf{T}\mathrm{vec}(B_j) = n_i$ if $i = j$, and zero otherwise. Thus, $A_2A_2^\mathsf{T} = \mathrm{diag}(n_1, \ldots, n_m)$. ∎

In essence, Proposition 7.1 states that the constraint sub-matrices corresponding to the matrix X in the SOS decomposition (7.11) are orthogonal. This fact is a basic structural property for *any* SOS program formulated using the usual monomial basis. It is not difficult to check that Proposition 7.1 also holds when the full monomial basis $v_d(x)$ is reduced using any of the techniques implemented in any of the modeling toolboxes [79, 80, 136].

Remark 7.2. In general, the product $A_1A_1^\mathsf{T}$ has no particular structure, and AA^T is not diagonal except for very special problem classes. For example, Figure 7.1 illustrates the sparsity pattern of AA^T, $A_1A_1^\mathsf{T}$, and $A_2A_2^\mathsf{T}$ for `sosdemo2` in SOSTOOLS [80], an SOS

formulation of a Lyapunov function search: $A_2 A_2^\mathsf{T}$ is diagonal, but $A_1 A_1^\mathsf{T}$ and AA^T are not. This makes the algorithms proposed in [133, 134] inapplicable, as they require that AA^T is diagonal.

Remark 7.3. Using the monomial basis to formulate the coefficient matching conditions (7.15) makes the matrix A sparse, because only a small subset of entries of the matrix $v_d(x)v_d(x)^\mathsf{T}$ are equal to a given monomial x^α. In particular, the density of the nonzero entries of A_2 is $\mathcal{O}(n^{-2d})$ [48]. However, the aggregate sparsity pattern of SDP (7.19) is dense, so methods that exploit aggregate sparsity in SDPs [23, 36–38] are not useful for general SOS programs.

7.4 A fast ADMM-based algorithm

Partial orthogonality of the constraint matrix A in conic programs of the form (7.19) allows for the extension of a first-order, ADMM-based method proposed in [64]. To make this chapter self-contained, we summarize this algorithm first.

7.4.1 The ADMM algorithm

The algorithm in [64] solves the homogeneous self-dual embedding [77] of the conic program (7.19) and its dual,

$$\begin{aligned} \underset{y,z}{\text{maximize}} \quad & b^\mathsf{T} y \\ \text{subject to} \quad & A^\mathsf{T} y + z = c. \\ & z \in \mathcal{K}^*, \end{aligned} \tag{7.20}$$

where the cone $\mathcal{K}^*$ is the dual of $\mathcal{K}$. When strong duality holds, optimal solutions for (7.19) and (7.20) or a certificate of primal or dual infeasibility can be recovered from a nonzero solution of the homogeneous linear system

$$\begin{bmatrix} z \\ s \\ \kappa \end{bmatrix} = \begin{bmatrix} 0 & -A^\mathsf{T} & c \\ A & 0 & -b \\ -c^\mathsf{T} & b^\mathsf{T} & 0 \end{bmatrix} \begin{bmatrix} \xi \\ y \\ \tau \end{bmatrix}, \tag{7.21}$$

provided that it also satisfies $(\xi, y, \tau) \in \mathcal{K} \times \mathbb{R}^m \times \mathbb{R}_+$ and $(z, s, \kappa) \in \mathcal{K}^* \times \{0\}^m \times \mathbb{R}_+$. The interested reader is referred to [64] and references therein for more details. Consequently, upon defining

$$u := \begin{bmatrix} \xi \\ y \\ \tau \end{bmatrix}, \qquad v := \begin{bmatrix} z \\ s \\ \kappa \end{bmatrix}, \qquad Q := \begin{bmatrix} 0 & -A^\mathsf{T} & c \\ A & 0 & -b \\ -c^\mathsf{T} & b^\mathsf{T} & 0 \end{bmatrix}, \tag{7.22}$$

and introducing the cones $\mathcal{C} := \mathcal{K} \times \mathbb{R}^m \times \mathbb{R}_+$ and $\mathcal{C}^* := \mathcal{K}^* \times \{0\}^m \times \mathbb{R}_+$ to ease notation, a primal-dual optimal point for problems (7.19) and (7.20) or a certificate of infeasibility can be computed from a nonzero solution of the homogeneous self-dual feasibility problem

$$\begin{array}{ll} \text{find} & (u, v) \\ \text{subject to} & v = Qu, \\ & (u, v) \in \mathcal{C} \times \mathcal{C}^*. \end{array} \tag{7.23}$$

It was shown in [64] that (7.23) can be solved using a simplified version of the classical ADMM algorithm (see *e.g.*, [72]), whose k-th iteration consists of the following three steps ($\mathbb{P}_{\mathcal{C}}$ denotes projection onto the cone $\mathcal{C}$, and the superscript (k) indicates the value of a variable after the k-th iteration):

$$\hat{u}^{(k)} = (I + Q)^{-1} \left(u^{(k-1)} + v^{(k-1)} \right), \tag{7.24a}$$

$$u^{(k)} = \mathbb{P}_{\mathcal{C}} \left(\hat{u}^{(k)} - v^{(k-1)} \right), \tag{7.24b}$$

$$v^{(k)} = v^{(k-1)} - \hat{u}^{(k)} + u^{(k)}. \tag{7.24c}$$

Practical implementations of the algorithm rely on being able to carry out these steps at moderate computational cost. We next show that partial orthogonality allows for an efficient implementation of (7.24a) when (7.23) represents an SOS program.

7.4.2 Application to SOS programming

Each iteration of the ADMM algorithm requires: a projection onto a linear subspace in (7.24a) through the solution of a linear system with coefficient matrix $I+Q$; a projection onto the cone $\mathcal{C}$ in (7.24b); and the inexpensive step (7.24c). The conic projection (7.24b) can be computed efficiently when the cone size is not too large. On the other hand, $Q \in \mathbb{S}^{t+N^2+m+1}$ and $m = \mathcal{O}(n^{2d})$ are extremely large in SDPs arising from SOS programs. For instance, an SOS program with polynomials of degree $2d = 6$ in $n = 16$ variables has a PSD variable of size $N = 969$ and $m = 74\,613$ equality constraints. This makes step (7.24a) computationally expensive not only if $I + Q$ is factorized directly, but also when applying the strategies proposed in [64]. Fortunately, Q is highly structured and, in the context of SOS programming, the block-entry A has partially orthogonal rows (cf. Propositions 7.1 and 7.6). As we will now show, these properties can be taken advantage of to achieve substantial computational savings.

To show how partial orthogonality can be exploited, we begin by noticing that (7.24a) requires the solution of a linear system of equations of the form

$$\begin{bmatrix} I & -A^\mathsf{T} & c \\ A & I & -b \\ -c^\mathsf{T} & b^\mathsf{T} & 1 \end{bmatrix} \begin{bmatrix} \hat{u}_1 \\ \hat{u}_2 \\ \hat{u}_3 \end{bmatrix} = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}. \tag{7.25}$$

After letting

$$M := \begin{bmatrix} I & -A^{\mathsf{T}} \\ A & I \end{bmatrix}, \quad \zeta := \begin{bmatrix} c \\ -b \end{bmatrix},$$

and eliminating $\hat{u}_3$ from the first and second block-equations in (7.25) we obtain

$$(M + \zeta\zeta^{\mathsf{T}}) \begin{bmatrix} \hat{u}_1 \\ \hat{u}_2 \end{bmatrix} = \begin{bmatrix} \omega_1 \\ \omega_2 \end{bmatrix} - \omega_3\zeta. \tag{7.26a}$$

$$\hat{u}_3 = \omega_3 + c^{\mathsf{T}}\hat{u}_1 - b^{\mathsf{T}}\hat{u}_2. \tag{7.26b}$$

Applying the matrix inversion lemma [3] to (7.26a) yields

$$\begin{bmatrix} \hat{u}_1 \\ \hat{u}_2 \end{bmatrix} = \left[I - \frac{(M^{-1}\zeta)\zeta^{\mathsf{T}}}{1 + \zeta^{\mathsf{T}}(M^{-1}\zeta)} \right] M^{-1} \begin{bmatrix} \omega_1 - c\omega_3 \\ \omega_2 + b\omega_3 \end{bmatrix}. \tag{7.27}$$

Note that the first matrix on the right-hand side of (7.27) only depends on problem data, and can be computed before iterating the ADMM algorithm. Consequently, all that is left to do at each iteration is to solve a linear system of equations of the form

$$\begin{bmatrix} I & -A^{\mathsf{T}} \\ A & I \end{bmatrix} \begin{bmatrix} \sigma_1 \\ \sigma_2 \end{bmatrix} = \begin{bmatrix} \hat{\omega}_1 \\ \hat{\omega}_2 \end{bmatrix}. \tag{7.28}$$

Eliminating σ_1 from the second block-equation in (7.28) gives

$$\sigma_1 = \hat{\omega}_1 + A^{\mathsf{T}}\sigma_2, \tag{7.29a}$$

$$(I + AA^{\mathsf{T}})\sigma_2 = -A\hat{\omega}_1 + \hat{\omega}_2. \tag{7.29b}$$

It is at this stage that partial orthogonality comes into play: by Propositions 7.1 and 7.6, there exists a diagonal matrix P such that $I + AA^{\mathsf{T}} = I + A_1A_1^{\mathsf{T}} + A_2A_2^{\mathsf{T}} = P + A_1A_1^{\mathsf{T}}$. Recalling from Section 7.3 that $A_1 \in \mathbb{R}^{m\times t}$ with $t \ll m$ for typical SOS programs (*e.g.*, $t = 3$ and $m = 58$ for problem `sosdemo2` in SOSTOOLS), it is therefore convenient to apply the matrix inversion lemma to (7.29b) and write

$$\begin{aligned}(I + AA^{\mathsf{T}})^{-1} &= (P + A_1A_1^{\mathsf{T}})^{-1} \\ &= P^{-1} - P^{-1}A_1(I + A_1^{\mathsf{T}}P^{-1}A_1)^{-1}A_1^{\mathsf{T}}P^{-1}.\end{aligned}$$

Since P is diagonal, its inverse is immediately computed. Then, σ_1 and σ_2 in (7.29) are found upon solving a $t \times t$ linear system with coefficient matrix

$$I + A_1^{\mathsf{T}}P^{-1}A_1 \in \mathbb{S}^t, \tag{7.30}$$

plus relatively inexpensive matrix-vector, vector-vector, and scalar-vector operations. Moreover, since the matrix $I + A_1^{\mathsf{T}}P^{-1}A_1$ depends only on the problem data and does not change at each iteration, its preferred factorization can be cached before iterating steps (7.24a)-(7.24c). Once σ_1 and σ_2 have been computed, the solution of (7.25) can be recovered using vector-vector and scalar-vector operations.

Remark 7.4. In [64], system (7.28) is solved either through a "direct" method based on a cached LDL^T factorization, or by applying the "indirect" conjugate-gradient (CG) method to (7.29b). Both these approaches are reasonably efficient, but exploiting partial orthogonality is advantageous because only a smaller linear system with size $t \times t$ need to be solved, with $t \leq m$ and typically $t \ll m$. When sparsity is ignored, each iteration of our method to solve (7.28) requires $\mathcal{O}(t^2 + mN^2 + mt)$ floating-point operations (flops), compared to $\mathcal{O}((t + N^2 + m)^2)$ flops for the "direct" method of [64] and $\mathcal{O}(n_{\text{cg}}m^2 + mN^2 + mt)$ flops for the "indirect" method with n_{cg} CG iterations. Of course, practical implementations of the methods of [64] exploit sparsity and have a much lower complexity than stated, but the results in Section 7.7 confirm that the strategy outlined in this work remains more efficient.

7.5 Matrix-valued SOS programs

Up to this point, we have discussed partial orthogonality for scalar-valued SOS programs, but our results and the algorithm proposed in Section 7.4 extend also to the matrix-valued case. Given symmetric matrices $C_\alpha \in \mathbb{S}^r$, we say that the symmetric matrix-valued polynomial

$$P(x) := \sum_{\alpha \in \mathbb{N}^n_{2d}} C_\alpha x^\alpha$$

is an SOS matrix if there exits a $q \times r$ polynomial matrix $H(x)$ such that $P(x) = H(x)^\mathsf{T} H(x)$. Clearly, an SOS matrix is positive semidefinite for all $x \in \mathbb{R}^n$. It is known that the problem of checking wether $P(x)$ is SOS can be cast as an SDP.

Lemma 7.5 ([131, 142, 143])**.** *P is an SOS matrix if and only if there exists a PSD matrix $Q \in \mathbb{S}^l_+$ with $l = rN$ and $N = \binom{n+d}{d}$ such that*

$$P(x) = (I_r \otimes v_d(x))^\mathsf{T} Q \, (I_r \otimes v_d(x)) \,, \tag{7.31}$$

where $v_d(x) = [1, x_1, x_2, \ldots, x_n, x_1^2, x_1x_2, \ldots, x_n^d]^\mathsf{T}$ is the vector of monomials of degree up to d and $\otimes$ denotes the Kronecker product.

Similar to (7.10), we consider the matrix-valued SOS program

$$\begin{aligned} \underset{u}{\text{minimize}} \quad & w^\mathsf{T} u \\ \text{subject to} \quad & P(x) = P_0(x) - \sum_{h=1}^{t} u_h P_h(x), \\ & P(x) \text{ is SOS}, \end{aligned} \tag{7.32}$$

where $P_0(x)$, …, $P_t(x)$ are given symmetric polynomial matrices. Using (7.31), matching coefficients, and vectorizing, the matrix-valued SOS program (7.32) can be recast as a conic program of standard primal-form (7.19), for which the following proposition holds.

Proposition 7.6. The constraint matrix A in the conic program formulation of the matrix-valued SOS problem (7.32) has partially orthogonal rows, *i.e.*, it can be partitioned into $A = \begin{bmatrix} A_1 \, A_2 \end{bmatrix}$ such that $A_2 A_2^{\mathsf{T}}$ is diagonal.

Proof. First, introduce matrices $C_\alpha(u)$, affinely dependent on u, such that

$$P_0(x) - \sum_{h=1}^{t} u_h P_h(x) = \sum_{\alpha \in \mathbb{N}^n_{2d}} C_\alpha(u)\, x^\alpha.$$

By virtue of (7.13), the SOS representation (7.31) of $P(x)$ can be written as

$$P(x) = \sum_{\alpha \in \mathbb{N}^n_{2d}} \begin{bmatrix} \langle Y_{11}, B_\alpha \rangle & \dots & \langle Y_{1r}, B_\alpha \rangle \\ \vdots & \ddots & \vdots \\ \langle Y_{r1}, B_\alpha \rangle & \dots & \langle Y_{rr}, B_\alpha \rangle \end{bmatrix} x^\alpha,$$

where $Y_{ij} \in \mathbb{S}^N$, $i, j = 1, \dots, r$ is the (i,j)-th block of matrix $Y \in \mathbb{S}^l_+$. Then, the equality constraints in (7.32) require

$$C_\alpha(u) = \begin{bmatrix} \langle Y_{11}, B_\alpha \rangle & \dots & \langle Y_{1r}, B_\alpha \rangle \\ \vdots & \ddots & \vdots \\ \langle Y_{r1}, B_\alpha \rangle & \dots & \langle Y_{rr}, B_\alpha \rangle \end{bmatrix}, \quad \forall \alpha \in \mathbb{N}^n_{2d}. \tag{7.33}$$

Upon vectorization, this set of affine equalities can be written compactly as

$$\begin{bmatrix} A_1 & A_2 \end{bmatrix} \begin{bmatrix} u \\ \mathrm{vec}(Y) \end{bmatrix} = b \tag{7.34}$$

for suitably defined matrices A_1, A_2 and a vector b.

The matrix A_1 depends on the matrices $C_\alpha(u)$, and generally has no particular structure. Instead, A_2 has orthogonal rows, hence $A_2 A_2^{\mathsf{T}}$ is diagonal. To see this, let $e_i \in \mathbb{R}^r$ be the standard unit vector in the i-th direction and define

$$E_i := e_i \otimes I_N \in \mathbb{R}^{l \times N},$$

so $E_i^{\mathsf{T}} Y E_j = Y_{ij}$ selects the (i,j)-th $N \times N$ block of Y. Moreover, let $(C_\alpha)_{ij}$ denote the (i,j)-th element of the matrix C_α. The linear equalities (7.33) require that, for all $i, j = 1, \dots, r$ and all $\alpha \in \mathbb{N}^n_{2d}$,

$$\langle E_i^{\mathsf{T}} Y E_j, B_\alpha \rangle = (C_\alpha)_{ij}. \tag{7.35}$$

Vectorization of the left-hand side yields

$$\mathrm{vec}(B_\alpha)^{\mathsf{T}} (E_j^{\mathsf{T}} \otimes E_i^{\mathsf{T}}) \mathrm{vec}(Y) = (C_\alpha)_{ij}.$$

It is then not difficult to see that the rows of the matrix A_2 in (7.34) are the vectors $\mathrm{vec}(B_\alpha)^{\mathsf{T}} \cdot (E_j^{\mathsf{T}} \otimes E_i^{\mathsf{T}})$ for all triples (α, i, j) (the precise order of the rows is not important).

To show that $A_2 A_2^\mathsf{T}$ is diagonal, therefore, it suffices to show that, for any two different triples (α_1, i_1, j_1) and (α_2, i_2, j_2),

$$\begin{aligned} 0 &= \mathrm{vec}(B_{\alpha_1})^\mathsf{T}(E_{j_1}^\mathsf{T} \otimes E_{i_1}^\mathsf{T})(E_{j_2} \otimes E_{i_2})\mathrm{vec}(B_{\alpha_2}) \\ &= \mathrm{vec}(B_{\alpha_1})^\mathsf{T}(E_{j_1}^\mathsf{T} E_{j_2} \otimes E_{i_1}^\mathsf{T} E_{i_2})\mathrm{vec}(B_{\alpha_2}), \end{aligned} \tag{7.36}$$

where the second equality follows from the properties of the Kronecker product. To show (7.36), we invoke the properties of the Kronecker product once again to write

$$E_i^\mathsf{T} E_j = (e_i^\mathsf{T} e_j) \otimes I_N = \begin{cases} I_N, & \text{if } i = j, \\ 0, & \text{otherwise}, \end{cases} \tag{7.37a}$$

$$\mathrm{vec}(B_\alpha)^\mathsf{T}\mathrm{vec}(B_\beta) = \begin{cases} n_\alpha, & \text{if } \alpha = \beta, \\ 0, & \text{otherwise}, \end{cases} \tag{7.37b}$$

where n_α is the number of nonzeros in B_α. It is then clear that (7.36) holds if, and in fact only if, $(\alpha_1, i_1, j_1) \neq (\alpha_2, i_2, j_2)$. Consequently, $A_2 A_2^\mathsf{T}$ is diagonal. ■

Proposition 7.6 reveals an inherent structural property of SDPs derived from matrix-valued SOS programs using the monomial basis, and the algorithm of Section 7.4 applies *verbatim* because the conic program representation of scalar- and matrix-valued SOS programs has the same general form.

7.6 Weighted SOS constraints

The discussion of Section 7.3 is general and encompasses all SOS programs once they are recast in the form (7.7). As already mentioned in Section 7.2.1, handling SOS constraints over semialgebraic sets through (7.7) requires introducing extra optimization variables, which is not desirable in practice. To overcome this difficulty, we show here that partial orthogonality holds also for so-called "weighted" SOS constraints. Specifically, consider a family of fixed polynomials $g_0, \ldots, g_t \in \mathbb{R}[x]_{n,2d}$, a second family of fixed polynomials $p_1 \in \mathbb{R}[x]_{n,d_1}, \ldots, p_k \in \mathbb{R}[x]_{n,d_k}$, and let $\omega_i := \lfloor d - d_i/2 \rfloor$ for each $i = 1, \ldots, k$. (We have assumed that $d_1, \ldots, d_k \leq 2d$ without loss of generality.) We say that the polynomial

$$g(x) := g_0(x) - \sum_{i=1}^{t} u_i g_i(x) \tag{7.38}$$

is a weighted SOS with respect to $p_1, \ldots, p_k$ if there exist SOS polynomials $s_0 \in \Sigma[x]_{n,2d}$ and $s_i \in \Sigma[x]_{n,2\omega_i}$, $i = 1, \ldots, k$, such that

$$g(x) = s_0(x) + \sum_{i=1}^{k} p_i(x) s_i(x). \tag{7.39}$$

It is not difficult to see that if $g(x)$ is a weighted SOS with respect to $p_1, \ldots, p_k$, then it is non-negative on the semialgebraic set $\mathcal{S} := \{x \in \mathbb{R}^n : p_1(x) \geq 0, \ldots, p_k(x) \geq 0\}$. Thus, weighted SOS constraints arise naturally when polynomial inequalities on semialgebraic sets are cast as SOS conditions using the *Positivstellensatz* [126].

To put (7.39) in the form used by the standard conic program (7.19), we begin by introducing Gram matrix representations for each SOS poynomial. That is, we consider matrices $X_0 \in \mathbb{S}_+^{N_0}$, $X_1 \in \mathbb{S}_+^{N_1}$, $\ldots$, $X_k \in \mathbb{S}_+^{N_k}$, with $N_0 := \binom{n+d}{d}$ and $N_i = \binom{n+\omega_i}{\omega_i}$ for $i = 1, \ldots, k$, and rewrite (7.39) as

$$g(x) = \langle v_d(x)v_d(x)^\mathsf{T}, X_0\rangle + \sum_{i=1}^{k} p_i(x)\langle v_{\omega_i}(x)v_{\omega_i}(x)^\mathsf{T}, X_i\rangle. \tag{7.40}$$

In this expression, the vector $v_d(x)$ is as in (7.5) and, similarly, $v_{\omega_i}(x)$ lists the monomials of degree no larger than ω_i.

At this stage, let B_α be the mutually orthogonal 0/1 indicator matrix for the monomial x^α in the outer product matrix $v_d(x)v_d(x)^\mathsf{T}$, defined as in (7.12), such that (7.13) holds. Similarly, introduce symmetric indicator matrices $B_\alpha^{(i)}$ such that

$$p_i(x)v_{\omega_i}(x)v_{\omega_i}^\mathsf{T}(x) = \sum_{\alpha \in \mathbb{N}^n_{2d}} B_\alpha^{(i)} x^\alpha.$$

Note that the matrices $B_\alpha^{(i)}$ are not pairwise orthogonal in general: their nonzero entries overlap to some extent because the entries of the matrix $p_i(x)v_{\omega_i}(x)v_{\omega_i}^\mathsf{T}(x)$ are typically polynomials rather than simple monomials. Pairwise orthogonality holds for $B_\alpha^{(i)}$ if p_i is a monomial, but this is uncommon in practice. Using such indicator matrices, (7.40) can be written as

$$g(x) = \sum_{\alpha \in \mathbb{N}^n_{2d}} \left(\langle B_\alpha, X_0\rangle + \sum_{i=1}^{k} \langle B_\alpha^{(i)}, X_i\rangle \right) x^\alpha, \tag{7.41}$$

and we require that the coefficients of the monomials x^α on both sides of this expression match. To do this in compact notation, we index the monomials x^α using integers $1, \ldots, m$ as in Section 7.3 and define the $m \times \sum_{i=1}^k N_i^2$ matrix

$$A_2 := \begin{bmatrix} \mathrm{vec}(B_1^{(1)})^\mathsf{T} & \cdots & \mathrm{vec}(B_1^{(k)})^\mathsf{T} \\ \vdots & & \vdots \\ \mathrm{vec}(B_m^{(1)})^\mathsf{T} & \cdots & \mathrm{vec}(B_m^{(k)})^\mathsf{T} \end{bmatrix}, \tag{7.42}$$

the $m \times N_0^2$ matrix

$$A_3 := \begin{bmatrix} \mathrm{vec}(B_1)^\mathsf{T} \\ \vdots \\ \mathrm{vec}(B_m)^\mathsf{T} \end{bmatrix}, \tag{7.43}$$

and the vector

$$\chi := \left[\mathrm{vec}(X_1)^\mathsf{T}, \cdots, \mathrm{vec}(X_k)^\mathsf{T}\right]^\mathsf{T}. \tag{7.44}$$

Recalling the definition of $g(x)$ in (7.38), we can then use the $m \times t$ matrix A_1 defined in (7.17) and the vector b in (7.18b) to write the coefficient matching conditions obtained from (7.41) in the matrix-vector form

$$\begin{bmatrix} A_1 & A_2 & A_3 \end{bmatrix} \begin{bmatrix} u \\ \chi \\ \mathrm{vec}(X_0) \end{bmatrix} = b. \tag{7.45}$$

As already noticed in Section 7.3, nonzero entries in B_i must be zero in B_j if $i \neq j$, so the rows of A_3 are mutually orthogonal. Since (7.45) corresponds to the equality constraints in the conic program formulation of a weighted SOS constraint, we obtain the following result.

Proposition 7.7. The constraint matrix in the conic program formulation of the weighted SOS constraint (7.39) has partially orthogonal rows, *i.e.*, it can be partitioned as $\begin{bmatrix} A_1\, A_2\, A_3 \end{bmatrix}$ such that $A_3 A_3^\mathsf{T}$ is diagonal.

In other words, partial orthogonality obtains also when weighted SOS constraints are dealt with directly. Thus, the ADMM algorithm descibed in Section 7.4 can in principle be applied to solve SOS programs with weighted SOS constraints. Applying the matrix inversion lemma as proposed in Section 7.4 is advantageous if $t + \sum_{i=1}^k N_i^2 < m$, meaning that the degree $\omega_1, \ldots, \omega_k$ of the SOS polynomials $s_1, \ldots, s_k$ in (7.39) should be small such that

$$t + \sum_{i=1}^{k} \binom{n+\omega_i}{\omega_i} < \binom{n+2d}{2d} =: m. \tag{7.46}$$

Table 7.1 confirms that this is not unusual for typical problems. When (7.46) does not hold, instead of implementing weighted SOS constraints directly, it may be more convenient to introduce extra polynomials $s_1, \ldots, s_k$, and to consider the equivalent problem

$$\begin{aligned} \text{find} \quad & s_0, s_1, \ldots, s_k, r_1, \ldots, r_k \\ \text{subject to} \quad & g(x) = s_0(x) + \sum_{i=1}^{k} r_i(x) p_i(x), \\ & s_j(x) = r_j(x), \quad j = 1, \ldots, k, \\ & s_0 \in \Sigma[x]_{n,2d}, \\ & s_j \in \Sigma[x]_{n,2\omega_j}, \quad j = 1, \ldots, k. \end{aligned} \tag{7.47}$$

This can be written in the form (7.7) for a suitable set of polynomials $\{g_i^j\}$. Consequently, the partial orthogonality property holds for the reformulation (7.47). Note that while the introduction of extra polynomials allows one to reformulate weighted SOS constraints in the framework given by (7.7), it may be undesirable in practice because it increases the number of optimization variables.

7.7 Numerical experiments

We implemented the algorithm of [64], extended to take into account partial orthogonality in SOS programs, as a new package in the open-source MATLAB solver CDCS [35]. Our implementation, which we refer to as CDCS-sos, solves step (7.24a) using a sparse permuted Cholesky factorization of the matrix in (7.30). The source code can be downloaded from

https://github.com/oxfordcontrol/cdcs.

We tested CDCS-sos on a series of SOS programs and our test scripts are available from https://github.com/zhengy09/sosproblems. CPU times were compared to the direct and indirect implementations of the algorithm of [64] provided by the solver SCS [65], referred to as SCS-direct and SCS-indirect, respectively. In our experiments, the termination tolerance for CDCS-sos and SCS was set to 10^{-3}, and the maximum number of iterations was 2 000. Since first-order methods only aim at computing a solution of moderate accuracy, we assessed the suboptimality of the solution returned by CDCS-sos by comparing it to an accurate solution computed with the interior-point solver SeDuMi [71]. Besides, to demonstrate the low memory requirements of first-order algorithms, we also tested the interior-point solvers SDPT3 [137], SDPA [112], CSDP [138] and Mosek [139] for comparison. All interior-point solvers were called with their default parameters and their optimal values (when available) agree to within 10^{-8}. All computations were carried out on a PC with a 2.8 GHz Intel® Core™ i7 CPU and 8GB of RAM; memory overflow is marked by ** in the tables below.

7.7.1 Constrained polynomial optimization

As our first numerical experiment, we considered the constrained quartic polynomial minimization problem

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \sum_{1\le i<j\le n} (x_i x_j + x_i^2 x_j - x_j^3 - x_i^2 x_j^2) \\ \text{subject to} \quad & \sum_{i=1}^{n} x_i^2 \le 1. \end{aligned} \tag{7.48}$$

We used the Lasserre relaxation of order $2d = 4$ and the parser GloptiPoly [136] to recast (7.48) into an SDP.

Table 7.1 reports the CPU time (in seconds) required by each of the solvers we tested to solve the SDP relaxations as the number of variables n was increased. CDCS-sos is the fastest method in all cases. For large-scale POPs ($n \ge 29$), the number of constraints in the resulting SDP is over 40, 000, and all interior-point solvers (SeDuMi, SDPT3, SDPA, CSDP and Mosek) ran out of memory on our machine. The first-order solvers do not suffer from this limitation, and for POPs with $n \ge 29$ variables our MATLAB solver was

Table 7.1: CPU time (in seconds) to solve the SDP relaxations of (7.48). N is the size of the largest PSD cone, m is the number of constraints, t is the size of the matrix factorized by CDCS-sos.

	Dimensions			CPU time (s)							
n	N	m	t	SeDuMi	SDPT3	SDPA	CSDP	Mosek	SCS-direct	SCS-indirect	CDCS-sos
10	66	1 000	66	2.6	2.1	1.6	2.5	0.8	0.4	0.4	0.4
12	91	1 819	91	12.3	7.0	5.7	4.0	2.4	0.7	0.8	0.7
14	120	3 059	120	68.4	24.2	18.1	13.5	6.5	1.7	1.7	1.4
17	171	5 984	171	516.9	129.6	97.9	75.8	38.1	4.6	4.4	3.5
20	231	10 625	231	2 547.4	494.1	452.7	374.2	178.9	10.6	10.6	8.5
24	325	20 474	325	**	**	2 792.8	2 519.3	1 398.3	32.0	31.2	22.8
29	465	40 919	465	**	**	**	**	**	125.9	126.3	67.1
35	666	82 250	666	**	**	**	**	**	425.3	431.3	216.9
42	946	163 184	946	**	**	**	**	**	1 415.8	1 436.9	686.6

Table 7.2: Terminal objective value from interior-point solvers, SCS-direct, SCS-indirect and CDCS-sos for the SDP relaxation of (7.48).

n	†Interior-point solvers	SCS-direct	SCS-indirect	CDCS-sos
10	−9.11	−9.12	−9.13	−9.10
12	−11.12	−11.10	−11.10	−11.11
14	−13.12	−13.09	−13.09	−13.12
17	−16.12	−16.09	−16.09	−16.06
20	−19.12	−19.17	−19.17	−19.08
24	−23.12	−23.04	−23.04	−23.15
29	**	−28.17	−28.18	−28.17
35	**	−34.05	−34.05	−34.08
42	**	−41.21	−41.21	−41.05

approximately twice as fast as SCS. This is remarkable considering the SCS is written in C, and is due to the fact that $t \ll m$, cf. Table 7.1, so the cost of the affine projection step (7.24a) in CDCS-sos is greatly reduced compared to the methods implemented in SCS. Figure 7.2(a) illustrates that, for all test problems, CDCS-sos was faster than both SCS-direct and SCS-indirect also in terms of average CPU time per 100 iterations (this metric is unaffected by differences in the termination criteria used by different solvers). Finally, Table 7.2 shows that although first-order methods only aim to provide solutions of moderate accuracy, the objective value returned by CDCS-sos and SCS was always within 0.5% of the high-accuracy optimal value computed using interior-point solvers. Such a small difference may be considered negligible in many applications.

7.7.2 Finding Lyapunov functions

In our next numerical experiment, we considered the problem of constructing Lyapunov functions to verify local stability of polynomial systems, *i.e.*, we solved the SOS relaxation of (7.3a)-(7.3b) for different system instances. We used SOSTOOLS [80] to generate the corresponding SDPs.

In the experiment, we randomly generated polynomial dynamical systems $\dot{x} = f(x)$ of degree three with an asymptotically stable equilibrium at the origin. We then checked for local nonlinear stability in the ball $\mathcal{D} = \{x \in \mathbb{R}^n : \sum_{i=1}^n x_i^2 \leq 0.1\}$ using a quadratic Lyapunov function of the form $V(x) = x^\mathsf{T} Q x$ and Positivstellensatz to derive SOS conditions from (7.3a) and (7.3b) (see *e.g.*, [126] for more details). The total CPU time required by the solvers we tested are reported in Table 7.3, while Figure 7.2(b) shows the average CPU times per 100 iterations for SCS and CDCS-sos. As in our previous experiment, the results clearly show that the iterations in CDCS-sos are faster than in SCS for all our random problem instances, and that both first-order solvers have low memory requirements and are able to solve large-scale problems ($n \geq 29$) beyond the reach of interior-point solvers.

Table 7.3: CPU time (in seconds) to solve the SDP relaxations of (7.3a)-(7.3b). N is the size of the largest PSD cone, m is the number of constraints, t is the size of the matrix factorized by CDCS-sos.

	Dimensions			CPU time (s)							
n	N	m	t	SeDuMi	SDPT3	SDPA	CSDP	Mosek	SCS-direct	SCS-indirect	CDCS-sos
10	65	1 100	110	2.8	1.8	2.0	2.6	0.7	0.2	0.2	0.3
12	90	1 963	156	6.3	4.9	3.5	1.0	2.1	0.3	0.3	0.4
14	119	3 255	210	36.2	16.3	44.8	2.6	5.5	0.8	0.7	0.6
17	170	6 273	306	265.1	78.0	204.7	9.5	26.9	1.3	1.3	1.1
20	230	11 025	420	1 346.0	361.3	940.5	40.4	112.5	3.1	3.0	2.4
24	324	21 050	600	**	**	8 775.5	238.4	632.2	15.1	6.6	5.1
29	464	41 760	870	**	**	**	**	**	17.1	16.9	14.3
35	665	83 475	1260	**	**	**	**	**	67.6	57.1	37.4
42	945	164 948	1806	**	**	**	**	**	133.7	129.2	92.8

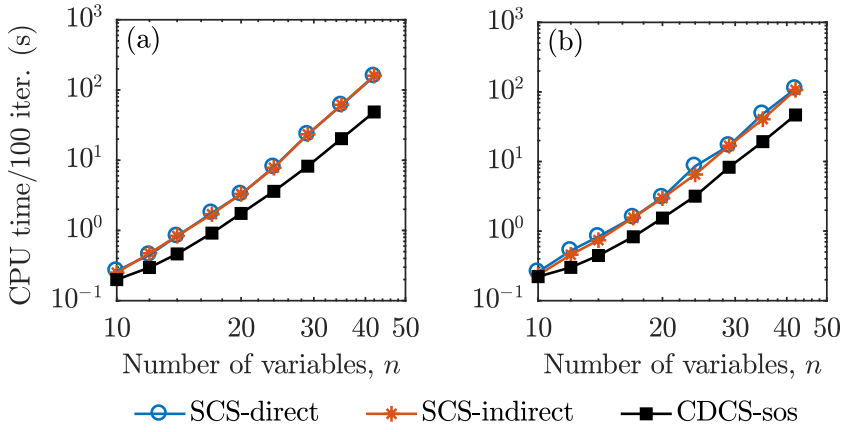


Figure 7.2: Average CPU time per 100 iterations for the SDP relaxations of: (a) the POP (7.48); (b) the Lyapunov function search problem.

7.7.3 A practical example: Nuclear receptor signalling

As our last example, we considered a 37-state model of nuclear receptor signalling with a cubic vector field and an equilibrium point at the origin [144, Chapter 6]. We verified its local stability within a ball of radius 0.1 by constructing a quadratic Lyapunov function. SOSTOOLS [80] was used to recast the SOS relaxation of (7.3a)-(7.3b) as an SDP with constraint matrix of size $102\,752 \times 553\,451$ and a large PSD cone of linear dimension 741. Such a large-scale problem is currently beyond the reach of interior-point methods on a regular desktop computer, and all of the interior point solvers we tested (SeDuMi, SDPT3, SDPA, CSDP and Mosek) ran out of memory on our machine. On the other hand, the first-order solvers CDCS-sos and SCS managed to construct a valid Lyapunov function, with our partial-orthogonality-exploiting algorithm being more than twice as fast as SCS (148 s vs. $\approx$ 400 s for both SCS-direct and SCS-indirect).

7.8 Conclusion

In this chapter, we proved that SDPs arising from SOS programs formulated using the standard monomial basis possess a structural property that we call partial orthogonality. We then demonstrated that this property can be leveraged to substantially reduce the computational cost of an ADMM algorithm for conic programs proposed in [64]. Specifically, we showed that the iterates of this algorithm can be projected efficiently onto a set defined by the affine constraints of the SDP. The key idea is to exploit a "diagonal plus low rank" structure of a large matrix that needs to be inverted/factorized, which is a direct consequence of partial orthogonality. Numerical experiments on large-scale SOS programs demonstrate that the method proposed in this chapter yield considerable savings compared to many state-of-the-art solvers. For this reason, we expect that our method will facilitate the use of SOS programming for the analysis and design of large-scale systems.

8

Decomposition and completion of sum-of-squares matrices

This chapter introduces a notion of decomposition and completion of *sum-of-squares* (SOS) matrices. We show that a subset of sparse SOS matrices with chordal sparsity patterns can be equivalently decomposed into a sum of multiple SOS matrices that are nonzero only on a principal submatrix. Also, the completion of an SOS matrix is equivalent to a set of SOS conditions on its principal submatrices and a consistency condition on the Gram representation of the principal submatrices. These results are partial extensions of chordal decomposition and completion of constant matrices to matrices with polynomial entries. We apply the SOS decomposition result to exploit sparsity in matrix-valued SOS programs. Numerical results demonstrate the high potential of this approach for solving large-scale sparse matrix-valued SOS programs.

8.1 Introduction

Decomposition and completion of sparse positive semidefinite (PSD) matrices arise in a wide range of applications, including systems analysis and synthesis [34, 39] or optimal power flow [69], and have attracted considerable research attention [15, 19–21, 23, 25, 36, 38, 145, 146]. The concept of PSD decomposition refers to cases in which a sparse PSD matrix can be written as a sum of two or more PSD matrices, each of which is nonzero only on one principal submatrix. A simple example is

$$\underbrace{\begin{bmatrix} 2 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix}}_{\succeq 0} = \underbrace{\begin{bmatrix} 2 & 1 & 0 \\ 1 & 0.5 & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{\succeq 0} + \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0.5 & 1 \\ 0 & 1 & 2 \end{bmatrix}}_{\succeq 0}. \tag{8.1}$$

As one key ingredient of this thesis, it is known that this kind of decomposition always exists for a class of PSD matrices with chordal sparsity patterns [20, 21] (recall the

definition in Section 2.3). The PSD matrix completion problem, instead, asks whether a partially specified symmetric matrix, for instance

$$Z = \begin{bmatrix} 2 & 1 & ? \\ 1 & 0.5 & 1 \\ ? & 1 & 2 \end{bmatrix}, \tag{8.2}$$

can be completed into a PSD matrix by a suitable choice of the unspecified entries (indicated by the question mark ?). Clearly, a necessary condition for the existence of a PSD completion is that the fully specified principal submatrices are PSD. It turns out that this condition is also sufficient for matrices with chordal sparsity patterns [19] (see Theorem 2.13 in Chapter 2.3). The partial matrix in (8.2) has a chordal sparsity pattern and its specified principal submatrices are PSD, so it admits a PSD completion. One such completion is

$$Z = \begin{bmatrix} 2 & 1 & 2 \\ 1 & 0.5 & 1 \\ 2 & 1 & 2 \end{bmatrix} \succeq 0.$$

The decomposition and completion results for sparse PSD matrices mentioned above [19–21] have a very important practical implication for optimization problems with PSD matrices: they allow us to replace a large PSD constraint with an equivalent set of smaller, coupled PSD constraints, thereby promising better computational scalability. This feature indeed underpins the idea of this thesis, as well as much of the recent research on exploiting sparsity in conic programs, either by interior-point methods [23, 25] or by first-order methods [26, 36, 38].

In this chapter, we provide a partial extension of the results in [19–21] to sparse polynomial matrices, which are common in fields such as robust semidefinite programming [142] and control theory [147]. Specifically, we consider the problem of decomposing and completing sparse matrices whose entries are polynomials with real coefficients in n variables $x_1, \ldots, x_n$. For example, given the polynomial matrix

$$\begin{bmatrix} x^2+1 & x & 0 \\ x & x^2-2x+3 & x+1 \\ 0 & x+1 & x^2+2 \end{bmatrix}, \tag{8.3}$$

which has the same pattern as the matrix in (8.1) and is PSD for all $x \in \mathbb{R}$, we seek to determine whether it can be decomposed into a sum of two PSD polynomial matrices of the form

$$\underbrace{\begin{bmatrix} * & * & 0 \\ * & * & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{\succeq 0} + \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & * & * \\ 0 & * & * \end{bmatrix}}_{\succeq 0},$$

where $*$ denotes a polynomial in x. Also, we look for conditions under which a partially specified polynomial matrix, for instance

$$\begin{bmatrix} x^2+1 & x & ? \\ x & x^2-2x+2.5 & x+1 \\ ? & x+1 & x^2+2 \end{bmatrix}, \tag{8.4}$$

can be completed into a PSD polynomial matrix.

Since checking the positive semidefiniteness of a given symmetric polynomial matrix is NP-hard in general [52], this chapter focuses on a subset of PSD matrices given by *sum-of-squares (SOS)* matrices, which can be identified with polynomial-time algorithms using semidefinite programs (SDPs) [131, 142, 143]. Our motivation is the fact that SOS techniques are a powerful tool for systems analysis, control, and optimization (see, *e.g.*, [11, 52]), but they do not scale well with problem size. Most existing approaches to mitigate the scalability issue are based on the SOS representations of scalar polynomials; see [47] for an overview of recent advances. This chapter describes sufficient conditions for the decomposition of sparse SOS matrices into smaller ones, and for the existence of an SOS completion of a partial polynomial matrix. Thus, sparsity can be exploited to reduce the cost of computing with large and sparse SOS matrices.

The notion of decomposition and completion of SOS matrices has not been reported in the literature before. In this chapter, we use hyper-graphs to combine the Gram representation of SOS matrices [131, 142, 143] with the normal decomposition and completion results [19–21]. We prove that 1) the decomposition results for scalar matrices [20, 21] can be extended to a subset of sparse SOS matrices, and 2) the conditions for the existence of an SOS completion are similar to those for scalar matrices [19], with the addition of a consistency condition. Due to the non-uniqueness of their Gram matrix representation, however, our results of decomposition and completion for SOS matrices are not identical to those for scalar matrices in [19–21]. As a direct application, we use the new decomposition result to exploit sparsity in matrix-value SOS programs. Preliminary numerical results show the effectiveness of this approach.

The rest of this chapter is organized as follows. Section 8.2 presents a brief review on nonnegativity and sum-of-squares matrices. The results on decomposition and completion of sparse SOS matrices are given in Section 8.3 and Section 8.4, respectively. Section 8.5 discusses an application to matrix-valued SOS programs, including preliminary numerical examples. We conclude this chapter in Section 8.6.

8.2 Nonnegativity and sum-of-squares

For convenience, we recall some notation in this section. Let $\mathbb{R}[x]_{n,2d}$ be the set of polynomials in n variables with real coefficients of degree no more than $2d$. The set of $q \times r$ polynomial matrices with entries in $\mathbb{R}[x]_{n,2d}$ is denoted by $\mathbb{R}[x]_{n,2d}^{q\times r}$. A polynomial $p \in \mathbb{R}[x]_{n,2d}$ is PSD if $p(x) \geq 0$ for all $x \in \mathbb{R}^n$, and a symmetric polynomial matrix $P(x) \in \mathbb{R}[x]_{n,2d}^{r\times r}$ is PSD if $P(x) \succeq 0$ for all $x \in \mathbb{R}^n$.

Checking positive semidefiniteness of a polynomial $p(x)$ or a polynomial matrix $P(x)$ is NP-hard in general [52] and a popular tractable approach is to replace the PSD constraint by a sum-of-squares (SOS) constraint. We say that $p(x) \in \mathbb{R}[x]_{n,2d}$ is an SOS polynomial if there exists polynomials $f_i(x) \in \mathbb{R}[x]_{n,d}$, $i = 1, \ldots, s$ such that $p(x) = \sum_{i=1}^{s} f_i^2(x)$. Also, we define an SOS matrix as follows [131, 142, 143].

Definition 8.1. A symmetric polynomial matrix $P(x) \in \mathbb{R}[x]_{n,2d}^{r\times r}$ is an SOS matrix if there exists a polynomial matrix $M \in \mathbb{R}[x]_{n,d}^{s\times r}$ such that $P(x) = M^{\mathsf{T}}(x)M(x)$.

Clearly, the existence of an SOS representation ensures positive semidefiniteness. For simplicity, we denote the set of $r \times r$ SOS matrices with entries in $\mathbb{R}[x]_{n,2d}$ by $\Sigma_{n,2d}^r$. It is known that the problem of checking membership of $\Sigma_{n,2d}^r$ can be cast as an SDP; see Lemma 7.5 for details. Recall that the matrix Q in (7.31) is called the *Gram matrix* of the SOS representation and is usually not unique.

8.3 Decomposition of sparse SOS matrices

In practice, we may encounter sparse polynomial matrices in the sense that some entries are zeros, *e.g.*, (8.3). Similar to the sparse scalar matrix case in Section 2.3, we define the set of sparse SOS matrices characterized by an undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ as

$$\Sigma_{n,2d}^r(\mathcal{E}, 0) := \left\{ P(x) \in \Sigma_{n,2d}^r \mid p_{ij}(x) = p_{ji}(x) = 0, \text{ if } (i,j) \notin \mathcal{E}^* \right\}.$$

Given a sparse SOS matrix $P(x)$, Lemma 7.5 guarantees that its SOS representation can be written as

$$P(x) = \begin{bmatrix} v_d(x)^{\mathsf{T}} Q_{11} v_d(x) & \ldots & v_d(x)^{\mathsf{T}} Q_{1r} v_d(x) \\ \vdots & \ddots & \vdots \\ v_d(x)^{\mathsf{T}} Q_{r1} v_d(x) & \ldots & v_d(x)^{\mathsf{T}} Q_{rr} v_d(x) \end{bmatrix},$$

where $Q_{ij} \in \mathbb{R}^{N\times N}$, $i, j = 1, \ldots, r$ is the (i,j)-th block of the Gram matrix Q. If $P(x) \in \Sigma_{n,2d}^r(\mathcal{E}, 0)$, then

$$p_{ij}(x) = v_d(x)^{\mathsf{T}} Q_{ij} v_d(x) = 0 \text{ if } (i,j) \notin \mathcal{E}^*, \tag{8.5}$$

but Q_{ij} may be a nonzero matrix. Indeed, while Q is a symmetric matrix, the off-diagonal block Q_{ij} need not be so. This means the Gram matrix Q for a sparse SOS matrix $P(x) \in \Sigma^r_{n,2d}(\mathcal{E}, 0)$ can be dense. To maintain the sparsity of $P(x)$ in the Gram matrix Q, we consider the subset of SOS matrices

$$\widetilde{\Sigma}^r_{n,2d}(\mathcal{E}, 0) := \Big\{ P(x) \in \Sigma^r_{n,2d}(\mathcal{E}, 0) \mid P(x) \text{ admits a Gram matrix } Q \succeq 0 \text{ with } Q_{ij} = 0 \text{ when } p_{ij}(x) = 0 \Big\}.$$

With this restriction, the following result holds.

Theorem 8.2 (SOS matrix decomposition)**.** Let $\mathcal{G}(\mathcal{V}, \mathcal{E})$ be a chordal graph with maximal cliques $\mathcal{C}_1, \ldots, \mathcal{C}_t$. Then, $P(x) \in \widetilde{\Sigma}^r_{n,2d}(\mathcal{E}, 0)$ if and only if there exist SOS matrices $P_k(x) \in \Sigma^{|\mathcal{C}_k|}_{n,2d}$, $k = 1, \ldots, t$, such that

$$P(x) = \sum_{k=1}^{t} E_{\mathcal{C}_k}^{\mathsf{T}} P_k(x) E_{\mathcal{C}_k}.$$

Proof. The "if" part is obvious. To prove the "only if" part we combine the Gram representation of SOS matrices with Theorem 2.10 in three steps.

Step 1 (Sparse Gram matrix): Since $P(x) \in \widetilde{\Sigma}^r_{n,2d}(\mathcal{E}, 0)$, the Gram matrix Q has a block sparsity pattern defined by $\mathcal{G}(\mathcal{V}, \mathcal{E})$. To describe the sparsity of Q in detail, we define a hyper-graph $\widetilde{\mathcal{G}}(\widetilde{\mathcal{V}}, \widetilde{\mathcal{E}})$ with a hyper-node set defined as

$$\widetilde{\mathcal{V}} = \{1, \ldots, N, N+1, \ldots, 2N, \ldots, (r-1)N+1, \ldots, rN\},$$

and a hyper-edge set defined as $\widetilde{\mathcal{E}} = \bigcup_{k=1}^{t} \widetilde{\mathcal{C}}_k \times \widetilde{\mathcal{C}}_k$, where each hyper-clique $\widetilde{\mathcal{C}}_k$ is defined as

$$\widetilde{\mathcal{C}}_k = \bigcup_{j \in \mathcal{C}_k} \{(j-1)N + 1, \ldots, jN\}. \tag{8.6}$$

The sparsity pattern of the Gram matrix Q is fully described by the graph $\widetilde{\mathcal{G}}(\widetilde{\mathcal{V}}, \widetilde{\mathcal{E}})$, that is, $Q \in \mathbb{S}^{rN}_+(\widetilde{\mathcal{E}}, 0)$. Moreover, since $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is chordal with maximal cliques $\mathcal{C}_1, \ldots, \mathcal{C}_t$ by assumption, the hyper-graph $\widetilde{\mathcal{G}}(\widetilde{\mathcal{V}}, \widetilde{\mathcal{E}})$ is also chordal and its maximal cliques are $\widetilde{\mathcal{C}}_1, \ldots, \widetilde{\mathcal{C}}_t$, as shown in Section 2.4.

Step 2 (Block chordal decomposition): Since the Gram matrix Q has a chordal sparsity pattern, Theorem 2.10 guarantees that $Q \in \mathbb{S}^{rN}_+(\widetilde{\mathcal{E}}, 0)$ if and only if

$$Q = \sum_{k=1}^{t} E_{\widetilde{\mathcal{C}}_k}^{\mathsf{T}} Q_k E_{\widetilde{\mathcal{C}}_k} \tag{8.7}$$

for some PSD matrices $Q_k \in \mathbb{S}^{|\widetilde{\mathcal{C}}_k|}_+$, $k = 1, \ldots, t$.

Step 3 (SOS matrix decomposition): According to (8.6), it is not difficult to see that

$$E_{\widetilde{\mathcal{C}}_k} = E_{\mathcal{C}_k} \otimes I_N, k = 1, \ldots, t. \tag{8.8}$$

Combining this with (8.7) yields

$$\begin{aligned} P(x) &= (I_r \otimes v_d(x))^{\mathsf{T}} Q (I_r \otimes v_d(x)) \\ &= (I_r \otimes v_d(x))^{\mathsf{T}} \left(\sum_{k=1}^{t} E_{\widetilde{\mathcal{C}}_k}^{\mathsf{T}} Q_k E_{\widetilde{\mathcal{C}}_k} \right) (I_r \otimes v_d(x)) \\ &= \sum_{k=1}^{t} \left[(I_r \otimes v_d(x))^{\mathsf{T}} E_{\widetilde{\mathcal{C}}_k}^{\mathsf{T}} Q_k E_{\widetilde{\mathcal{C}}_k} (I_r \otimes v_d(x)) \right]. \end{aligned} \tag{8.9}$$

Furthermore, using the properties of the Kronecker product and (8.8) we obtain

$$\begin{aligned} E_{\widetilde{\mathcal{C}}_k} (I_r \otimes v_d(x)) &= (E_{\mathcal{C}_k} \otimes I_N) \cdot (I_r \otimes v_d(x)) \\ &= E_{\mathcal{C}_k} \otimes v_d(x) \\ &= \left(I_{|\mathcal{C}_k|} \otimes v_d(x) \right) \cdot (E_{\mathcal{C}_k} \otimes 1) \\ &= \left(I_{|\mathcal{C}_k|} \otimes v_d(x) \right) \cdot E_{\mathcal{C}_k}. \end{aligned} \tag{8.10}$$

Finally, substituting (8.10) into (8.9) we arrive at

$$\begin{aligned} P(x) &= \sum_{k=1}^{t} \left[E_{\mathcal{C}_k}^{\mathsf{T}} \left(I_{|\mathcal{C}_k|} \otimes v_d(x) \right)^{\mathsf{T}} Q_k \left(I_{|\mathcal{C}_k|} \otimes v_d(x) \right) E_{\mathcal{C}_k} \right] \\ &= \sum_{k=1}^{t} E_{\mathcal{C}_k}^{\mathsf{T}} P_k(x) E_{\mathcal{C}_k}, \end{aligned}$$

where $P_k(x) \in \Sigma_{n,2d}^{|\mathcal{C}_k|}$ for all $k = 1, \ldots, t$. ■

Remark 8.3. The proof of Theorem 8.2 is based on hyper-graphs, combining the Gram representation of SOS matrices (*i.e.*, Lemma 7.5) with the normal chordal decomposition result (Theorem 2.10). Alternatively, it is not difficult to see $Q \in \mathbb{S}_{\alpha,+}^{Nr}(\mathcal{E}, 0)$, with partition $\alpha = \{N, \ldots, N\}$. Then one can also prove Theorem 8.2 directly using the block chordal decomposition result (Theorem 2.17).

Theorem 8.2 states a necessary and sufficient condition for checking membership to $\widetilde{\Sigma}_{n,2d}^{r}(\mathcal{E}, 0)$, which is a *strict* subset of the cone of SOS matrices since, as discussed previously, $p_{ij}(x) = 0$ does not require $Q_{ij} = 0$ in general. In addition, given $P(x) \in \widetilde{\Sigma}_{n,2d}^{r}(\mathcal{E}, 0)$, the proof of Theorem 8.2 offers a method to construct the small SOS matrices $P_k(x)$:

1. Find a sparse Gram matrix $Q \in \mathbb{S}_{+}^{rN}(\widetilde{\mathcal{E}}, 0)$;
2. Find a chordal decomposition $Q = \sum_{k=1}^{t} E_{\widetilde{\mathcal{C}}_k}^{\mathsf{T}} Q_k E_{\widetilde{\mathcal{C}}_k}$ (see, *e.g.*, [15, Chapter 9]);

3. Let $P_k(x) = \left(I_{|\mathcal{C}_k|} \otimes v_d(x)\right)^{\mathsf{T}} Q_k \left(I_{|\mathcal{C}_k|} \otimes v_d(x)\right)$ for each $k = 1, \ldots, t$.

For example, consider the polynomial matrix in (8.3). The relevant vector of monomials is $v_d(x) = [1, x]^{\mathsf{T}}$ and there exists a sparse Gram matrix with zero $(1,3)$ and $(3,1)$ blocks,

$$Q = \left[\begin{array}{cc|cc|cc} 1 & 0 & 0 & 0.4 & 0 & 0 \\ 0 & 1 & 0.6 & 0 & 0 & 0 \\ \hline 0 & 0.6 & 3 & -1 & 1 & 0.8 \\ 0.4 & 0 & -1 & 1 & 0.2 & 0 \\ \hline 0 & 0 & 1 & 0.2 & 2 & 0 \\ 0 & 0 & 0.8 & 0 & 0 & 1 \end{array}\right] \in \mathbb{S}^6_+(\widetilde{\mathcal{E}}, 0).$$

Thus, the matrix in (8.3) belongs to $\widetilde{\Sigma}^3_{1,2}$ and Theorem 8.2 applies. In particular, the Gram matrix above can be decomposed as $Q = E_{\widetilde{\mathcal{C}}_1}^{\mathsf{T}} Q_1 E_{\widetilde{\mathcal{C}}_1} + E_{\widetilde{\mathcal{C}}_2}^{\mathsf{T}} Q_2 E_{\widetilde{\mathcal{C}}_2}$ with

$$Q_1 = \left[\begin{array}{cc|cc} 1 & 0 & 0 & 0.4 \\ 0 & 1 & 0.6 & 0 \\ \hline 0 & 0.6 & 1.11 & -0.545 \\ 0.4 & 0 & -0.545 & 0.56 \end{array}\right] \in \mathbb{S}^4_+.$$

$$Q_2 = \left[\begin{array}{cc|cc} 1.89 & -0.455 & 1 & 0.8 \\ -0.455 & 0.44 & 0.2 & 0 \\ \hline 1 & 0.2 & 2 & 0 \\ 0.8 & 0 & 0 & 1 \end{array}\right] \in \mathbb{S}^4_+.$$

Then, an SOS decomposition for (8.3) is given as

$$P_1(x) = \begin{bmatrix} x^2 + 1 & x \\ x & 0.56x^2 - 1.09x + 1.11 \end{bmatrix} \in \Sigma^2_{1,2}$$

$$P_2(x) = \begin{bmatrix} 0.44x^2 - 0.91x + 1.89 & x + 1 \\ x + 1 & x^2 + 2 \end{bmatrix} \in \Sigma^2_{1,2}$$

We emphasize that the main interest of Theorem 8.2 is not on computing an actual SOS decomposition. Instead, this theorem offers a computationally efficient way to check if a matrix $P(x)$ belongs to $\widetilde{\Sigma}^r_{n,2d}(\mathcal{E}, 0)$, enabling the solution of large matrix-valued SOS programs (see Section 8.5).

8.4 Completion of sparse SOS matrices

Here, we give an analogue result to Theorem 2.13 for partial SOS matrices. Given a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, we say $P(x)$ is a partial symmetric polynomial matrix if $p_{ij}(x) = p_{ji}(x)$ are given when $(i,j) \in \mathcal{E}^*$. Moreover, we say that $F(x)$ is an SOS completion of the partial symmetric matrix $P(x)$ if $F(x)$ is SOS and $f_{ij}(x) = p_{ij}(x)$ when $(i,j) \in \mathcal{E}^*$. Precisely, we define the set of SOS completable matrices as

$$\Sigma^r_{n,2d}(\mathcal{E}, ?) = \left\{ P : \mathbb{R}^n \to \mathbb{S}^r(\mathcal{E}, 0) \mid \exists F \in \Sigma^r_{n,2d} \text{ such that } f_{ij}(x) = p_{ij}(x)\ \forall (i,j) \in \mathcal{E}^* \right\}.$$

For instance, the matrix in (8.4) is a partial symmetric polynomial matrix defined by a chain of three nodes, and we will show below that it is also SOS completable.

Theorem 8.4 (SOS matrix completion)**.** Let $\mathcal{G}(\mathcal{V},\mathcal{E})$ be a chordal graph with maximal cliques $\mathcal{C}_1,\ldots,\mathcal{C}_t$ and let $\widetilde{\mathcal{C}}_1,\ldots,\widetilde{\mathcal{C}}_t$ be the hyper-cliques defined in (8.6). Then, $P(x)\in\Sigma^r_{n,2d}(\mathcal{E},?)$ if and only if, for all $k=1,\ldots,t$,

$$P_k(x) := E_{\mathcal{C}_k}P(x)E_{\mathcal{C}_k}^{\mathsf{T}} \in \Sigma^{|\mathcal{C}_k|}_{n,2d} \tag{8.11}$$

and the Gram matrix Q_k corresponding to each $P_k(x)$ satisfies the following consistency condition:

$$\widetilde{\mathcal{C}}_i\cap\widetilde{\mathcal{C}}_j\neq\emptyset \implies E_{\widetilde{\mathcal{C}}_i\cap\widetilde{\mathcal{C}}_j}\left(E_{\widetilde{\mathcal{C}}_i}^{\mathsf{T}}Q_iE_{\widetilde{\mathcal{C}}_i}-E_{\widetilde{\mathcal{C}}_j}^{\mathsf{T}}Q_jE_{\widetilde{\mathcal{C}}_j}\right)E_{\widetilde{\mathcal{C}}_i\cap\widetilde{\mathcal{C}}_j}^{\mathsf{T}}=0. \tag{8.12}$$

Remark 8.5. Condition (8.12) states that elements of Q_i and Q_j, $i\neq j$, must be identical if they map to the same entries of the global Gram matrix Q, which represents the original polynomial matrix $P(x)$.

Proof. Similar to the proof of Theorem 8.2, we rely on the hyper-graph $\widetilde{\mathcal{G}}(\widetilde{\mathcal{V}},\widetilde{\mathcal{E}})$, which is chordal with a set of maximal cliques $\widetilde{\mathcal{C}}_1,\ldots,\widetilde{\mathcal{C}}_t$.

$\Leftarrow$: If $P(x)\in\Sigma^r_{n,2d}(\mathcal{E},?)$, then there exists an SOS matrix $F(x)$ with a Gram matrix $Q\in\mathbb{S}^{rN}_+$ such that

$$P_k(x)=E_{\mathcal{C}_k}P(x)E_{\mathcal{C}_k}^{\mathsf{T}}=E_{\mathcal{C}_k}F(x)E_{\mathcal{C}_k}^{\mathsf{T}},\quad k=1,\ldots t.$$

Also, using a property similar to (8.10), we have

$$\begin{aligned}E_{\mathcal{C}_k}F(x)E_{\mathcal{C}_k}^{\mathsf{T}}&=E_{\mathcal{C}_k}\left((I_r\otimes v_d(x))^{\mathsf{T}}Q(I_r\otimes v_d(x))\right)E_{\mathcal{C}_k}^{\mathsf{T}}\\&=(I_{|\mathcal{C}_k|}\otimes v_d(x))^{\mathsf{T}}Q_k(I_{|\mathcal{C}_k|}\otimes v_d(x)),\end{aligned}$$

where

$$Q_k=E_{\widetilde{\mathcal{C}}_k}QE_{\widetilde{\mathcal{C}}_k}^{\mathsf{T}}\in\mathbb{S}^{|\mathcal{C}_k|N}_+. \tag{8.13}$$

Therefore, $P_k(x)\in\Sigma^{|\mathcal{C}_k|}_{n,2d}$ and the Gram matrices Q_k in (8.13) satisfy the consistency condition (8.12).

$\Rightarrow$: When (8.11) and (8.12) hold we can form a partial symmetric matrix Q with $Q_k=E_{\widetilde{\mathcal{C}}_k}QE_{\widetilde{\mathcal{C}}_k}^{\mathsf{T}}$, $k=1,\ldots,t$. Since $Q_k\succeq 0$ and they hyper-graph $\widetilde{\mathcal{G}}(\widetilde{\mathcal{V}},\widetilde{\mathcal{E}})$ is chordal, Theorem 2.13 ensures that $Q\in\mathbb{S}^{rN}_+(\widetilde{\mathcal{E}},?)$. Then we can find a PSD completion $\hat{Q}\succeq 0$ for Q and hence an SOS completion $F(x)$ for $P(x)$, given by

$$F(x)=(I_r\otimes v_d(x))^{\mathsf{T}}\hat{Q}(I_r\otimes v_d(x))\in\Sigma^r_{n,2d}. \tag{8.14}$$

Therefore, we have $P(x)\in\Sigma^r_{n,2d}(\mathcal{E},?)$. ■

The proof of Theorem 8.4 is similar to that of Theorem 8.2, both of which utilize hyper-graphs and then apply the normal results of chordal decomposition and completion. Moreover, as before, given $P(x) \in \Sigma^r_{n,2d}(\mathcal{E}, ?)$ we can find an SOS completion using the following steps:

1. Find a PSD completable Gram matrix, $Q \in \mathbb{S}^{rN}_+(\widetilde{\mathcal{E}}, ?)$;
2. Find a PSD completion $\hat{Q}$ using any PSD completion algorithm (*e.g.*, [15, Chapter 10] and [23, Section 2]);
3. Construct the SOS completion as in (8.14).

Using this procedure, we are able to find an SOS completion for the matrix in (8.4), namely

$$\begin{bmatrix} x^2+1 & x & 0.3x^2+0.6x+0.3 \\ x & x^2-2x+2.5 & x+1 \\ 0.3x^2+0.6x+0.3 & x+1 & x^2+2 \end{bmatrix}.$$

8.5 Application to matrix-valued SOS programs

The SOS decomposition can be readily applied to exploit sparsity in matrix-valued SOS programs, which arise, for example, in robust semidefinite programming [142] and control theory [147]. Consider the matrix-valued SOS program

$$\begin{aligned} \underset{u}{\text{minimize}} \quad & w^{\mathsf{T}} u \\ \text{subject to} \quad & P(x) = P_0(x) - \sum_{i=1}^{h} u_i P_i(x), \\ & P(x) \in \Sigma^r_{n,2d}, \end{aligned} \tag{8.15}$$

where $u \in \mathbb{R}^h$ is the decision variable, $w \in \mathbb{R}^h$ defines a linear objective function, and $P_0(x), \ldots, P_t(x)$ are $r \times r$ symmetric polynomial matrices with a common sparsity pattern characterized by a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$. It is assumed that $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is chordal with maximal cliques $\mathcal{C}_1, \ldots, \mathcal{C}_t$, or that a chordal extension can be found [15]. Clearly, (8.15) is equivalent to

$$\begin{aligned} \underset{u}{\text{minimize}} \quad & w^{\mathsf{T}} u \\ \text{subject to} \quad & P(x) = P_0(x) - \sum_{i=1}^{h} u_i P_i(x), \\ & P \in \Sigma^r_{n,2d}(\mathcal{E}, 0). \end{aligned} \tag{8.16}$$

To exploit sparsity we replace the sparse SOS constraint $P \in \Sigma^r_{n,2d}(\mathcal{E}, 0)$ by the stronger condition $P \in \widetilde{\Sigma}^r_{n,2d}(\mathcal{E}, 0)$ and solve

$$\begin{aligned} \underset{u}{\text{minimize}} \quad & w^{\mathsf{T}} u \\ \text{subject to} \quad & P(x) = P_0(x) - \sum_{i=1}^{h} u_i P_i(x), \\ & P \in \widetilde{\Sigma}^r_{n,2d}(\mathcal{E}, 0). \end{aligned} \tag{8.17}$$

Table 8.1: CPU time (in seconds) required to solve (8.18) using different formulations.

Dimension r	10	20	30	40	50
Using (8.16)	0.25	4.1	72.6	425.2	1 773.1
Using (8.17)	0.11	0.13	0.23	0.25	0.38

Table 8.2: Objective value γ for (8.18) using different formulations.

Dimension r	10	20	30	40	50
Using (8.16)	−0.8516	−0.8403	−0.8364	−0.8344	−0.8332
Using (8.17)	−0.8516	−0.8403	−0.8364	−0.8344	−0.8332

Then, Theorem 8.2 enables us to decompose the single large SOS constraint $P \in \widetilde{\Sigma}^r_{n,2d}(\mathcal{E}, 0)$ with a set of coupled matrix SOS constraints with smaller dimensions. This can reduce the computational cost of (8.17) significantly if the size of maximal cliques $\mathcal{C}_1, \ldots, \mathcal{C}_t$ is small. To demonstrate this, we consider the special matrix-valued SOS program

$$\begin{array}{ll} \underset{\gamma}{\text{minimize}} & \gamma \\ \text{subject to} & P(x) + \gamma I \text{ is SOS}, \end{array} \tag{8.18}$$

where $P(x)$ is an $r \times r$ polynomial matrix with an "arrow" sparsity pattern defined as

$$P(x) = \begin{bmatrix} p_1(x) & p_2(x) & \cdots & p_2(x) \\ p_2(x) & p_3(x) & & \\ \vdots & & \ddots & \\ p_2(x) & & & p_3(x) \end{bmatrix} \tag{8.19}$$

with

$$p_1(x) = r(x_1^2 + x_2^2 + 1), \quad p_2(x) = x_1 + x_2, \quad p_3(x) = x_1^2 + x_2^2 + 1.$$

Problem (8.18) provides a lower bound $-\gamma$ for the minimum eigenvalue of $P(x)$ uniformly in x. Theorem 8.2 applies because the graph representing the sparsity pattern of $P(x)$ in (8.19) is chordal with its maximal cliques $\mathcal{C}_k = \{1, k\}$, $k = 2, \ldots, r$.

We used YALMIP [79] and SeDuMi [71] to solve the SDP relaxations of problem (8.18) corresponding to both the usual SOS problem (8.16) and the decomposed version of (8.17). SeDuMi was called with its default parameters on a PC with a 2.8 GHz Intel® Core™ i7 CPU and 8GB of RAM. Table 8.1 lists the CPU time required. Clearly, the computational time was reduced significantly when using Theorem 8.2 to decompose (8.17). This is not surprising because a single large SOS constraint of dimension r has been replaced by $r - 1$ smaller SOS constraints on 2×2 polynomial matrices. Table 8.2 shows that using the stronger condition $P \in \widetilde{\Sigma}^r_{n,2d}(\mathcal{E}, 0)$ brings no conservatism compared to the usual SOS methods for this particular example.

However, the inclusion $\widetilde{\Sigma}^r_{n,2d}(\mathcal{E},0) \subset \Sigma^r_{n,2d}(\mathcal{E},0)$ is strict, so replacing (8.16) by (8.17) introduces a degree of conservatism in general. This is the case, for instance, when solving (8.18) for

$$P(x) = \begin{bmatrix} p_1(x) & p_2(x) & p_3(x) \\ p_2(x) & p_4(x) & 0 \\ p_3(x) & 0 & p_5(x) \end{bmatrix},$$

with

$$\begin{aligned} p_1(x) &= 0.8x_1^2 + 0.9x_1x_2 + 0.3x_2^2 + 1.4x_1 + 0.9x_2 + 0.8, \\ p_2(x) &= 0.3x_1 + 0.91x_2 + 0.2, \\ p_3(x) &= 0.1x_1 + x_2 + 0.8, \\ p_4(x) &= 0.4x_1^2 + 1.3x_1x_2 + 1.1x_2^2 + 1.4x_1 + 2.3x_2 + 1.3, \\ p_5(x) &= 0.7x_1^2 + 1.3x_1x_2 + 0.9x_2^2 + x_1 + 1.1x_2 + 0.4. \end{aligned}$$

Solving the original SOS program (8.16) returns an objective value $\gamma_1 = 2.007$, while the optimal value of (8.17) is $\gamma_2 = 2.041$. The conservatism comes from the fact that we enforce $Q_{23} = 0$ when $p_{23}(x) = 0$. Nonetheless, the formulation (8.17) provides a highly scalable way to deal with large sparse matrix-valued SOS programs, as confirmed by the wall times reported in Table 8.1.

8.6 Conclusion

In this chapter, we introduced two theorems for the decomposition and completion of sparse SOS matrices. Specifically, we proved that a subset of SOS matrices with chordal sparsity patterns can be decomposed into a sum of multiple SOS matrices of smaller dimensions. This property can be easily applied to exploit sparsity in matrix-valued SOS programs.

It should be noted that a notion of *correlative sparsity techniques* has been proposed to exploit chordal sparsity in scalar polynomials [31]. Checking whether a polynomial matrix $P(x)$ is SOS can be reduced to a scalar problem by requiring that the polynomial $y^\mathsf{T}P(x)y$ is SOS in $[x;y]$ [131]. In fact, it is not difficult to show that our decomposition result in Theorem 8.2 corresponds to applying the correlative sparsity technique [31] to the scalar polynomial $y^\mathsf{T}P(x)y$ (more details are given in Section 9.5.2 of the next chapter). Instead, the scalar interpretation of the SOS completion result (Theorem 8.4) is not clear and requires further investigation.

Finally, our preliminary numerical experiments on a simple test problem demonstrate that exploiting chordal sparsity in matrix-valued SOS programs can bring dramatic computational savings at the cost of mild conservatism.

9

Chordal decomposition in sparse SOS optimization

In this chapter, we reveal the relationship between chordal decomposition in SOS polynomials and two other recent techniques. Specifically, we investigate the relation between three tractable relaxations for optimizing over *sparse* non-negative polynomials: sparse sum-of-squares (SSOS) optimization, diagonally dominant sum-of-squares (DSOS) optimization, and scaled diagonally dominant sum-of-squares (SDSOS) optimization. We prove that the set of SSOS polynomials, an inner approximation of the cone of SOS polynomials, strictly contains the spaces of sparse DSOS/SDSOS polynomials. For problems with sparse polynomials, therefore, SSOS optimization is less conservative than its DSOS/SDSOS counterparts. Numerical results for large-scale sparse polynomial optimization problems demonstrate this fact, and also that SSOS optimization can be faster than DSOS/SDSOS methods despite requiring the solution of semidefinite programs instead of less expensive linear/second-order cone programs.

9.1 Introduction

Optimization over non-negative polynomials plays a fundamental role in analysis and control of systems with polynomial dynamics. For instance, the construction of polynomial Lyapunov or Lyapunov-type functions subject to suitable polynomial inequalities can prove nonlinear stability of equilibrium solutions [140], approximate basins of attraction [148, 149], stability analysis of partial differential equations [150, 151] and provide bounds on infinite-time averages [152–154].

As already seen in Chapter 7 and Chapter 8, since deciding whether a given polynomial $p(x)$ is non-negative is NP-hard in general, a popular alternative is to look for a decomposition of $p(x)$ as a sum-of-squares (SOS) of polynomials with lower degree. Checking this sufficient condition for non-negativity is attractive because it amounts to solving a semidefinite program (SDP) [52, 127], a well-known type of convex optimization problem for which polynomial-time algorithms exist. However, the dimension

of this SDP typically grows in a combinatorial fashion as the number of variables and the polynomial degree increase [52], and very large SDPs are required to solve even when one employs well-known dimension reduction techniques, such as the Newton polytope [129], diagonal inconsistency [130], and symmetry [131] or facial reduction [128]. Consequently, SOS-based analysis is only practical for polynomial dynamical systems with few states and/or low degree.

In order to improve scalability, it has been proposed by many authors to replace positivity certificates based on SOS representations with other sufficient conditions for non-negativity, which are stronger but have a lower computational complexity. For correlatively sparse polynomials, characterized by sparse couplings between different independent variables, Waki *et al.* [31] proposed to look for a decomposition as a sum of SOS polynomials, each involving only small subsets of the independent variables. While being more restrictive, the search for such a sparse sum-of-squares (SSOS) decomposition can be carried out at a fraction of the computational cost required for a standard SOS decomposition, because an SDP with one large matrix variable is replaced with an SDP with multiple much smaller matrix variables. The latter can be solved more efficiently, a fact that also underpins the more recent sparse-BSOS [155] and multi-ordered Lasserre relaxation hierarchies [156] for sparse polynomial optimization. Two other alternatives to SOS optimization, applicable also to polynomials without correlative sparsity, were put forward by Ahmadi and Majumdar [124], who observed that the cones of *diagonally dominant sum-of-squares* (DSOS) polynomials and of *scaled diagonally dominant sum-of-squares* (SDSOS) polynomials are strict subsets of the cone of SOS polynomials. Optimization problems over DSOS and SDSOS polynomials can be recast as linear programs (LPs) and second-order cone programs (SOCP), respectively, and both of these can be solved with algorithms that scale more favourably than those for SDPs [124]. On the other hand, DSOS/SDSOS optimization might be very conservative (although the conservatism may be reduced using iterative methods based on basis pursuit [132] or column generation [157]).

The availability of such a variety of approaches poses a simple but important dilemma: when more than one method can be applied, which one should be used? To answer this question, theoretical results comparing the degree of conservatism and computational complexity for each of the aforementioned approaches would be desirable. In this chapter, therefore, we study the relation between DSOS/SDSOS/SSOS positivity certificates for polynomials with correlative sparsity. Specifically, we prove that if a DSOS/SDSOS decomposition exists *and* the correlative sparsity is *chordal* (meaning that it can be represented by a chordal graph), then an SSOS decomposition is also available. In other words, the cones of DSOS/SDSOS polynomials with chordal correlative sparsity are strictly

contained within the cone of polynomials that admit an SSOS decomposition in the sense of Waki *et al.* [31]. Thus, for polynomials with chordal correlative sparsity, DSOS/SDSOS optimization is provably more conservative than SSOS optimization (at least when the iterative improvement techniques for DSOS/SDSOS optimization mentioned above are not utilised). Also, SSOS optimization promises better scalability compared to standard SOS optimization. Therefore, we argue that SSOS optimization is a suitable candidate to bridge the gap between DSOS/SDSOS and SOS optimization.

The rest of this chapter is organized as follows. Section 9.2 reviews some basic facts about SOS/DSOS/SDSOS polynomials, useful notions from graph theory, and correlatively sparse polynomials. We give a new interpretation of the positivity certificates of Waki *et al* [31] in Section 9.3, which enables the connection to DSOS/SDSOS conditions for correlatively sparse polynomials in Section 9.4. In Section 9.5, we extend our analysis to the cones of sparse DSOS/SDSOS/SSOS polynomial matrices. Section 9.6 presents numerical results. Finally, Section 9.7 concludes this chapter.

9.2 Preliminaries

9.2.1 SOS, DSOS, and SDSOS polynomials

For completeness, we briefly review some sum-of-squares notion again[1]. Given a vector of variables $x \in \mathbb{R}^n$ and a multi-index $\alpha \in \mathbb{N}^n$, the quantity $x^\alpha := \prod_{i=1}^n x_i^{\alpha_i}$ is a monomial of degree $|\alpha| := \sum_{i=1}^n \alpha_i$. For $d \in \mathbb{N}$, we let $\mathbb{N}^n_d = \{\alpha \in \mathbb{N}^n : |\alpha| \leq d\}$. An n-variate polynomial of degree $2d$ can be written as $p(x) = \sum_{\alpha \in \mathbb{N}^n_{2d}} c_\alpha x^\alpha$. We denote the set of polynomials in n variables with real coefficients of degree no more than $2d$ by $\mathbb{R}[x]_{n,2d}$, and the subset of nonnegative polynomials in $\mathbb{R}[x]_{n,2d}$ by $PSD_{n,2d}$.

Checking if $p(x) \in PSD_{n,2d}$ is NP-hard already for polynomials of degree 4, but it is computationally tractable to test membership to the following subsets of $PSD_{n,2d}$.

- *SOS polynomials:* A polynomial $p(x) \in \mathbb{R}[x]_{n,2d}$ is an SOS polynomial if there exist $f_i \in \mathbb{R}[x]_{n,d}$, $i = 1, \ldots, s$ such that $p(x) = \sum_{i=1}^s f_i^2(x)$. We denote the set of n-variate SOS polynomials of degree no larger than $2d$ by $SOS_{n,2d}$. It is known [52] that $p(x) \in SOS_{n,2d}$ if and only if there exists a PSD matrix Q (denoted $Q \succeq 0$), such that
$$p(x) = v_d(x)^\mathsf{T} Q\, v_d(x), \tag{9.1}$$
where $v_d(x) = [1, x_1, x_2, \ldots, x_n, x_1^2, x_1x_2, \ldots, x_n^d]^\mathsf{T}$ is the vector of monomials of x of degree d or less. Following [52], we refer to (9.1) as the *Gram matrix* representation of the polynomial $p(x)$. Note that the size of the Gram matrix Q is $\binom{n+d}{d} \times \binom{n+d}{d}$ in general.

[1] Unlike Chapter 7 and Chapter 8, we use $SOS_{n,2d}$ to denote the set of sum-of-squares polynomials throughout this chapter, to be consistent with the upcoming notation for DSOS/SDSOS polynomials.

- *DSOS polynomials:* Recall that a symmetric matrix $A \in \mathbb{S}^r$ is diagonally dominant (DD) if $A_{ii} \geq \sum_{j=1}^{r} |A_{ij}|$ for all $i = 1, \ldots, r$, and that DD matrices are positive semidefinite (this directly follows, for example, from Gershgorin's circle theorem). Following [124], we say that polynomial $p(x) \in \mathbb{R}[x]_{n,2d}$ is a *diagonally dominant sum-of-squares* (DSOS) if it admits a Gram matrix representation (9.1) with a DD Gram matrix Q. We denote the set of DSOS polynomials in n variables and degree no larger than $2d$ by $DSOS_{n,2d}$.
- *SDSOS polynomials:* Recall that a symmetric matrix $A \in \mathbb{S}^r$ is scaled diagonally dominant (SDD) if there exists a positive definite $r \times r$ diagonal matrix D such that DAD is diagonally dominant. Following [124], we say that polynomial $p(x) \in \mathbb{R}[x]_{n,2d}$ is a *scaled diagonally dominant sum-of-squares* (SDSOS) if it admits a Gram matrix representation (9.1) with an SDD Gram matrix Q. We denote the set of SDSOS polynomials in n variables and degree no larger than $2d$ by $SDSOS_{n,2d}$.

Let $p_0, \ldots, p_t \in \mathbb{R}[x]_{n,2d}$ be given polynomials. An SOS optimization problem takes the standard form

$$\begin{aligned} \min_{u} \quad & w^{\mathsf{T}} u \\ \text{subject to} \quad & p_0(x) + \sum_{i=1}^{t} u_i p_i(x) \in SOS_{n,2d}, \end{aligned} \tag{9.2}$$

where $u \in \mathbb{R}^t$ is the decision variable. It is not difficult to see that (9.1) enables one to recast (9.2) as an SDP [52]. It has also been proved that if $SOS_{n,2d}$ is replaced with $DSOS_{n,2d}$ in (9.2) (resp. $SDSOS_{n,2d}$), then one obtains an LP (resp. SOCP) [124]. In principle, when one moves from SOS optimization to SDSOS/DSOS optimization, the ability of scalability increases and the quality of solutions decreases. Thus, DSOS/SDSOS optimization are more scalable alternatives to SOS optimization, but are typically more conservative since the strict inclusion $DSOS_{n,2d} \subset SDSOS_{n,2d} \subset SOS_{n,2d}$ holds.

9.2.2 Correlatively sparse polynomials

The notion of correlative sparsity was introduced by Waki *et al.* [31] to describe couplings between the variables $x_1, \ldots, x_n$ of a polynomial $p(x) = \sum_{|\alpha| \leq 2d} c_\alpha x^\alpha$. Key to this description is the so-called *correlative sparsity matrix* (CSP matrix), a symmetric matrix $\mathrm{csp}(p) \in \mathbb{S}^n$ where

$$[\mathrm{csp}(p)]_{ij} = \begin{cases} 1, & \text{if } i = j \text{ or } \exists \alpha \mid \alpha_i, \alpha_j \geq 1 \text{ and } c_\alpha \neq 0, \\ 0, & \text{otherwise.} \end{cases}$$

For example, we have

$$\mathrm{csp}(x_1^2 + x_2 x_3^3) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}.$$

A polynomial $p(x) \in \mathbb{R}[x]_{n,2d}$ is said to have a correlative sparsity pattern characterized by an undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ if $\mathrm{csp}(p) \in \mathbb{S}^n(\mathcal{E}, 0)$. It is then natural to define the vector space of polynomials with the same correlative sparsity as

$$\mathbb{R}[x]_{n,2d}(\mathcal{E}) := \{p \in \mathbb{R}[x]_{n,2d} \mid \mathrm{csp}(p) \in \mathbb{S}^n(\mathcal{E}, 0)\},$$

and its subset of sparse SOS polynomials as

$$SOS_{n,2d}(\mathcal{E}) := \mathbb{R}[x]_{n,2d}(\mathcal{E}) \cap SOS_{n,2d}.$$

Throughout this chapter, we assume that the correlative sparsity pattern $\mathcal{E}$ is chordal, or that a suitable chordal extension has been found.

9.3 Revisiting sparse SOS decompositions

Suppose we wish to determine whether a correlatively sparse polynomial $p(x) \in \mathbb{R}[x]_{n,2d}(\mathcal{E})$ is non-negative using an SOS certificate, meaning that we seek a Gram matrix representation of the form

$$p(x) = v_d(x)^{\mathsf{T}} Q\, v_d(x), \quad Q \succeq 0. \tag{9.3}$$

Using multi-indices $\beta \in \mathbb{N}^n_d$ and $\gamma \in \mathbb{N}^n_d$ to index the entries of Q, the equality constraints in (9.3) can be rewritten as

$$p(x) = \sum_{\beta,\gamma \in \mathbb{N}^n_d} Q_{\beta,\gamma} x^{\beta+\gamma} = \sum_{\alpha \in \mathbb{N}^n_{2d}} \left(\sum_{\beta+\gamma=\alpha} Q_{\beta,\gamma} \right) x^\alpha.$$

It is clear that, even if $p(x)$ is correlatively sparse, its Gram matrix Q need not be sparse: the only requirement is that $\sum_{\beta+\gamma=\alpha} Q_{\beta,\gamma} = 0$ if $p(x)$ does not contain the monomial x^α. Nonetheless, in order to exploit correlative sparsity and reduce the cost of searching for a suitable PSD Gram matrix, we insist that Q should be sparse by imposing that $Q_{\beta,\gamma} = 0$ if the monomial $x^{\beta+\gamma}$ does not appear in any polynomial of $\mathbb{R}[x]_{n,2d}(\mathcal{E})$. More precisely, we define

$$\begin{aligned} SSOS_{n,2d}(\mathcal{E}) = \{p \in \mathbb{R}[x]_{n,2d} \mid (9.3) \text{ holds and } Q_{\beta,\gamma} = 0 \\ \text{if } \exists (i,j) \notin \mathcal{E} \text{ s.t. } \beta_i + \gamma_i \neq 0 \text{ and } \beta_j + \gamma_j \neq 0\}. \end{aligned} \tag{9.4}$$

Another method to exploit sparsity, proposed by Waki *et al.* [31], is to search for a sparse SOS (SSOS) decomposition: let $\mathcal{C}_1, \ldots, \mathcal{C}_t$ be the maximal cliques of $\mathcal{G}(\mathcal{V}, \mathcal{E})$, let $E_{\mathcal{C}_k}$ be as in (2.14) (see Section 2.3) for all $k = 1, \ldots, t$, and try to find PSD matrices $Q_1, \ldots, Q_t$ such that

$$p(x) = \sum_{k=1}^{t} v_d(E_{\mathcal{C}_k} x)^{\mathsf{T}} Q_k\, v_d(E_{\mathcal{C}_k} x). \tag{9.5}$$

In other words, one can try to write p as a sum of SOS polynomials $p_k(E_{\mathcal{C}_k}x) := v_d(E_{\mathcal{C}_k}x)^\mathsf{T} Q_k\, v_d(E_{\mathcal{C}_k}x)$, each of which depends only on the corresponding subset of variables $E_{\mathcal{C}_k}x$. Our first main result is to show that these two strategies—imposing that Q is sparse according to (9.4) and looking for the SSOS decomposition (9.5)—are equivalent.

Theorem 9.1. Let $\mathcal{G}(\mathcal{V},\mathcal{E})$ be a chordal graph with maximal cliques $\{\mathcal{C}_1,\mathcal{C}_2,\ldots,\mathcal{C}_t\}$. Then,

$$p(x) \in SSOS_{n,2d}(\mathcal{E}) \Leftrightarrow p(x) = \sum_{k=1}^{t} p_k(E_{\mathcal{C}_k}x), \tag{9.6}$$

where $p_k(E_{\mathcal{C}_k}x)$ is an SOS polynomial in the subset of variables $E_{\mathcal{C}_k}x$.

Proof. To prove the $\Rightarrow$ part, we show that the Gram matrix Q has a chordal pattern when (9.4) holds, so the chordal decomposition (Theorem 2.10) can be applied to recover (9.5). The $\Leftarrow$ part will also follows from this.

$\Rightarrow$ If β and γ are such that an entry $Q_{\beta,\gamma}$ is not required to vanish due to (9.4), then $(i,j) \in \mathcal{E}$ for all i,j such that $\beta_i + \gamma_i \neq 0$ or $\beta_j + \gamma_j \neq 0$. Consequently, the set $\mathcal{C}_{\beta,\gamma} = \{i \in \mathcal{V} \mid \beta_i + \gamma_i \neq 0\}$ is a clique of the graph $\mathcal{G}(\mathcal{V},\mathcal{E})$ and is therefore contained in one of its maximal cliques $\mathcal{C}_1, \ldots, \mathcal{C}_t$. In other words, $\mathcal{C}_{\beta,\gamma} \subseteq \mathcal{C}_k$ for some $k \in \{1,\ldots,t\}$. Clearly, this also implies that

$$\{i \in \mathcal{V} \mid \beta_i \neq 0\} \subseteq \mathcal{C}_k, \quad \{i \in \mathcal{V} \mid \gamma_i \neq 0\} \subseteq \mathcal{C}_k.$$

Thus, if β and γ do not satisfy the condition in (9.4), then there exists a value $k \in \{1,\ldots,t\}$ such that

$$\beta,\gamma \in \mathcal{C}_k^d := \{\alpha \in \mathbb{N}_d^n \mid \alpha_i \neq 0 \implies i \in \mathcal{C}_k\}. \tag{9.7}$$

At this stage, define a hyper-graph $\mathcal{G}^d(\mathcal{V}^d,\mathcal{E}^d)$ with the multi-indices $\mathcal{V}^d = \{\alpha \in \mathbb{N}_d^n \mid x^\alpha \in v_d(x)\}$ as nodes and

$$\mathcal{E}^d = \bigcup_{k=1}^{t} \mathcal{C}_k^d \times \mathcal{C}_k^d \tag{9.8}$$

as edges. Since $\mathcal{C}_1,\ldots,\mathcal{C}_t$ are the maximal cliques of the chordal graph $\mathcal{G}(\mathcal{V},\mathcal{E})$, it can be shown [41] that $\mathcal{G}^d(\mathcal{V}^d,\mathcal{E}^d)$ is also chordal, and that $\mathcal{C}_1^d,\ldots,\mathcal{C}_t^d$ are its maximal cliques. Moreover, given that condition (9.7) holds for some $k \in \{1,\ldots,t\}$ for each pair (β,γ) such that $Q_{\beta,\gamma}$ is not required to vanish by (9.4), the hyper-graph $\mathcal{G}^d(\mathcal{V}^d,\mathcal{E}^d)$ characterizes the sparsity pattern of the PSD matrix Q in (9.3), *i.e.*, $Q \in \mathbb{S}_+^N(\mathcal{E}^d,0)$ with $N = \binom{n+d}{d}$. Thus, according to Theorem 2.10, Q in (9.3) can be decomposed as

$$Q = \sum_{k=1}^{t} E_{\mathcal{C}_k^d}^\mathsf{T} Q_k E_{\mathcal{C}_k^d}, \tag{9.9}$$

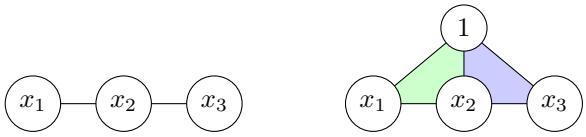


Figure 9.1: Graph patterns for polynomial (9.10): (a) the correlative sparsity pattern $\mathcal{G}(\mathcal{V},\mathcal{E})$ of (9.10) is a line graph; (b) the corresponding hyper-graph $\mathcal{G}^d(\mathcal{V}^d,\mathcal{E}^d)$ is chordal with maximal cliques $\mathcal{C}_1^d = \{1, x_1, x_2\}, \mathcal{C}_2^d = \{1, x_2, x_3\}$.

where $Q_k \in \mathbb{S}_+^{|\mathcal{C}_k^d|}$ for all $k = 1, \ldots, t$. Upon noticing that $E_{\mathcal{C}_k^d} v_d(x) = v_d(E_{\mathcal{C}_k} x)$ by virtue of the definition of $\mathcal{C}_k^d$ in (9.7), we then obtain from (9.3) that

$$\begin{aligned} p(x) &= v_d(x)^\mathsf{T} \left(\sum_{k=1}^{t} E_{\mathcal{C}_k^d}^\mathsf{T} Q_k E_{\mathcal{C}_k^d} \right) v_d(x) \\ &= \sum_{k=1}^{t} (E_{\mathcal{C}_k^d} v_d(x))^\mathsf{T} Q_k (E_{\mathcal{C}_k^d} v_d(x)), \\ &= v_d(E_{\mathcal{C}_k} x)^\mathsf{T}\, Q_k\, v_d(E_{\mathcal{C}_k} x), \end{aligned}$$

which is exactly (9.5) as claimed.

$\Leftarrow$ This follows after rearranging the last set of equalities in a suitable way. ■

Note that, in fact, we have shown that $p(x) \in SSOS_{n,2d}$ if and only if it admits a sparse Gram matrix $Q \in \mathbb{S}_+^N(\mathcal{E}^d, 0)$, so searching for an SSOS decomposition (9.5) amounts to imposing a sparsity constraint on the Gram matrix.

Example 9.2. Consider the quadratic polynomial

$$\begin{aligned} p(x) &= 2(2 + x_1 + x_2 + x_3 + x_1^2 + x_1x_2 + 2x_2^2 + x_2x_3 + x_3^2) \\ &= \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}^\mathsf{T} \underbrace{\begin{bmatrix} 4 & 1 & 1 & 1 \\ 1 & 2 & 1 & 0 \\ 1 & 1 & 4 & 1 \\ 1 & 0 & 1 & 2 \end{bmatrix}}_{Q \succeq 0} \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}, \end{aligned} \tag{9.10}$$

where $\mathrm{csp}(p)$ is the line graph with maximal cliques $\mathcal{C}_1 = \{1, 2\}$ and $\mathcal{C}_2 = \{2, 3\}$ shown in Figure 9.1. The Gram matrix Q (unique in this case) is PSD and its sparsity pattern, represented by the chordal graph in Figure 9.1(b) with maximal cliques $\mathcal{K}_1 = \{1, x_1, x_2\}$ and $\mathcal{K}_2 = \{1, x_2, x_3\}$, satisfies the condition in (9.4). According to Theorem 2.10, Q can be written as the sum of two PSD matrices supported on $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively, for instance

$$Q = \underbrace{\begin{bmatrix} 2 & 1 & 1 & 0 \\ 1 & 2 & 1 & 0 \\ 1 & 1 & 3 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}}_{\succeq 0} + \underbrace{\begin{bmatrix} 2 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 2 \end{bmatrix}}_{\succeq 0}.$$

Thus, the 3-variate polynomial $p(x)$ can clearly be written as the sum of the two bi-variate SOS polynomials

$$p_1(x_1, x_2) = (1 + x_1)^2 + (x_1 + x_2)^2 + (1 + x_2)^2 + x_2^2,$$
$$p_2(x_2, x_3) = (1 + x_3)^2 + (x_2 + x_3)^2 + 1.$$

9.4 Relating SSOS to sparse DSOS/SDSOS

We have seen that finding an SSOS decomposition of a correlatively sparse polynomial $p \in \mathbb{R}[x]_{n,2d}(\mathcal{E})$ amounts to constraining the sparsity of its Gram matrix. This fact makes it possible to draw a connection between the space $SSOS_{n,2d}(\mathcal{E})$ of sparse SOS polynomials and those of sparse DSOS/SDSOS polynomials, defined as

$$DSOS_{n,2d}(\mathcal{E}) := DSOS_{n,2d} \cap \mathbb{R}[x]_{n,2d}(\mathcal{E}),$$
$$SDSOS_{n,2d}(\mathcal{E}) := SDSOS_{n,2d} \cap \mathbb{R}[x]_{n,2d}(\mathcal{E}).$$

Specifically, we have the following result.

Proposition 9.3. For any sparsity pattern $\mathcal{E}$

$$DSOS_{n,2d}(\mathcal{E}) \subset SDSOS_{n,2d}(\mathcal{E}) \subset SSOS_{n,2d}(\mathcal{E}) \subseteq SOS_{n,2d}(\mathcal{E}), \tag{9.11}$$

and the first two inclusions are strict. The third inclusion is strict unless $\mathcal{E}$ is full or $d = 1$, in which cases $SSOS_{n,2d}(\mathcal{E}) \equiv SOS_{n,2d}(\mathcal{E})$.

Proof. We only need to prove that $SDSOS_{n,2d}(\mathcal{E}) \subset SSOS_{n,2d}(\mathcal{E})$; the rest is true by definition, and the identity $SSOS_{n,2}(\mathcal{E}) \equiv SOS_{n,2}(\mathcal{E})$ holds because the Gram matrix representation of correlatively sparse quadratic polynomials is unique and must be sparse. Recall that any $p \in SDSOS_{n,2d}(\mathcal{E})$ can be represented by an SDD Gram matrix Q, not necessarily sparse. We then construct a sparse Gram matrix $\hat{Q}$ according to

$$\hat{Q}_{\beta,\gamma} = \begin{cases} 0 & \text{if } \exists (i,j) \notin \mathcal{E}, \beta_i + \gamma_i \neq 0, \beta_j + \gamma_j \neq 0, \\ Q_{\beta,\gamma} & \text{otherwise.} \end{cases}$$

It is not difficult to see that $p(x) = v_d(x)\hat{Q}v_d(x)$, and that $\hat{Q}$ is also SDD. Indeed, replacing any off-diagonal entries with zeros does not affect the scaled diagonal dominance of a matrix, while if a diagonal entry $Q_{\beta,\beta}$ is replaced by zero, then there exists $(i,j) \notin \mathcal{E}$ such that $\beta_i \neq 0$ and $\beta_j \neq 0$, and so the entire row $Q_{\beta,\bullet}$ and column $Q_{\bullet,\beta}$ are also replaced with zeros. Thus, $\hat{Q}$ satisfies the condition (9.4) and $p(x) \in SSOS_{n,2d}(\mathcal{E})$. Finally, the inclusion $SDSOS_{n,2d}(\mathcal{E}) \subset SSOS_{n,2d}(\mathcal{E})$ is strict because there exist SSOS polynomials whose Gram matrix is not SDD. ∎

Table 9.1: Details of problem types for SOS, SSOS, SDSOS, and SOS optimization with degree $2d$ polynomials in n variables. The value m is the size of the largest clique of the underlying correlative sparsity graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; for many problem instances, $m \ll n$.

Problem	Cone	Program	Maximum PSD cone size
$\mathcal{P}_{\text{sos}}$	$SOS_{n,2d}(\mathcal{E})$	SDP	$\binom{n+d}{d}$
$\mathcal{P}_{\text{ssos}}$	$SSOS_{n,2d}(\mathcal{E})$	SDP	$\binom{m+d}{d}$
$\mathcal{P}_{\text{sdsos}}$	$SDSOS_{n,2d}(\mathcal{E})$	SOCP	2
$\mathcal{P}_{\text{dsos}}$	$DSOS_{n,2d}(\mathcal{E})$	LP	1

The implication of Proposition 9.3 is simple but important: $SSOS_{n,2d}(\mathcal{E})$ is a strictly better inner approximation of $SOS_{n,2d}(\mathcal{E})$, compared to the DSOS/SDSOS counterparts. Consequently, given correlatively sparse polynomials $p_0, \ldots, p_t \in \mathbb{R}[x]_{n,2d}(\mathcal{E})$ and an optimization variable $u \in \mathbb{R}^t$, the SOS optimization problem

$$\begin{aligned} f^*_{\text{sos}} := \underset{u}{\text{minimize}} \quad & w^{\mathsf{T}} u \\ \text{subject to} \quad & p_0(x) + \sum_{i=1}^{t} u_i p_i(x) \in SOS_{n,2d}(\mathcal{E}), \end{aligned} \tag{$\mathcal{P}_{\text{sos}}$}$$

is better approximated if the cone $SOS_{n,2d}(\mathcal{E})$ is replaced by $SSOS_{n,2d}(\mathcal{E})$ instead of $SDSOS_{n,2d}(\mathcal{E})$ or $DSOS_{n,2d}(\mathcal{E})$. Specifically, if we denote the optimization problems arising in each of these cases by $\mathcal{P}_{\text{ssos}}$, $\mathcal{P}_{\text{sdsos}}$, and $\mathcal{P}_{\text{dsos}}$, and let f^*_{ssos}, f^*_{sdsos}, and f^*_{dsos} be their respective optimal values[2], Proposition 9.3 implies that

$$f^*_{\text{dsos}} \geq f^*_{\text{sdsos}} \geq f^*_{\text{ssos}} \geq f^*_{\text{sos}}. \tag{9.12}$$

Additionally, it should be clear from Theorem 9.1 that the SSOS optimization problem $\mathcal{P}_{\text{ssos}}$ can be recast as an SDP with multiple PSD matrix variables whose size is bounded by $\binom{m+d}{d}$, where m is the size of the largest clique of the underlying correlative sparsity graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$. Therefore, even though problems $\mathcal{P}_{\text{dsos}}$ and $\mathcal{P}_{\text{sdsos}}$ can be solved as LPs and SOCPs [124], the added representation power offered by SSOS constraints need not add much computational cost when $m \ll n$. Table 9.1 summarizes the problem types for SOS, SSOS, SDSOS, and SOS optimization. In fact, as will be demonstrated by the numerical examples in Section 9.6, SSOS optimization can be much faster than DSOS/SDSOS optimization provided by the package SPOTless [158]. Seen in this light, *SSOS optimization bridges the gap between DSOS/SDSOS and SOS optimization for problems with correlatively sparse polynomials.*

[2]For an infeasible problem, we denote the optimal cost value as infinity.

9.5 Extension to sparse matrix-valued polynomials

The results of the previous sections can be extended to sparse matrix-valued polynomials, which arise naturally in some applications [142, 147]. Let $\mathbb{R}[x]^{r\times s}_{n,2d}$ be the space of $r \times s$ matrices whose entries are polynomials in $\mathbb{R}[x]_{n,2d}$, and $\mathbb{S}[x]^{r}_{n,2d}$ be the space of $r \times r$ symmetric polynomial matrices. A symmetric polynomial matrix $P(x) \in \mathbb{S}[x]^{r}_{n,2d}$ is PSD if $P(x) \succeq 0, \forall\, x \in \mathbb{R}^n$, and it belongs to the space $SOS^{r}_{n,2d}$ of SOS matrices if there exists $M \in \mathbb{R}[x]^{s\times r}_{n,d}$ such that $P(x) = M^{\mathsf{T}}(x)M(x)$ [131, 142, 143].

Clearly, SOS matrices are PSD. As stated in Lemma 7.5, it is well-known that $P(x) \in SOS^{r}_{n,2d}$ if and only if there exists a Gram matrix $Q \succeq 0$ such that

$$P(x) = (I_r \otimes v_d(x))^{\mathsf{T}}\, Q\, (I_r \otimes v_d(x))\,, \tag{9.13}$$

where I_r is the $r \times r$ identity matrix and $\otimes$ is the usual Kronecker product. Similarly to DSOS/SDSOS polynomials, we can define DSOS/SDSOS matrices as follows.

Definition 9.4. A polynomial matrix $P \in \mathbb{S}[x]^{r}_{n,2d}$ is

- DSOS, denoted $P \in DSOS^{r}_{n,2d}$, if it admits a Gram matrix representation (9.13) with a DD matrix Q.
- SDSOS, denoted $P \in SDSOS^{r}_{n,2d}$, if it admits a Gram matrix representation (9.13) with an SDD matrix Q.

An alternative characterization of SOS/DSOS/SDSOS matrices can be given as follows [131].

Proposition 9.5. A polynomial matrix $P \in \mathbb{S}[x]^{r}_{n,2d}$ is SOS (resp., DSOS or SDSOS) if and only if, given $y \in \mathbb{R}^r$, the polynomial $y^{\mathsf{T}}P(x)y$ is SOS (resp. DSOS or SDSOS) in $[x; y] \in \mathbb{R}^{m+n}$.

Proof. Using (9.13),

$$\begin{aligned} y^{\mathsf{T}}P(x)y &= y^{\mathsf{T}}\,(I_r \otimes v_d(x))^{\mathsf{T}}\, Q\, (I_r \otimes v_d(x))\, y \\ &= (I_r \otimes v_d(x) \cdot y \otimes 1)^{\mathsf{T}}\, Q\, (I_r \otimes v_d(x) \cdot y \otimes 1) \\ &= (y \otimes v_d(x))^{\mathsf{T}}\, Q\, (y \otimes v_d(x)) \\ &= z(x,y)^{\mathsf{T}} Q z(x,y), \end{aligned}$$

where $z(x, y) = y \otimes v_d(x)$ is a subset of the vector of monomials in x and y. Therefore, $P(x) \in SSOS^{r}_{n,2d}$ (resp. $P(x) \in DSOS^{r}_{n,2d}$ and $P(x) \in SDSOS^{r}_{n,2d}$) if and only if Q is PSD (resp., DD and SDD). ∎

9.5.1 Sparse SOS, SDSOS, and DSOS matrices

Similar to the spaces of sparse matrices mentioned in Section 9.2, we can define the space of sparse symmetric polynomial matrices whose sparsity pattern is characterized by an undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ as

$$\mathbb{S}^r_{n,2d}(\mathcal{E}, 0) := \left\{ P \in \mathbb{S}[x]^r_{n,2d} \mid (i,j) \notin \mathcal{E}^* \Rightarrow P_{ij}(x) = P_{ji}(x) = 0 \right\}.$$

We can also introduce the subspaces of sparse SOS/SDSOS/DSOS matrices,

$$\begin{aligned} SOS^r_{n,2d}(\mathcal{E}) &:= SOS^r_{n,2d} \cap \mathbb{S}^{r\times r}_{n,2d}(\mathcal{E}, 0), \\ DSOS^r_{n,2d}(\mathcal{E}) &:= DSOS^r_{n,2d} \cap \mathbb{S}^{r\times r}_{n,2d}(\mathcal{E}, 0), \\ SDSOS^r_{n,2d}(\mathcal{E}) &:= SDSOS^r_{n,2d} \cap \mathbb{S}^{r\times r}_{n,2d}(\mathcal{E}, 0). \end{aligned} \tag{9.14}$$

The Gram matrix representation (9.13) of a sparse SOS matrix $P \in SOS^r_{n,2d}(\mathcal{E})$ can be rewritten as

$$P(x) = \begin{bmatrix} v_d(x)^\mathsf{T} Q_{11} v_d(x) & \dots & v_d(x)^\mathsf{T} Q_{1r} v_d(x) \\ \vdots & \ddots & \vdots \\ v_d(x)^\mathsf{T} Q_{r1} v_d(x) & \dots & v_d(x)^\mathsf{T} Q_{rr} v_d(x) \end{bmatrix},$$

where the (i,j)-th block $Q_{ij} \in \mathbb{S}^N$ of the Gram matrix is to be chosen such that $Q \succeq 0$ and

$$v_d(x)^\mathsf{T} Q_{ij} v_d(x) = p_{ij}(x) = 0 \text{ if } (i,j) \notin \mathcal{E}^*. \tag{9.15}$$

Note that Q_{ij} need not be a zero matrix to satisfy (9.15), so the Gram matrix Q for a sparse SOS matrix is dense in general and checking that $P \in SOS^r_{n,2d}(\mathcal{E})$ can be computationally expensive. For this reason, in Chapter 8, we proposed to impose sparsity in the Gram matrix Q and test if P belongs to the space of polynomial matrices that admit a sparse SOS decomposition, defined as

$$\begin{aligned} SSOS^r_{n,2d}(\mathcal{E}) := \Big\{ P \in SOS^r_{n,2d}(\mathcal{E}) \mid P(x) \text{ admits a Gram} \\ \text{matrix } Q \succeq 0 \text{ with } Q_{ij} = 0 \text{ when } p_{ij}(x) = 0 \big\}. \end{aligned} \tag{9.16}$$

The following proposition shows that this space is larger than both $DSOS^r_{n,2d}(\mathcal{E})$ and $SDSOS^r_{n,2d}(\mathcal{E})$, and is the matrix-valued analogue of Proposition 9.3.

Proposition 9.6. For any pattern $\mathcal{E}$, we have

$$DSOS^r_{n,2d}(\mathcal{E}) \subset SDSOS^r_{n,2d}(\mathcal{E}) \subset SSOS^r_{n,2d}(\mathcal{E}) \subseteq SOS^r_{n,2d}(\mathcal{E}). \tag{9.17}$$

and the first two inclusions are strict. The third inclusion is strict unless $\mathcal{E}$ is full in which $SSOS^r_{n,2d}(\mathcal{E}) \equiv SOS^r_{n,2d}(\mathcal{E})$.

Proof. We only need to prove that

$$SDSOS^r_{n,2d}(\mathcal{E},0) \subset SSOS^r_{n,2d}(\mathcal{E},0), \tag{9.18}$$

since the other inclusions follow directly from the definition of each space. To this end, note that any $P \in SDSOS^r_{n,2d}(\mathcal{E})$ admits a Gram matrix representation (9.13) with an SDD matrix Q. Then, consider the matrix

$$\hat{Q}_{ij} = \begin{cases} Q_{ij}, & \text{if } (i,j) \in \mathcal{E}^*, \\ 0 & \text{if } (i,j) \notin \mathcal{E}^*, \end{cases}$$

obtained by setting to zero blocks of Q corresponding to zero entries of P. The matrix $\hat{Q}$ is still SDD, and hence PSD, and satisfies

$$P(x) = (I_r \otimes v_d(x))^\mathsf{T} \hat{Q} (I_r \otimes v_d(x)).$$

Hence, $P(x) \in SSOS^r_{n,2d}(\mathcal{E})$ and (9.18) is true; the inclusion is strict because there clearly exist polynomial matrices in $SSOS_{n,2d}(\mathcal{E})$ whose Gram matrix is not SDD. Finally, the identity $SSOS^r_{n,2d}(\mathcal{E}) \equiv SOS^r_{n,2d}(\mathcal{E})$ holds obviously when $\mathcal{E}$ contains all edges. ■

As in the scalar case, sparse matrix SOS certificates are expected to be less conservative than their DSOS/SDSOS counterparts. Additionally, if the sparsity pattern of a sparse polynomial matrix is chordal, then working with $SSOS_{n,2d}(\mathcal{E})$ can be computationally efficient because—similar to the SSOS decomposition for scalar polynomials—it requires solving SDPs with small PSD cones. This follows from the next theorem, originally proven in [46] (see, also, Chapter 8 of this thesis), which extends Theorem 2.10 to sparse polynomial matrices.

Theorem 9.7. Let $\mathcal{G}(\mathcal{V},\mathcal{E})$ be a chordal graph with maximal cliques $\{\mathcal{C}_1,\mathcal{C}_2,\ldots,\mathcal{C}_t\}$. Then,

$$P \in SSOS^r_{n,2d}(\mathcal{E}) \Leftrightarrow P(x) = \sum_{k=1}^{t} E_{\mathcal{C}_k}^\mathsf{T} P_k(x) E_{\mathcal{C}_k} \tag{9.19}$$

with $P_k \in SOS^{|\mathcal{C}_k|}_{n,2d}$ for each $k = 1,\ldots,t$.

Proof. A detailed proof can be found in Theorem 8.2 (Section 8.3 of this thesis). Briefly, the "if" part is obvious, while the "only if" part relies on the fact that, when $\mathcal{G}(\mathcal{V},\mathcal{E})$ is chordal, the sparse Gram matrix Q of P has a chordal sparsity pattern also, and can be decomposed using Theorem 2.10 to obtain (9.19). ■

Consequently, one can use the cones $DSOS^r_{n,2d}(\mathcal{E})$, $SDSOS^r_{n,2d}(\mathcal{E})$, and $SSOS^r_{n,2d}(\mathcal{E})$ to formulate increasingly better and computationally tractable approximations of large-scale matrix-valued SOS optimization problems of the form

$$\begin{aligned} \underset{u}{\text{minimize}} \quad & w^{\mathsf{T}}u \\ \text{subject to} \quad & P_0(x) + \sum_{i=1}^{t} u_i P_i(x) \in SOS^r_{n,2d}(\mathcal{E}), \end{aligned} \tag{9.20}$$

where $P_0, \ldots, P_t \in \mathbb{S}^r_{n,2d}(\mathcal{E}, 0)$ are given sparse symmetric polynomial matrices and $u \in \mathbb{R}^t$ is the decision variable. All comments given at the end of Section 9.4 on the relation between scalar SSOS/SDSOS/DSOS optimization are also valid for matrix-valued problems.

9.5.2 Reduction to the scalar analysis

Proposition 9.5 states that a matrix-valued polynomial $P(x)$ is SOS (resp., DSOS or SDSOS) if the associated scalar polynomial $p(x, y) = y^{\mathsf{T}}P(x)y$ is so, and it is natural to ask if the same is true for the SSOS condition. Here we show that applying Theorem 9.7 to $P \in SSOS^r_{n,2d}(\mathcal{E})$ is indeed equivalent to applying Theorem 9.1 to $p(x, y)$, but only if any correlative sparsity with respect to the variable x is ignored.

Indeed, $p(x, y)$ has correlative sparsity pattern $\mathcal{E}$ with respect to y since the monomial $y_i y_j$ appears if and only if $P_{ij}(x) \neq 0$, meaning that $(i, j) \in \mathcal{E}$. It is then not difficult to verify that, if any correlative sparsity with respect to the variable x is ignored, Theorem 9.1 guarantees that $p(x, y)$ can be decomposed as

$$p(x, y) = \sum_{k=1}^{t} p_k(x, E_{\mathcal{C}_k} y) \tag{9.21}$$

for some SOS polynomials p_k, $k = 1, \ldots, t$. In particular, each p_k is quadratic in y and has degree d in x, so it admits the Gram matrix representation

$$p_k(x, E_{\mathcal{C}_k} y) = [E_{\mathcal{C}_k} y \otimes v_d(x)]^{\mathsf{T}} Q_k [E_{\mathcal{C}_k} y \otimes v_d(x)]$$

with $Q_k \succeq 0$. But then, upon defining

$$V_{d,k} := I_{|\mathcal{C}_k|} \otimes v_d(x)$$

and using the properties of the Kronecker product to rewrite $E_{\mathcal{C}_k} y \otimes v_d(x) = V_{d,k} E_{\mathcal{C}_k} y$, we

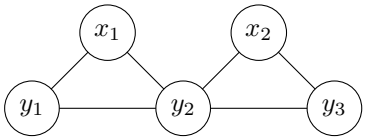


Figure 9.2: Graph pattern for polynomial $y^\mathsf{T} P y$ in (9.24).

see that

$$\begin{aligned} p(x,y) &= \sum_{k=1}^{t} p_k(x, E_{\mathcal{C}_k} y) \\ &= \sum_{k=1}^{t} [E_{\mathcal{C}_k} y \otimes v_d(x)]^\mathsf{T} Q_k [E_{\mathcal{C}_k} y \otimes v_d(x)] \\ &= \sum_{k=1}^{t} (E_{\mathcal{C}_k} y)^\mathsf{T} \underbrace{V_{d,k}^\mathsf{T} Q_k V_{d,k}}_{=:P_k(x)} E_{\mathcal{C}_k} y \\ &= y^\mathsf{T} \left(\sum_{k=1}^{t} E_{\mathcal{C}_k}^\mathsf{T} P_k(x) E_{\mathcal{C}_k} \right) y. \end{aligned} \tag{9.22}$$

Since $p(x,y) = y^\mathsf{T} P(x) y$ we conclude that

$$P(x) = \sum_{k=1}^{t} E_{\mathcal{C}_k}^\mathsf{T} P_k(x) E_{\mathcal{C}_k}, \tag{9.23}$$

which is exactly the statement of Theorem 9.7. The argument can easily be reversed to show that Theorem 9.7 implies the existence of an SSOS decomposition of $p(x,y)$.

Remark 9.8. It is important to note that the equivalence outlined above holds *only* when any correlative sparsity of the entries of the polynomial matrix P with respect to x is disregarded, because we do not take it into account in our analysis of polynomial matrices. However, it may be possible to exploit correlative sparsity in x when applying Theorem 9.1 to $y^\mathsf{T} P(x) y$. In this case, working at the scalar level will not be equivalent to applying Theorem 9.7 to P directly, and will instead result in a stronger (but possibly computationally cheaper) constraint. Therefore, our results for matrix-valued polynomials remain of independent interest. Consider the following example,

$$P(x) = \begin{bmatrix} x_1^2 + 1 & x_1 & 0 \\ x_1 & x_1^2 + x_2^2 + 2 & x_2 \\ 0 & x_2 & x_2^2 + 1 \end{bmatrix}. \tag{9.24}$$

Applying Theorem 9.7 leads to a decomposition of the from $y^\mathsf{T} P(x) y = p_1(y_1, y_2, x_1, x_2) + p_2(y_2, y_3, x_1, x_2)$, where the correlative sparsity in x_1, x_2 is ignored. Instead, the correlative sparsity pattern of $y^\mathsf{T} P(x) y$ is shown in Figure 9.2, and applying Theorem 9.1 would lead to a decomposition of the form $y^\mathsf{T} P(x) y = \hat{p}_1(y_1, y_2, x_1) + \hat{p}_2(y_2, y_3, x_2)$.

Table 9.2: Optimal γ for the SOS/SSOS/SDSOS/DSOS relaxations of problem (9.25), as a function of the number of variables n.

Dimension n	10	15	20	30	40	50
$\mathcal{P}_{\text{sos}}$	0.00	0.00	0.00	*	*	*
$\mathcal{P}_{\text{ssos}}$	0.00	0.00	0.00	0.00	0.00	0.00
$\mathcal{P}_{\text{sdsos}}$	44.7	46.0	46.6	47.2	44.4	47.5
$\mathcal{P}_{\text{dsos}}$	**	**	**	**	**	**

*: Out of memory. **: Infeasible program.

9.6 Numerical examples

To demonstrate that DSOS/SDSOS constraints are indeed more conservative than sparse SOS conditions in practice, we report the results of numerical experiments on sparse versions of the examples considered in [124]. We implemented sparse SOS conditions in YALMIP [79], adapting the undocumented option `sos.csp` to exploit correlative sparsity using the chordal extension methods described in [31]. For the DSOS/SDSOS constraints, instead, we used SPOTless [158]. The solver MOSEK [139] was used to solve the LPs, SOCPs, and SDPs arising, respectively, from DSOS, SDSOS and SSOS constraints. All computations were carried out on a PC with a 2.8 GHz Intel Core i7 CPU and 8GB of RAM.

9.6.1 Lower bounds on scalar polynomials

Given the Broyden tridiagonal polynomial

$$p(x) = ((3-2x_1)x_1 - 2x_2 + 1)^2 + \sum_{i=2}^{n-1}((3-2x_i)x_i - x_{i-1} - 2x_{i+1} + 1)^2 + ((3-2x_n)x_n - x_{n-1} + 1)^2,$$

consider the best lower bound problem

$$\begin{array}{ll} \underset{\gamma}{\text{minimize}} & \gamma \\ \text{subject to} & p(x) + \gamma x^{\mathsf{T}} x \geq 0 \quad \forall x \in \mathbb{R}^n. \end{array} \tag{9.25}$$

Upon replacing the non-negativity constraint with an SOS/SSOS/SDSOS/DSOS conditions, this problem can be reformulated as an SDP/SDP/SOCP/LP, respectively. The optimal γ obtained in each case for different values of n is reported in Table 9.2, and MOSEK's runtime is reported in Table 9.3. For all values of n the cone of DSOS polynomials is too restrictive and the DSOS constraint is infeasible. Moreover, as expected from Proposition 9.3, the SDSOS condition is more conservative that the SSOS one[3]. For this example, SSOS conditions appear not to introduce any conservativeness: they

[3]For this and all other problems solved in this chapter, the methods of [132, 157] are likely to improve the optimal objective value compared to the basic SDSOS method used here, but add computational cost.

Table 9.3: CPU time, in seconds, required by MOSEK to solve the SOS/SSOS/SDSOS/DSOS relaxations of problem (9.25), as a function of the number of variables n.

Dimension n	10	15	20	30	40	50
$\mathcal{P}_{\text{sos}}$	1.26	22.21	326.8	*	*	*
$\mathcal{P}_{\text{ssos}}$	0.48	0.47	0.48	0.63	0.54	0.53
$\mathcal{P}_{\text{sdsos}}$	0.69	1.80	4.96	25.47	88.50	232.78
$\mathcal{P}_{\text{dsos}}$	**	**	**	**	**	**

*: Out of memory. **: Infeasible program.

Table 9.4: Optimal γ for the SOS/SSOS/SDSOS/DSOS relaxations of problem (9.26), as a function of the matrix size r.

r	30	40	50	60	70	80
$\mathcal{P}_{\text{sos}}$	5.917	4.154	21.61	10.09	7.364	10.19
$\mathcal{P}_{\text{ssos}}$	5.917	4.498	21.64	12.71	7.558	11.39
$\mathcal{P}_{\text{sdsos}}$	1 254.4	145.5	762.8	1 521.1	1 217.3	598.0
$\mathcal{P}_{\text{dsos}}$	**	**	**	**	**	**

**: Infeasible program.

yield the same optimal value as the classical SOS relaxation, and at a fraction of the computational cost. Interestingly, solving the SSOS conditions was also faster than solving SDSOS conditions. This is likely due to the fact that the SSOS condition translates to an SDP with $n-1$ PSD matrix variables of size 6×6 for this particular problem (9.25), while, the number of second-order cones required for an SDSOS constraint is $\mathcal{O}(n^2)$. Whether sparsity can be exploited in SPOTless to formulate a smaller SOCP for sparse SDSOS constraints remains an interesting open question for future work.

9.6.2 Eigenvalue bounds on matrix polynomials

Let $\mathcal{G}(\mathcal{V},\mathcal{E})$ be the 5-node star graph, and let $P \in \mathbb{S}^{r\times r}_{2,2}(\mathcal{E},0)$ be a sparse polynomial matrix whose entries are randomly generated quadratic polynomials in 2 variables. The best lower bound on the smallest eigenvalue of $P(x)$ valid for all $x \in \mathbb{R}^5$ is given by the solution of the optimization problem

$$\begin{aligned} \underset{\gamma}{\text{minimize}} \quad & \gamma \\ \text{subject to} \quad & P(x) + \gamma I \succeq 0 \quad \forall x \in \mathbb{R}^5. \end{aligned} \tag{9.26}$$

We solved this problem for $P(x)$ of increasing size r after replacing the PSD constraint with SOS, SSOS, DSOS and SDSOS conditions. The optimal γ for each case is reported in Table 9.4, while the CPU time is shown in Table 9.5. As in the previous example, SSOS conditions exhibit the best trade-off between conservativeness and computational cost.

Table 9.5: CPU time, in seconds, required by MOSEK to solve the SOS/SSOS/SDSOS/DSOS relaxations of problem (9.26), as a function of the matrix size r.

r	30	40	50	60	70	80
$\mathcal{P}_{\text{sos}}$	6.64	27.3	108.1	308.7	541.3	1 018.6
$\mathcal{P}_{\text{ssos}}$	0.35	0.35	0.33	0.32	0.32	0.33
$\mathcal{P}_{\text{sdsos}}$	1.09	1.29	2.67	3.70	4.40	6.02
$\mathcal{P}_{\text{dsos}}$	**	**	**	**	**	**

**: Infeasible program.

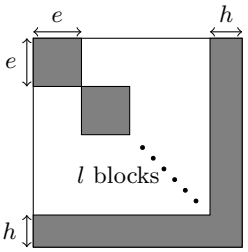


Figure 9.3: Block-arrow sparsity pattern (dots indicate repeating diagonal blocks). The parameters are: the number of blocks, l; block size, e; the width of the arrow head, h.

9.6.3 Co-positive programming

Our next experiment is an optimization problem over the cone $\mathbb{CP}^n$ of co-positive $n \times n$ matrices, which has recently attracted attention since it can model several combinatorial optimization problems exactly [159]. A symmetric matrix $Z \in \mathbb{S}^n$ is co-positive if $y^{\mathsf{T}} Z y \geq 0$ for all $y \geq 0$, or [52]

$$Z \in \mathbb{CP}^n \Leftrightarrow \sum_{i,j=1}^{n} Z_{ij} x_i^2 x_j^2 \geq 0 \quad \forall x \in \mathbb{R}^n.$$

Replacing the non-negativity constraint with SOS, SSOS, SDSOS and DSOS conditions yields tractable inner approximations of $\mathbb{CP}^n$. Here, we solve such approximations for optimization problems of the form

$$\begin{array}{ll} \underset{\gamma}{\text{minimize}} & \gamma \\ \text{subject to} & Z + \gamma I \in \mathbb{CP}^n, \end{array} \tag{9.27}$$

where Z is a randomly generated symmetric matrix with a block-arrow sparsity pattern with l blocks of size $e \times e$, and arrow head h; see Figure 9.3 for an illustration. Such a sparsity pattern is chordal, with l maximal cliques of size $e+h$. We fixed the block size $e = 3$, arrow head size $h = 2$, and varied the number of blocks l. Table 9.6 shows that the upper bound on the optimal solution of (9.27) obtained with SSOS constraints is always strictly better than that obtained with SDSOS and DSOS optimization, and gives the same result as the classical SOS relaxation in all cases for which this could be implemented. Again, SSOS constraints are also extremely competitive in terms of CPU time, cf. Table 9.7.

Table 9.6: Optimal γ for the SOS/SSOS/SDSOS/DSOS relaxations of problem (9.27) with block size $e = 3$ and arrow head size $h = 2$, as a function of the number of blocks, l.

l	2	4	6	8	10
$\mathcal{P}_{\text{sos}}$	1.137	4.197	2.836	*	*
$\mathcal{P}_{\text{ssos}}$	1.137	4.197	2.836	4.043	4.718
$\mathcal{P}_{\text{sdsos}}$	1.184	4.500	3.282	4.562	5.146
$\mathcal{P}_{\text{dsos}}$	2.551	7.775	6.452	12.057	15.203

*: Out of memory.

Table 9.7: CPU time, in seconds, required by MOSEK to solve the SOS/SSOS/SDSOS/DSOS relaxations of problem (9.27). Results are given as a function of the number of blocks, l, for block size $e = 3$ and arrow head size $h = 2$.

l	2	4	6	8	10
$\mathcal{P}_{\text{sos}}$	0.45	7.34	248.9	*	*
$\mathcal{P}_{\text{ssos}}$	0.39	0.41	0.38	0.49	0.40
$\mathcal{P}_{\text{sdsos}}$	0.54	1.22	4.99	11.07	32.18
$\mathcal{P}_{\text{dsos}}$	0.59	0.76	2.19	5.72	17.11

*: Out of memory.

9.6.4 Lyapunov stability analysis

As our final example, we considered randomly generated n-dimensional, degree-3 polynomial dynamical systems of the form

$$\begin{cases} \dot{x}_1 = f_1(x_1, x_2), \\ \dot{x}_2 = f_2(x_1, x_2, x_3), \\ \vdots \\ \dot{x}_{n-1} = f_{n-1}(x_{n-2}, x_{n-1}, x_n) \\ \dot{x}_n = f_n(x_{n-1}, x_n), \end{cases} \tag{9.28}$$

with an asymptotically stable equilibrium at the origin. In the simulation, we first generated a random $n \times n$ matrix A with a banded sparsity pattern, whose nonzero entries are drawn from the uniform distribution on the open interval $(0; 1)$ and then imposed it to be stable by setting $A := A - \alpha I$ with a proper $\alpha > 0$. The linear terms of each function f_i are chosen as $A(i, :)x$, and the coefficients of high-order (*i.e.*, degree two and degree three) terms of each function f_i is randomly generated from the open interval $(-5, 5)$. In this way, the dynamical system $f(x)$ is guaranteed to have an asymptotically stable equilibrium since the matrix A is stable.

We searched for quadratic Lyapunov functions of the form

$$V(x) = V_1(x_1, x_2) + V_2(x_1, x_2, x_3) + \ldots + V_n(x_{n-1}, x_n)$$

Table 9.8: CPU time, in seconds, required by MOSEK to construct a quadratic Lyapunov function for a locally stable, degree-3 polynomial system of the form (9.28).

n	10	15	20	30	40	50
$\mathcal{P}_{\text{sos}}$	1.36	21.26	262.08	*	*	*
$\mathcal{P}_{\text{ssos}}$	0.57	0.69	0.76	1.02	1.22	1.41
$\mathcal{P}_{\text{sdsos}}$	1.21	6.78	5.20	28.61	104.36	292.05
$\mathcal{P}_{\text{dsos}}$	0.74	1.33	2.89	14.61	61.52	275.95

*: Out of memory.

that certify the local stability of the origin in the box $\mathcal{D} = [-0.1, 0.1]^n$. Specifically, we looked for $V(x)$ that satisfies

$$\begin{aligned} V(0) &= 0, \\ V(x) &\geq \epsilon x^{\mathsf{T}} x \quad \forall x \in \mathcal{D}, \\ -f(x)^{\mathsf{T}} \nabla V(x) &\geq 0 \quad \forall x \in \mathcal{D}, \end{aligned}$$

(where we used $\epsilon = 10^{-6}$ in the simulation) after replacing the non-negativity conditions with SOS, SSOS, SDSOS, and DSOS constraints in turn. Table 9.8 lists the CPU time required by MOSEK to construct suitable Lyapunov functions in each case. The classical SOS constraints could not be implemented for $n > 20$ on our PC due to RAM limitations, while all other constraints could be implemented successfully. Although in this case all of SSOS, SDSOS and DSOS conditions are feasible, the results clearly demonstrate that SSOS are the fastest, with an approximately 200× speed improvement compared to the DSOS/SDSOS formulations set up by SPOTless when $n = 50$.

9.7 Conclusion

In this chapter, we demonstrated that, for correlatively sparse polynomials, sparse SOS positivity certificates are more general and typically less conservative than those based on DSOS and SDSOS methods. Key to this result is a new interpretation of the sparse SOS conditions proposed by Waki *et al.* [31] in terms of a sparsity constraint of the Gram matrix Q used to represent sparse SOS polynomials, to which a well known chordal decomposition theorem can be applied. Numerical examples have confirmed our theoretical findings, and also demonstrated that SSOS conditions can be dramatically more efficient than the DSOS/SDSOS conditions formulated by the dedicated package SPOTless [158]. Thus, although DSOS/SDSOS methods remain one of the few methods to implement non-negativity constraints for dense polynomials with many variables and/or high degree, one should try to utilize SSOS constraints when possible.

10

Conclusion and outlook

Many large-scale problems have inherent structures that can be exploited to facilitate their solutions. This thesis has focused on taking advantage of chordal sparsity to develop scalable methods for solving three classes of problems: large-scale sparse SDPs, distributed control of networked systems, and large-scale SOS programs. We now conclude the thesis by summarizing the main findings and discussing some future research directions.

10.1 Summary

As discussed in Section 2.2, chordal graphs possess many useful properties. In the context of sparse PSD matrices, chordal sparsity allows one to derive an equivalent decomposition of sparse PSD matrix cone (*i.e.*, Theorem 2.10 and Theorem 2.17) and a dual result on the decomposition of sparse PSD completable matrix cone (*i.e.*, Theorem 2.13 and Theorem 2.18). Overall, the main theme of this thesis has focused on exploiting the chordal decomposition idea to decompose PSD constraints that appear in some large-scale control and optimization problems, thus improving the solution efficiency.

Large-scale sparse SDPs

Part I of this thesis focused on general large-scale SDPs with chordal sparsity. Chapter 3 presented a conversion framework for this type of SDPs, which is analogous to the conversion techniques for interior-point methods of [23, 24] and is more suitable for the application of first-order methods. We then developed efficient ADMM algorithms for sparse SDPs in either primal or dual standard form, and for their homogeneous self-dual embedding. A single iteration of our ADMM algorithms only required parallel projections onto small PSD cones and a projection onto an affine subspace. When fixing the number of constraints, the complexity of each iteration is determined by the size of the largest maximal clique, rather than the size of the original problem. All our algorithms have been implemented in the open-source MATLAB package CDCS,

which has solved large sparse instances that are beyond the reach of standard interior-point and/or other first-order methods.

In Chapter 4, we demonstrated the performance of CDCS in solving standard systems analysis problems, including stability, $\mathcal{H}_2$, and $\mathcal{H}_\infty$ norms, where block-diagonal Lyapunov functions were used to maintain the sparsity pattern of the systems in the analysis problems.

Distributed control of networked systems

In Chapters 5 and 6, we applied chordal decomposition in sparse Lyapunov-type LMIs arising from structured stabilization and optimal decentralized control of networked systems, respectively. By assuming the existence of block-diagonal Lyapunov functions for the closed-loop systems, these two controller synthesis problems were converted into certain LMIs which inherited the sparsity of networked systems. Accordingly, chordal decomposition (Theorem 2.17) was used to decompose a single big LMI constraint into multiple smaller and coupled LMI constraints according to maximal cliques of an undirected graph. We discussed two different strategies to deal with the overlapping elements among different maximal cliques in Chapters 5 and 6.

Chapter 5 introduced an equal-splitting strategy to divide the coupling effects between the coupled LMI constraints, and proposed a sequential strategy to solve structured feedback gains clique-by-clique over a clique tree. In this strategy, each maximal clique solved a subproblem once and passed information along the clique tree. Consequently, this sequential strategy greatly improved the computational efficiency of solving structured feedback gains for large-scale sparse systems. Chapter 6 applied a "negotiating" process based on the framework of ADMM, where each maximal clique solved subproblems iteratively to reach consensus among overlapping variables. Only the existence of block-diagonal Lyapunov functions was required for the feasibility of the ADMM algorithm.

Large-scale SOS programs

In Part III, we focused on exploiting structures and sparsity in large-scale SOS programs. Chapter 7 revealed the structural property of partial orthogonality in the SDPs arising from general SOS programs. We used this property to develop a fast ADMM algorithm for the solution of SDPs describing SOS programs. This algorithm was implemented as a new package in the solver CDCS.

Chapters 8 and 9 considered the role of chordal sparsity patterns in SOS problems. In particular, in Chapter 8, we introduced the notion of decomposition and completion of SOS matrices with chordal sparsity, and proved that 1) a subset of sparse SOS matrices can be decomposed into a sum of SOS matrices that are nonzero only on a

principal submatrix (Theorem 8.2), and 2) a sparse polynomial matrix can be completed into an SOS matrix if certain nonzero principal submatrices satisfy suitable SOS and consistency conditions (Theorem 8.4). These two theorems provided an extension of the well-established decomposition/completion results (Theorem 2.10 and Theorem 2.13) for sparse positive semidefinite matrices to a subset of polynomial matrices.

The results in Chapter 8 motivated the topic of Chapter 9, where we built a unified viewpoint to study SSOS, DSOS and SDSOS polynomials with sparsity based on a novel graph-theoretic interpretation of their Gram matrix representation. We showed that SSOS optimization is provably less conservative than its DSOS/SDSOS counterparts, and that for polynomials with correlative sparsity patterns, SSOS optimization relies on SDPs with multiple smaller positive semidefinite cones. Our results suggested that SSOS optimization bridges the existing theoretical and computational gaps between DSOS/SDSOS and SOS optimization for sparse instances.

10.2 Future research directions

In this final section, we outline some possible directions for future research related to the work in this thesis.

Large-scale SDPs: Parallelization, accelerations, and rank constraints

In Chapter 3, we introduced efficient ADMM algorithms for SDPs with chordal sparsity. The current implementation of our algorithms is sequential, but many steps can be carried out in parallel, so further computational gains may be achieved by taking full advantage of parallel computing architectures. Besides, it would be interesting to integrate some acceleration techniques (*e.g.*, [160, 161]) that promise to improve the convergence performance of ADMM in practice.

Another interesting topic is to investigate decomposition methods for sparse SDPs with rank constraints, which have many applications [162, 163]. Indeed, the chordal decomposition and completion results (Theorem 2.10 and Theorem 2.13) can incorporate certain rank constraints. Precisely, given a chordal sparsity pattern $\mathcal{E}$ with maximal cliques $\mathcal{C}_1, \ldots, \mathcal{C}_t$, we have 1) $Z \in \mathbb{S}^n_+(\mathcal{E}, 0)$ if and only if there exist matrices $Z_k \in \mathbb{S}^{|\mathcal{C}_k|}_+$ for $k = 1, \ldots, t$ such that $Z = \sum_{k=1}^t E_{\mathcal{C}_k}^{\mathsf{T}} Z_k E_{\mathcal{C}_k}$ and $\operatorname{rank}(Z) = \sum_{k=1}^t \operatorname{rank}(Z_k)$ [22, Theorem 1]; 2) $\forall X \in \mathbb{S}^n_+(\mathcal{E}, ?)$, there exists a minimum PSD completion $\hat{X}$ such that $\operatorname{rank}(\hat{X}) = \max_k \operatorname{rank}(E_{\mathcal{C}_k} X E_{\mathcal{C}_k}^{\mathsf{T}})$ [164, Theorem 1.9]. Further work is required to investigate the applications of these decomposition results with rank constraints in SDPs (*e.g.*, when combining them with the nuclear norm approximation [163]).

Distributed control: Beyond block-diagonal Lyapunov functions

The methods developed in Chapters 4, 5, and 6 and of this thesis rely on the existence of block-diagonal Lyapunov functions (see also Appendix A for more discussions). An interesting question is to look for certain classes of networked systems where block-diagonal Lyapunov functions are necessary and sufficient for stability verification or $\mathcal{H}_2/\mathcal{H}_\infty$ performance analysis. It is known that the stability of positive systems is equivalent to the existence of diagonal Lyapunov functions [89]. One interesting question is whether one can build a "block" version of positive systems.

Another natural research topic is to investigate Lyapunov functions that are beyond block-diagonal for the distributed control of networked systems. Two important issues for this direction are: how to guarantee the structures of distributed controllers, and how to maintain the sparsity pattern of the system to facilitate chordal decomposition (*i.e.*, sparsity invariance within both controllers and optimization problems). It would be also interesting to investigate possible connections between the idea of sparsity invariance with other synthesis methods, *e.g.*, the quadratic invariance framework [5], and the system level approach [123].

SOS problems: Sparsity structure, rational polynomials, Positivstellensatz, moment interpretation

The results in Chapter 9 motivate the development of robust methods that exploit sparsity in the system's governing equations and result in sparse polynomial non-negativity conditions. In the examples of Section 9.6.4, we have done this by choosing a Lyapunov function with a structure such that the correlative sparsity of the governing equations is inherited in the eventual non-negativity constraints, but this construction was made possible by the simplicity of our example. One interesting future topic is to identify a general procedure to formulate sparse SOS conditions that is essential to enable the analysis of large-scale sparse nonlinear systems.

One main theme of this thesis is the investigation of chordal decomposition/completion results in large-scale systems. A simple case is the following decomposition ($*$ denotes a scalar or a block with suitable dimension)

$$\underbrace{\begin{bmatrix} * & * & 0 \\ * & * & * \\ 0 & * & * \end{bmatrix}}_{\succeq 0} = \underbrace{\begin{bmatrix} * & * & 0 \\ * & * & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{\succeq 0} + \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & * & * \\ 0 & * & * \end{bmatrix}}_{\succeq 0}. \tag{10.1}$$

Chapter 8 extended this decomposition to a subset of SOS matrices. One can show that the decomposition (10.1) does not hold for PSD polynomial matrices in general[1].

[1]Indeed, counterexamples can be found via convex optimization using the fact a univariate polynomial matrix is PSD if and only if it is an SOS matrix [165].

Still, the induction proof in [22, 56] suggests that this decomposition may hold when each addend is allowed to have rational polynomials.

The SOS condition in Chapter 8 is a global certificate of positive semidefiniteness of a polynomial matrix in $\mathbb{R}^n$. A very interesting topic is to restrict our attention to compact basic semi-algebraic sets $\mathbb{K} \in \mathbb{R}^n$. Recall that a basic semi-algebraic set is defined by a finite number of polynomial inequalities: $\mathbb{K} := \{x \in \mathbb{R}^n \mid g_i(x) \leq 0, i = 1, \ldots, m\}$. Then, we can investigate certificates of positive semidefiniteness on $\mathbb{K}$. This leads to the representation theory of *Positivstellensatz* (see, *e.g*, [125, Chapter 2] and [166] for excellent surveys). In particular, a sparse version of Putinar's Positivstellensatz [167] was proven in [32] and a matrix version of Putinar's Positivstellensatz was proven in [142]. An interesting question is whether one can establish a sparse-matrix version of Putinar's Positivstellensatz. We can also investigate positive semidefiniteness certificates over spectrahedra (a set defined by finitely many LMIs), especially when the LMIs have chordal sparsity.

As a final remark of this thesis, it is known that the SOS theory has a dual facet corresponding to the moment theory [125]. It would be very interesting to look into the moment interpretation of the decomposition results in Chapters 8 and 9.

Appendices

A

On block-diagonal Lyapunov functions

This appendix provides some discussions on block-diagonal Lyapunov functions and strongly decentralized stabilization.

A.1 Block-diagonal Lyapunov functions

A linear system $\dot{x}(t) = Ax(t)$ is asymptotically stable if and only if there exists a symmetric matrix $P \succ 0$ satisfying the Lyapunov LMI [10]

$$A^{\mathsf{T}}P + PA \prec 0. \tag{A.1}$$

In general, the solution P is a dense matrix, which defines a Lyapunov function of the form

$$V(x) = x^{\mathsf{T}}(t)Px(t).$$

In some applications, we are interested in a *diagonal* or *block-diagonal* solution, leading to the notion of *diagonal stability* or *block-diagonal stability* [168].

Definition A.1. A linear system is called *block-diagonally stable* if there exists a block-diagonal P satisfying (A.1), where the block sizes of P are compatible with those of the subsystems. Further, if the subsystem size is scalar, then P is diagonal and the system is called *diagonally stable*.

There exist necessary and sufficient conditions for diagonal stability of systems of dimension three or four [168]. For higher dimensional systems, conditions have been established under certain assumptions on the dynamical systems: it is known that for positive systems, the stability is equivalent to the existence of diagonal Lyapunov functions [89]; necessary and sufficient conditions were also derived for diagonal stability of special classes of interconnected systems that are widespread in biological networks, such as the secant criterion in cyclic structures [169] and its generalization on cactus graphs [170]. The conditions for block-diagonal stability were discussed in [49, 171–173].

A.2 Strongly decentralized stabilization

Here, we consider a linear system with control input, as considered in (5.1). For convenience, we restate the dynamics as

$$\dot{x}(t) = Ax(t) + Bu(t), \tag{A.2}$$

where $x(t) = [x_1(t)^\mathsf{T}, \ldots, x_N(t)^\mathsf{T}]^\mathsf{T}$ and similarly for $u(t)$.

Definition A.2 (Stabilization)**.** System (A.2) is called *stabilizable*, if there exists a centralized controller $u = -Kx$ such that the closed-loop system $\dot{x} = (A - BK)x$ is asymptotically stable.

Definition A.3 (Decentralized stabilization [115])**.** System (A.2) is called *decentralized stabilizable*, if there exists a decentralized controller $u_i = -K_{ii}x_i, i \in \mathcal{V}$ such that the closed-loop system $\dot{x} = (A - BK)x$ is asymptotically stable.

Definition A.4 (Strongly decentralized stabilization [110])**.** System (A.2) is called *strongly decentralized stabilizable* if there exists a decentralized $u_i = -K_{ii}x_i, i \in \mathcal{V}$ such that the closed-loop system $\dot{x} = (A - BK)x$ admits a block-diagonal Lyapunov function $V(x) = \sum_{i=1}^{N} x_i^T P_i x_i$.

Then, we define three classes of complex systems:

$$\Sigma_0 = \{(A, B) \mid \text{System (A.2) is stabilizable}\},$$
$$\Sigma_1 = \{(A, B) \mid \text{System (A.2) is decentralized stabilizable}\},$$
$$\Sigma_2 = \{(A, B) \mid \text{System (A.2) is strongly decentralized stabilizable}\}.$$

It is easy to see $\Sigma_2 \subseteq \Sigma_1 \subseteq \Sigma_0$. In fact, simple counterexamples can show the inclusion relationship is strict $\Sigma_2 \subset \Sigma_1 \subset \Sigma_0$. The sets Σ_0 and Σ_1 can be algebraically characterized by centralized fixed modes and decentralized fixed modes [115, 174]. Some results are available to characterize the decentralized fixed modes of a complex system [174, 175]. In this section, we discuss two classes of systems: 1) fully actuated systems, and 2) weakly coupled systems.

A.2.1 Fully actuated systems

Definition A.5 (Fully actuated systems)**.** System (A.2) is called fully actuated, if each input matrix B_i has full row rank, $i \in \mathcal{V}$.

Proposition A.6. If system (6.2) is fully actuated, then we have $(A, B) \in \Sigma_2$.

Proof. Consider the singular value decomposition of the input matrix B_i,

$$B_i = U_i \begin{bmatrix} \Gamma_i & 0 \end{bmatrix} V_i^{\mathsf{T}}, \tag{A.3}$$

where 0 is a zero block of appropriate size, and $\Gamma_i \in \mathbb{R}^{n_i \times n_i}$ is invertible since B_i has full row rank. We then consider a decentralized feedback controller

$$K_{ii} = V_i \begin{bmatrix} \Gamma_i^{-1} \\ 0 \end{bmatrix} U_i^{\mathsf{T}} (A_{ii} + \alpha_i I_{n_i}), i \in \mathcal{V}, \tag{A.4}$$

where $\alpha_i \in \mathbb{R}$. This choice leads to

$$A_{ii} - B_i K_{ii} = -\alpha_i I_{n_i}, i \in \mathcal{V}.$$

Using the decentralized controller (A.4), the closed-loop system matrix becomes

$$A - BK = \begin{bmatrix} -\alpha_1 I_{n_1} & A_{12} & \dots & A_{1N} \\ A_{21} & -\alpha_2 I_{n_2} & \dots & A_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ A_{N1} & A_{N2} & \dots & -\alpha_N I_{n_N} \end{bmatrix}. \tag{A.5}$$

By choosing an appropriate $\alpha_i > 0$, we can always make $A - BK$ diagonally dominant with negative diagonal elements. Therefore, $A - BK$ is diagonally stable, *i.e.*, there exists a diagonal Lyapunov function to certify the stability of (A.5). Therefore, we have $(A, B) \in \Sigma_2$. ■

In essence, a fully actuated system is able to actuate each individual state directly. If the dimension of each subsystem is scalar, *i.e.*, $n_i = 1$, then the condition in Proposition A.6 means that the system pair of (A_i, B_i) is controllable. For general subsystems, the condition that B_i has full rank is stronger than the controllability of (A_i, B_i).

A.2.2 Weakly coupled systems

Here, we discuss two types of weakly coupled systems: topologically weakly coupled systems and dynamically weakly coupled systems. A directed graph $\mathcal{G}$ is called *acyclic* if there exist no directed cycles in $\mathcal{G}$. A complex system with an acyclic $\mathcal{G}_p$ means that the dynamical influence among subsystems is unidirectional.

Definition A.7 (Topologically weakly coupled system)**.** System (A.2) is called weakly coupled in terms of topological connections, if the plant graph $\mathcal{G}_p$ is acyclic.

Proposition A.8. For the class of topologically weakly coupled systems, we have

$$\Sigma_1 = \Sigma_2 = \{(A, B) \mid (A_{ii}, B_i) \text{ is stabilizable}, i \in \mathcal{V}\}.$$

Proof. This result is a simple consequence of [49, 171]. If $\mathcal{G}_p$ is acyclic, then there exists an ordering of the nodes such that for every edge (v_1, v_2), node v_1 comes before node v_2 in the ordering. For this ordering, the resulting system matrix A is block lower triangular. Thus, without loss of generality, for a topologically weakly coupled system (6.2), the closed-loop system with a decentralized controller remains block lower triangular. It is known that a block triangular matrix is stable if and only if it is block-diagonally stable [49, 171], *i.e.*, there exists a block-diagonal Lyapunov function to certify the stability of the closed-loop system. Therefore, for the class of topologically weakly coupled systems, we have $(A, B) \in \Sigma_1 \Leftrightarrow (A, B) \in \Sigma_2$. Meanwhile, considering the block triangular structure, the overall closed-loop system is stable if and only if each isolated closed-loop subsystem $A_{ii} - B_i K_{ii}$ is stable, $i \in \mathcal{V}$. This completes the proof. ■

Next, we consider dynamically weakly coupled systems. If each pair (A_{ii}, B_i) is stabilizable, then there exists a local feedback K_{ii} such that $A_{ii} - B_i K_{ii}$ is stable. Consequently, given any $Q_i \succ 0$, there exists a $P_i \succ 0$, such that

$$(A_{ii} - B_i K_{ii})^\mathsf{T} P_i + P_i (A_{ii} - B_i K_{ii}) + Q_i \prec 0.$$

If the coupling term A_{ij} is element-wise small (*i.e.*, low magnitude interactions), there may still exist a solution $P_i \succ 0$ for the following inequality

$$(A_{ii} - B_i K_{ii})^\mathsf{T} P_i + P_i (A_{ii} - B_i K_{ii}) + P_i \Big(\sum_{j \in \mathbb{N}_i} A_{ij} A_{ij}^\mathsf{T} \Big) P_i + Q_i \prec 0. \tag{A.6}$$

This observation leads to a concept of dynamically weakly coupled systems.

Definition A.9 (Dynamically weakly coupled systems)**.** System (A.2) is weakly coupled in terms of dynamical interactions, if there exists a local feedback K_{ii} such that the following inequality holds

$$(A_{ii} - B_i K_{ii})^\mathsf{T} P_i + P_i (A_{ii} - B_i K_{ii}) + P_i \left(\sum_{j \in \mathbb{N}_i} A_{ij} W_{ij}^{-1} A_{ij}^\mathsf{T} \right) P_i + \sum_{j \in \hat{\mathbb{N}}_i} W_{ji} \prec 0, \tag{A.7}$$

for some $W_{ij} \succ 0, j \in \mathbb{N}_i, P_i \succ 0, i \in \mathcal{V}$, where $\hat{\mathbb{N}}_i$ denotes the set of nodes coming out of node i in $\mathcal{G}_p$.

Definition A.9 is more general than condition (A.6), since inequality (A.7) is reduced to (A.6) when setting $W_{ij} = I_{n_j}, j \in \mathbb{N}_i$, and $Q_i = \sigma_i I_{n_i}$, where σ_i denotes the number of nodes in $\hat{\mathbb{N}}_i$.

Lemma A.10. Given two matrices X, Y of appropriate dimensions, we have $X^\mathsf{T} W X + Y^\mathsf{T} W^{-1} Y \succeq X^\mathsf{T} Y + Y^\mathsf{T} X$ for any $W \succ 0$ of appropriate dimension.

Proposition A.11. For a dynamically weakly coupled system (A.2), *i.e.*, (A.7) holds, we have $(A, B) \in \Sigma_2$.

Proof. Consider a decentralized controller $K = \mathrm{diag}(K_{11}, \ldots, K_{NN})$. Upon defining $\hat{A}_{ii} = A_{ii} - B_i K_{ii}$ and ignoring the disturbance, the closed-loop dynamics for each subsystem become

$$\dot{x}_i(t) = \hat{A}_{ii} x_i(t) + \sum_{j \in \mathbb{N}_i} A_{ij} x_j(t), \quad \forall\, i \in \mathcal{V}. \tag{A.8}$$

Next, we consider a block-diagonal Lyapunov function $V(x) = \sum_{i=1}^{N} x_i^{\mathsf{T}}(t) P_i x_i(t)$. The derivative of $V(x)$ along the closed-loop trajectory (A.8) is

$$\begin{aligned} \dot{V}(x) &= \sum_{i=1}^{N} \left(\dot{x}_i^{\mathsf{T}} P_i x_i + x_i^{\mathsf{T}} P_i \dot{x}_i\right) \\ &= \sum_{i=1}^{N} \Bigg(x_i^{\mathsf{T}} \left(\hat{A}_{ii}^{\mathsf{T}} P_i + P_i \hat{A}_{ii}\right) x_i + \underbrace{\Big(\sum_{j \in \mathbb{N}_i} A_{ij} x_j\Big)^{\mathsf{T}} P_i x_i + x_i^{\mathsf{T}} P_i \Big(\sum_{j \in \mathbb{N}_i} A_{ij} x_j\Big)}_{\text{coupling term}} \Bigg). \end{aligned} \tag{A.9}$$

For the coupling term in (A.9), according to Lemma 1, we have

$$\begin{aligned} \Big(\sum_{j \in \mathbb{N}_i} A_{ij} x_j\Big)^{\mathsf{T}} P_i x_i + x_i^{\mathsf{T}} P_i \Big(\sum_{j \in \mathbb{N}_i} A_{ij} x_j\Big) &= \sum_{j \in \mathbb{N}_i} \left(x_j^{\mathsf{T}} A_{ij}^{\mathsf{T}} P_i x_i + x_i^{\mathsf{T}} P_i A_{ij} x_j\right) \\ &\leq \sum_{j \in \mathbb{N}_i} \left(x_i^{\mathsf{T}} P_i A_{ij} W_{ij}^{-1} A_{ij}^{\mathsf{T}} P_i x_i + x_j^{\mathsf{T}} W_{ij} x_j\right), \end{aligned} \tag{A.10}$$

for any $W_{ij} \succ 0, j \in \mathbb{N}_i$. Substituting (A.10) into (A.9), we get

$$\begin{aligned} \dot{V}(x) \leq & \sum_{i=1}^{N} \left(x_i^{\mathsf{T}} \big(\hat{A}_{ii}^{\mathsf{T}} P_i + P_i \hat{A}_{ii} + P_i \big(\sum_{j \in \mathbb{N}_i} A_{ij} W_{ij}^{-1} A_{ij}^{\mathsf{T}}\big) P_i\big) x_i + \sum_{j \in \mathbb{N}_i} x_j^{T} W_{ij} x_j \right) \\ = & \sum_{i=1}^{N} x_i^{\mathsf{T}} \left(\hat{A}_{ii}^{T} P_i + P_i \hat{A}_{ii} + P_i \big(\sum_{j \in \mathbb{N}_i} A_{ij} W_{ij}^{-1} A_{ij}^{\mathsf{T}}\big) P_i + \sum_{j \in \hat{\mathbb{N}}_i} W_{ji} \right) x_i. \end{aligned}$$

If condition (A.7) holds for some $W_{ij} \succ 0, j \in \mathbb{N}_i$, $P_i \succ 0, i \in \mathcal{V}$, then, $\dot{V}(x)$ is negative definite. Thus, $V(s)$ is a block-diagonal Lyapunov function for the closed-loop system. ■

References

[1] D. D. Siljak. *Decentralized control of complex systems*. Courier Corporation, 2011.

[2] Y.-Y. Liu, J.-J. Slotine, and A.-L. Barabási. "Controllability of complex networks". In: *Nature* 473.7346 (2011), p. 167.

[3] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[4] D. P. Bertsekas. *Convex optimization theory*. Athena Scientific Belmont, 2009.

[5] M. Rotkowitz and S. Lall. "A Characterization of Convex Problems in Decentralized Control". In: *IEEE Transactions on Automatic Control* 51.2 (2006), pp. 274–286.

[6] J. Lavaei and S. H. Low. "Zero duality gap in optimal power flow problem". In: *IEEE Transactions on Power Systems* 27.1 (2012), p. 92.

[7] P. A. Parrilo. "Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization". PhD thesis. California Institute of Technology, 2000.

[8] J.-B. Lasserre. *Moments, positive polynomials and their applications*. Vol. 1. World Scientific, 2010.

[9] A. Papachristodoulou and S. Prajna. "A tutorial on sum of squares techniques for systems analysis". In: *Proceedings of the American Control Conference*. IEEE. 2005, pp. 2686–2700.

[10] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear matrix inequalities in system and control theory*. Vol. 15. SIAM, 1994.

[11] G. Blekherman, P. A. Parrilo, and R. R. Thomas. *Semidefinite optimization and convex algebraic geometry*. SIAM, 2012.

[12] S. Kim, M. Kojima, and H. Waki. "Exploiting sparsity in SDP relaxation for sensor network localization". In: *SIAM Journal on Optimization* 20.1 (2009), pp. 192–215.

[13] R. P. Mason and A. Papachristodoulou. "Chordal sparsity, decomposing SDPs and the Lyapunov equation". In: *American Control Conference (ACC)*. IEEE. 2014, pp. 531–537.

[14] 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.

[15] L. Vandenberghe and M. S. Andersen. "Chordal graphs and semidefinite optimization". In: *Foundations and Trends® in Optimization* 1.4 (2015), pp. 241–433.

[16] F. Gavril. "Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph". In: *SIAM Journal on Computing* 1.2 (1972), pp. 180–187.

[17] D. J. Rose. "Triangulated graphs and the elimination process". In: *Journal of Mathematical Analysis and Applications* 32.3 (1970), pp. 597–609.

[18] J. Dahl, L. Vandenberghe, and V. Roychowdhury. "Covariance selection for nonchordal graphs via chordal embedding". In: *Optimization Methods & Software* 23.4 (2008), pp. 501–520.

[19] R. Grone, C. R. Johnson, E. M. Sá, and H. Wolkowicz. "Positive definite completions of partial Hermitian matrices". In: *Linear Algebra and its Applications* 58 (1984), pp. 109–124.

[20] J. Agler, W. Helton, S. McCullough, and L. Rodman. "Positive semidefinite matrices with a given sparsity pattern". In: *Linear Algebra and its Applications* 107 (1988), pp. 101–149.

[21] A. Griewank and P. L. Toint. "On the existence of convex decompositions of partially separable functions". In: *Mathematical Programming* 28.1 (1984), pp. 25–49.

[22] N. Kakimura. "A direct proof for the matrix decomposition of chordal-structured positive semidefinite matrices". In: *Linear Algebra and its Applications* 433.4 (2010), pp. 819–823.

[23] M. Fukuda, M. Kojima, K. Murota, and K. Nakata. "Exploiting sparsity in semidefinite programming via matrix completion I: General framework". In: *SIAM Journal on Optimization* 11.3 (2001), pp. 647–674.

[24] S. Kim, M. Kojima, M. Mevissen, and M. Yamashita. "Exploiting sparsity in linear and nonlinear matrix inequalities via positive semidefinite matrix completion". In: *Mathematical Programming* 129.1 (2011), pp. 33–68.

[25] M. S. Andersen, J. Dahl, and L. Vandenberghe. "Implementation of nonsymmetric interior-point methods for linear optimization over sparse matrix cones". In: *Mathematical Programming Computation* 2.3-4 (2010), pp. 167–201.

[26] Y. Sun, M. S. Andersen, and L. Vandenberghe. "Decomposition in conic optimization with partially separable structure". In: *SIAM Journal on Optimization* 24.2 (2014), pp. 873–897.

[27] A. Kalbat and J. Lavaei. "A fast distributed algorithm for decomposable semidefinite programs". In: *Decision and Control (CDC), IEEE 54th Annual Conference on*. IEEE. 2015, pp. 1742–1749.

[28] R. Madani, A. Kalbat, and J. Lavaei. "ADMM for sparse semidefinite programming with applications to optimal power flow problem". In: *Decision and Control (CDC), IEEE 54th Annual Conference on*. IEEE. 2015, pp. 5932–5939.

[29] M. S. Andersen, A. Hansson, and L. Vandenberghe. "Reduced-complexity semidefinite relaxations of optimal power flow problems". In: *IEEE Transactions on Power Systems* 29.4 (2014), pp. 1855–1863.

[30] R. A. Jabr. "Exploiting sparsity in SDP relaxations of the OPF problem". In: *IEEE Transactions on Power Systems* 27.2 (2012), pp. 1138–1139.

[31] H. Waki, S. Kim, M. Kojima, and M. Muramatsu. "Sums of squares and semidefinite program relaxations for polynomial optimization problems with structured sparsity". In: *SIAM Journal on Optimization* 17.1 (2006), pp. 218–242.

[32] J. B. Lasserre. "Convergent SDP-relaxations in polynomial optimization with sparsity". In: *SIAM Journal on Optimization* 17.3 (2006), pp. 822–843.

[33] R. Mason. "A chordal sparsity approach to scalable linear and nonlinear systems analysis". PhD thesis. University of Oxford, 2015.

[34] M. Andersen, S. Pakazad, A. Hansson, and A. Rantzer. "Robust stability analysis of sparsely interconnected uncertain systems". In: *IEEE Transactions on Automatic Control* 59.8 (2014), pp. 2151–2156.

[35] Y. Zheng, G. Fantuzzi, A. Papachristodoulou, P. Goulart, and A. Wynn. CDCS: Cone Decomposition Conic Solver. https://github.com/oxfordcontrol/CDCS. Sept. 2016.

[36] Y. Zheng, G. Fantuzzi, A. Papachristodoulou, P. Goulart, and A. Wynn. "Chordal decomposition in operator-splitting methods for sparse semidefinite programs". In: *Mathematical Programming, series A., accepted* (2019).

[37] Y. Zheng, G. Fantuzzi, A. Papachristodoulou, P. Goulart, and A. Wynn. "Fast ADMM for homogeneous self-dual embedding of sparse SDPs". In: *IFAC-PapersOnLine* 50.1 (2017), pp. 8411–8416.

[38] Y. Zheng, G. Fantuzzi, A. Papachristodoulou, P. Goulart, and A. Wynn. "Fast ADMM for semidefinite programs with chordal sparsity". In: *American Control Conference (ACC)*. IEEE. 2017, pp. 3335–3340.

[39] Y. Zheng, M. Kamgarpour, A. Sootla, and A. Papachristodoulou. "Scalable analysis of linear networked systems via chordal decomposition". In: *2018 European Control Conference (ECC)*. 2018, pp. 2260–2265.

[40] 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.

[41] Y. Zheng, R. P. Mason, and A. Papachristodoulou. "Scalable design of structured controllers using chordal decomposition". In: *IEEE Transactions on Automatic Control* 63.3 (2018), pp. 752–767.

[42] Y. Zheng, M. Kamgarpour, A. Sootla, and A. Papachristodoulou. "Distributed design of decentralized controllers using chordal decomposition and ADMM". In: *In submission* (2018).

[43] Y. Zheng, G. Fantuzzi, and A. Papachristodoulou. "Fast ADMM for sum-of-squares programs using partial orthogonality". In: *IEEE Transactions on Automatic Control* PP.99 (2018).

[44] Y. Zheng, G. Fantuzzi, and A. Papachristodoulou. "Decomposition and completion of sum-of-squares matrices". In: *2018 IEEE Conference on Decision and Control (CDC)*. 2018, pp. 4026–4031.

[45] Y. Zheng, G. Fantuzzi, and A. Papachristodoulou. "Sparse sum-of-squares (SOS) optimization: A bridge between DSOS/SDSOS and SOS optimization for sparse polynomials". In: *arXiv preprint arXiv:1807.05463* (2018).

[46] Y. Zheng, G. Fantuzzi, and A. Papachristodoulou. "Decomposition methods for large-scale semidefinite programs with chordal aggregate sparsity and partial orthogonality". In: *Large-Scale and Distributed Optimization*. Ed. by P. Giselsson and A. Rantzer. Springer International Publishing, 2018. Chap. 3.

[47] A. A. Ahmadi, G. Hall, A. Papachristodoulou, J. Saunderson, and Y. Zheng. "Improving efficiency and scalability of sum of squares optimization: Recent advances and limitations". In: *Decision and Control (CDC), IEEE 56th Annual Conference on*. IEEE. 2017, pp. 453–462.

[48] Y. Zheng, G. Fantuzzi, and A. Papachristodoulou. "Exploiting sparsity in the coefficient matching conditions in sum-of-squares programming using ADMM". In: *IEEE Control System Letter* 1.1 (2017), pp. 80–85.

[49] A. Sootla, Y. Zheng, and A. Papachristodoulou. "Block-diagonal solutions to Lyapunov inequalities and generalisations of diagonal dominance". In: *Decision and Control (CDC), IEEE 56th Annual Conference on*. IEEE. 2017, pp. 6561–6566.

[50] A. Sootla, Y. Zheng, and A. Papachristodoulou. "Block factor-width-two matrices in semidefinite programming". In: *European Control Conference (ECC), accepted* (2019).

[51] L. Furieri, Y. Zheng, A. Papachristodoulou, and M. Kamgarpour. "On separable quadratic Lyapunov functions for convex design of distributed controllers". In: *European Control Conference (ECC), accepted* (2019).

[52] P. A. Parrilo. "Semidefinite programming relaxations for semialgebraic problems". In: *Mathematical programming* 96.2 (2003), pp. 293–320.

[53] R. E. Tarjan and M. Yannakakis. "Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs". In: *SIAM Journal on computing* 13.3 (1984), pp. 566–579.

[54] M. Yannakakis. "Computing the minimum fill-in is NP-complete". In: *SIAM Journal on Algebraic Discrete Methods* 2.1 (1981), pp. 77–79.

[55] K. Nakata, K. Fujisawa, M. Fukuda, M. Kojima, and K. Murota. "Exploiting sparsity in semidefinite programming via matrix completion II: Implementation and numerical results". In: *Mathematical Programming* 95.2 (2003), pp. 303–327.

[56] A. Rantzer. "Distributed performance analysis of heterogeneous systems". In: *49th IEEE Conference on Decision and Control (CDC)*. IEEE. 2010, pp. 2682–2685.

[57] R. A. Horn and C. R. Johnson. *Matrix analysis*. Cambridge university press, 2012.

[58] L. Vandenberghe and S. Boyd. "Semidefinite programming". In: *SIAM review* 38.1 (1996), pp. 49–95.

[59] F. Alizadeh, J.-P. A. Haeberly, and M. L. Overton. "Primal-dual interior-point methods for semidefinite programming: convergence rates, stability and numerical results". In: *SIAM Journal on Optimization* 8.3 (1998), pp. 746–768.

[60] C. Helmberg, F. Rendl, R. J. Vanderbei, and H. Wolkowicz. "An interior-point method for semidefinite programming". In: *SIAM Journal on Optimization* 6.2 (1996), pp. 342–361.

[61] J. Malick, J. Povh, F. Rendl, and A. Wiegele. "Regularization methods for semidefinite programming". In: *SIAM Journal on Optimization* 20.1 (2009), pp. 336–356.

[62] Z. Wen, D. Goldfarb, and W. Yin. "Alternating direction augmented Lagrangian methods for semidefinite programming". In: *Mathematical Programming Computation* 2.3-4 (2010), pp. 203–230.

[63] X.-Y. Zhao, D. Sun, and K.-C. Toh. "A Newton-CG augmented Lagrangian method for semidefinite programming". In: *SIAM Journal on Optimization* 20.4 (2010), pp. 1737–1765.

[64] B. O'Donoghue, E. Chu, N. Parikh, and S. Boyd. "Conic optimization via operator splitting and homogeneous self-dual embedding". In: *Journal of Optimization Theory and Applications* 169.3 (2016), pp. 1042–1068.

[65] B. O'Donoghue, E. Chu, N. Parikh, and S. Boyd. *SCS: Splitting Conic Solver, version 1.2.6*. https://github.com/cvxgrp/scs. Apr. 2016.

[66] M. Andersen, J. Dahl, Z. Liu, and L. Vandenberghe. "Interior-point methods for large-scale cone programming". In: *Optimization for Machine Learning* 5583 (2011).

[67] K Fujisawa, S Kim, M Kojima, Y Okamoto, and M Yamashita. "User's manual for SparseCoLO: Conversion methods for sparse conic-form linear optimization problems". In: *Research Report B-453, Dept. of Math. and Comp. Sci. Japan, Tech. Rep.* (2009), pp. 152–8552.

[68] S. Burer. "Semidefinite programming in the space of partial positive semidefinite matrices". In: *SIAM Journal on Optimization* 14.1 (2003), pp. 139–172.

[69] E. Dall'Anese, H. Zhu, and G. B. Giannakis. "Distributed optimal power flow for smart microgrids". In: *IEEE Transactions on Smart Grid* 4.3 (2013), pp. 1464–1475.

[70] B. Borchers. "SDPLIB 1.2, a library of semidefinite programming test problems". In: *Optimization Methods and Software* 11.1-4 (1999), pp. 683–690.

[71] J. F. Sturm. "Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones". In: *Optimization Methods and Software* 11.1-4 (1999), pp. 625–653.

[72] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, et al. "Distributed optimization and statistical learning via the alternating direction method of multipliers". In: *Foundations and Trends® in Machine learning* 3.1 (2011), pp. 1–122.

[73] Y. Saad. *Iterative methods for sparse linear systems*. Vol. 82. siam, 2003.

[74] M. Yan and W. Yin. "Self equivalence of the alternating direction method of multipliers". In: *Splitting Methods in Communication, Imaging, Science, and Engineering*. Springer, 2016, pp. 165–194.

[75] G. Banjac, P. Goulart, B. Stellato, and S. Boyd. "Infeasibility detection in the alternating direction method of multipliers for convex optimization". In: *optimization-online.org* (June 2017). URL: http://www.optimization-online.org/DB_HTML/2017/06/6058.html.

[76] Y. Liu, E. K. Ryu, and W. Yin. "A New Use of Douglas-Rachford Splitting and ADMM for Identifying Infeasible, Unbounded, and Pathological Conic Programs". In: *arXiv preprint arXiv:1706.02374* (2017).

[77] Y. Ye, M. J. Todd, and S. Mizuno. "An $\mathcal{O}\sqrt{n}L$-iteration homogeneous and self-dual linear programming algorithm". In: *Mathematics of Operations Research* 19.1 (1994), pp. 53–67.

[78] Y. Ye. *Interior point algorithms: theory and analysis*. John Wiley & Sons, 2011.

[79] J. Lofberg. "YALMIP: A toolbox for modeling and optimization in MATLAB". In: *Computer Aided Control Systems Design, IEEE International Symposium on*. IEEE. 2004, pp. 284–289.

[80] A. Papachristodoulou, J. Anderson, G. Valmorbida, S. Prajna, P. Seiler, and P. Parrilo. "SOSTOOLS version 3.00 sum of squares optimization toolbox for MATLAB". In: *arXiv preprint arXiv:1310.4716* (2013).

[81] T. Davis. *Direct Methods for Sparse Linear Systems*. SIAM, 2006.

[82] E. Ghadimi, A. Teixeira, I. Shames, and M. Johansson. "Optimal parameter selection for the alternating direction method of multipliers (ADMM): quadratic problems". In: *IEEE Transactions on Automatic Control* 60.3 (2015), pp. 644–658.

[83] T. A. Davis and Y. Hu. "The University of Florida sparse matrix collection". In: *ACM Transactions on Mathematical Software (TOMS)* 38.1 (2011), p. 1.

[84] P Moylan and D Hill. "Stability criteria for large-scale systems". In: *IEEE Transactions on Automatic Control* 23.2 (1978), pp. 143–149.

[85] M Vidyasagar. "New passivity-type criteria for large-scale interconnected systems". In: *IEEE Transactions on Automatic Control* 24.4 (1979), pp. 575–579.

[86] C. Meissen, L. Lessard, M. Arcak, and A. K. Packard. "Compositional performance certification of interconnected systems using ADMM". In: *Automatica* 61 (2015), pp. 55–63.

[87] J. Anderson and A. Papachristodoulou. "A decomposition technique for nonlinear dynamical system analysis". In: *IEEE Transactions on Automatic Control* 57.6 (2012), pp. 1516–1521.

[88] L. Farina and S. Rinaldi. *Positive linear systems: theory and applications*. Vol. 50. John Wiley & Sons, 2011.

[89] A. Rantzer. "Scalable control of positive systems". In: *European Journal of Control* 24 (2015), pp. 72–80.

[90] T. Tanaka and C. Langbort. "The bounded real lemma for internally positive systems and H-infinity structured static state feedback". In: *IEEE Transactions on Automatic Control* 56.9 (2011), pp. 2218–2223.

[91] A. Sootla and A. Rantzer. "Scalable positivity preserving model reduction using linear energy functions". In: *Decision and Control (CDC), IEEE 51st Annual Conference on*. IEEE. 2012, pp. 4285–4290.

[92] R. Albert and A.-L. Barabási. "Statistical mechanics of complex networks". In: *Reviews of Modern Physics* 74.1 (2002), p. 47.

[93] A. Zečević and D. Šiljak. "Control design with arbitrary information structure constraints". In: *Automatica* 44.10 (2008), pp. 2642–2647.

[94] J. Swigart and S. Lall. "Optimal controller synthesis for decentralized systems over graphs via spectral factorization". In: *IEEE Transactions on Automatic Control* 59.9 (2014), pp. 2311–2323.

[95] 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". In: *IEEE Transactions on Intelligent Transportation Systems* 17.1 (2016), pp. 14–26.

[96] F. DorIer, M. R. Jovanovic, M. Chertkov, and F. Bullo. "Sparsity-promoting optimal wide-area control of power networks". In: *IEEE Transactions on Power System* 29.5 (2014), pp. 2281–2291.

[97] D. M. Stipanović, G. Inalhan, R. Teo, and C. J. Tomlin. "Decentralized overlapping control of a formation of unmanned aerial vehicles". In: *Automatica* 40.8 (2004), pp. 1285–1296.

[98] Y. Zheng, S. E. Li, K. Li, and L.-Y. Wang. "Stability margin improvement of vehicular platoon considering undirected topology and asymmetric control". In: *IEEE Transactions on Control Systems Technology* 24.4 (2016), pp. 1253–1265.

[99] V. Blondel and J. N. Tsitsiklis. "NP-hardness of some linear control design problems". In: *SIAM Journal on Control and Optimization* 35.6 (1997), pp. 2118–2127.

[100] P. Shah and P. Parrilo. "H2-Optimal decentralized control over posets: A state-space solution for state-feedback". In: *IEEE Transactions on Automatic Control* 58.12 (2013), pp. 3084–3096.

[101] J.-H. Kim and S. Lall. "Explicit solutions to separable problems in optimal cooperative control". In: *IEEE Transactions on Automatic Control* 60.5 (2015), pp. 1304–1319.

[102] G. Fazelnia, R. Madani, A. Kalbat, and J. Lavaei. "Convex relaxation for optimal distributed control problems". In: *IEEE Transactions on Automatic Control* 62.1 (2017), pp. 206–221.

[103] K. Dvijotham, E. Todorov, and M. Fazel. "Convex structured controller design in finite horizon". In: *IEEE Transactions on Control of Network Systems* 2.1 (2015), pp. 1–10.

[104] F. Lin, M. Fardad, and M. R. Jovanovic. "Augmented Lagrangian approach to design of structured optimal state feedback gains". In: *IEEE Transactions on Automatic Control* 56.12 (2011), pp. 2923–2929.

[105] F. Lin, M. Fardad, and M. R. Jovanovic. "Design of optimal sparse feedback gains via the alternating direction method of multipliers". In: *IEEE Transactions on Automatic Control* 58.9 (2013), pp. 2426–2431.

[106] A. Satya Mohan Vamsi and N. Elia. "Optimal distributed controllers realizable over arbitrary networks". In: *IEEE Transactions on Automatic Control* 61.1 (2016), pp. 129–144.

[107] W. Su, H. Eichi, W. Zeng, and M.-Y. Chow. "A survey on the electrification of transportation in a smart grid environment". In: *IEEE Transactions on Industrial Informatics* 8.1 (2012), pp. 1–10.

[108] J. Zhang, F.-Y. Wang, K. Wang, W.-H. Lin, X. Xu, and C. Chen. "Data-driven intelligent transportation systems: A survey". In: *IEEE Transactions on Intelligent Transportation Systems* 12.4 (2011), pp. 1624–1639.

[109] M. J. Wainwright and M. I. Jordan. "Graphical models, exponential families, and variational inference". In: *Foundations and Trends® in Machine Learning* 1.1-2 (2008), pp. 1–305.

[110] J. C. Geromel, J. Bernussou, and P. L. D. Peres. "Decentralized control through parameter space optimization". In: *Automatica* 30.10 (1994), pp. 1565–1578.

[111] 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)*. IEEE. 2015, pp. 286–291.

[112] M. Yamashita, K. Fujisawa, and M. Kojima. "Implementation and evaluation of SDPA 6.0 (semidefinite programming algorithm 6.0)". In: *Optimization Methods and Software* 18.4 (2003), pp. 491–505.

[113] S. K. Pakazad, A. Hansson, M. S. Andersen, and A. Rantzer. "Distributed semidefinite programming with application to large-scale system analysis". In: *IEEE Transactions on Automatic Control* 63.4 (2018), pp. 1045–1058.

[114] M. Razeghi-Jahromi and A. Seyedi. "Stabilization of networked control systems with Sparse observer-controller networks". In: *IEEE Transactions on Automatic Control* 60.6 (2015), pp. 1686–1691.

[115] S.-H. Wang and E. Davison. "On the stabilization of decentralized control systems". In: *IEEE Transactions on Automatic Control* 18.5 (1973), pp. 473–478.

[116] M. R. Jovanović and N. K. Dhingra. "Controller architectures: Tradeoffs between performance and structure". In: *European Journal of Control* 30 (2016), pp. 76–91.

[117] C. Langbort and J.-C. Delvenne. "Distributed design methods for linear quadratic control and their limitations". In: *IEEE Transactions on Automatic Control* 55.9 (2010), pp. 2085–2093.

[118] J. Lunze. *Feedback control of large scale systems*. Prentice Hall PTR, 1992.

[119] F. Farokhi, C. Langbort, and K. H. Johansson. "Optimal structured static state-feedback control design with limited model information for fully-actuated systems". In: *Automatica* 49.2 (2013), pp. 326–337.

[120] P. Giselsson, M. D. Doan, T. Keviczky, B. De Schutter, and A. Rantzer. "Accelerated gradient methods and dual decomposition in distributed model predictive control". In: *Automatica* 49.3 (2013), pp. 829–833.

[121] F. Deroo, M. Meinel, M. Ulbrich, and S. Hirche. "Distributed control design with local model information and guaranteed stability". In: *19th IFAC World Congress*. 2014, pp. 4010–4017.

[122] M. Ahmadi, M. Cubuktepe, U. Topcu, and T. Tanaka. "Distributed synthesis using accelerated ADMM". In: *2018 Annual American Control Conference (ACC)*. IEEE. 2018, pp. 6206–6211.

[123] Y.-S. Wang, N. Matni, and J. C. Doyle. "Separable and localized system level synthesis for large-scale systems". In: *IEEE Transactions on Automatic Control* (2018).

[124] A. A. Ahmadi and A. Majumdar. "DSOS and SDSOS optimization: more tractable alternatives to sum of squares and semidefinite optimization". In: *arXiv preprint arXiv:1706.02586* (2017).

[125] J. B. Lasserre. "Moments and sums of squares for polynomial optimization and related problems". In: *Journal of Global Optimization* 45.1 (2009), pp. 39–61.

[126] J. Anderson and A. Papachristodoulou. "Advances in computational Lyapunov analysis using sum-of-squares programming". In: *Discrete & Continuous Dynamical Systems-Series B* 20.8 (2015).

[127] J. B. Lasserre. "Global optimization with polynomials and the problem of moments". In: *SIAM Journal on Optimization* 11.3 (2001), pp. 796–817.

[128] F. Permenter and P. A. Parrilo. "Basis selection for SOS programs via facial reduction and polyhedral approximations". In: *Decision and Control (CDC), IEEE 53rd Annual Conference on*. IEEE. 2014, pp. 6615–6620.

[129] B. Reznick et al. "Extremal PSD forms with few terms". In: *Duke mathematical journal* 45.2 (1978), pp. 363–374.

[130] J. Lofberg. "Pre-and post-processing sum-of-squares programs in practice". In: *IEEE Transactions on Automatic Control* 54.5 (2009), pp. 1007–1011.

[131] K. Gatermann and P. A. Parrilo. "Symmetry groups, semidefinite programs, and sums of squares". In: *Journal of Pure and Applied Algebra* 192.1-3 (2004), pp. 95–128.

[132] A. A. Ahmadi and G. Hall. *Sum of Squares Basis Pursuit with Linear and Second Order Cone Programming*. arXiv:1510.01597. 2015.

[133] D. Henrion and J. Malick. "Projection methods in conic optimization". In: *Handbook on Semidefinite, Conic and Polynomial Optimization*. Springer, 2012, pp. 565–600.

[134] D. Bertsimas, R. M. Freund, and X. A. Sun. "An accelerated first-order method for solving SOS relaxations of unconstrained polynomial optimization problems". In: *Optimization Methods and Software* 28.3 (2013), pp. 424–441.

[135] J. Nie and L. Wang. "Regularization methods for SDP relaxations in large-scale polynomial optimization". In: *SIAM Journal on Optimization* 22.2 (2012), pp. 408–428.

[136] D. Henrion, J.-B. Lasserre, and J. Löfberg. "GloptiPoly 3: moments, optimization and semidefinite programming". In: *Optimization Methods & Software* 24.4-5 (2009), pp. 761–779.

[137] K.-C. Toh, M. J. Todd, and R. H. Tutuncu. "SDPT3—a MATLAB software package for semidefinite programming, version 1.3". In: *Optimization Methods and Software* 11.1-4 (1999), pp. 545–581.

[138] B. Borchers. "CSDP, A C library for semidefinite programming". In: *Optimization Methods and Software* 11.1-4 (1999), pp. 613–623.

[139] A. Mosek. "The MOSEK optimization software". In: *Online at http://www.mosek.com* 54.2-1 (2010), p. 5.

[140] A. Papachristodoulou and S. Prajna. "On the construction of Lyapunov functions using the sum of squares decomposition". In: *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*. Vol. 3. IEEE. 2002, pp. 3482–3487.

[141] V. Powers and T. Wormann. "An algorithm for sums of squares of real polynomials". In: *Journal of Pure and Applied Algebra* 127.1 (1998), pp. 99–104.

[142] C. W. Scherer and C. W. Hol. "Matrix sum-of-squares relaxations for robust semi-definite programs". In: *Mathematical Programming* 107.1-2 (2006), pp. 189–211.

[143] M. Kojima. "Sums of squares relaxations of polynomial semidefinite programs". In: (2003).

[144] S. H. A. Khoshnaw. "Model reductions in biochemical reaction networks". PhD thesis. Department of Mathematics, University of Leicester, 2015.

[145] E. J. Candès and B. Recht. "Exact matrix completion via convex optimization". In: *Foundations of Computational Mathematics* 9.6 (2009), p. 717.

[146] V. Chandrasekaran, S. Sanghavi, P. A. Parrilo, and A. S. Willsky. "Rank-sparsity incoherence for matrix decomposition". In: *SIAM Journal on Optimization* 21.2 (2011), pp. 572–596.

[147] M. M. Peet, A. Papachristodoulou, and S. Lall. "Positive forms and stability of linear time-delay systems". In: *SIAM Journal on Control and Optimization* 47.6 (2009), pp. 3237–3258.

[148] W. Tan, A. Packard, et al. "Stability region analysis using polynomial and composite polynomial Lyapunov functions and sum-of-squares programming". In: *IEEE Transactions on Automatic Control* 53.2 (2008), p. 565.

[149] U. Topcu, A. Packard, and P. Seiler. "Local stability analysis using simulations and sum-of-squares programming". In: *Automatica* 44.10 (2008), pp. 2669–2675.

[150] G. Valmorbida, M. Ahmadi, and A. Papachristodoulou. "Stability analysis for a class of partial differential equations via semidefinite programming". In: *IEEE Transactions on Automatic Control* 61.6 (2016), pp. 1649–1654.

[151] M. Ahmadi, G. Valmorbida, and A. Papachristodoulou. "Dissipation inequalities for the analysis of a class of PDEs". In: *Automatica* 66 (2016), pp. 163–171.

[152] S. I. Chernyshenko, P Goulart, D Huang, and A. Papachristodoulou. "Polynomial sum of squares in fluid dynamics: a review with a look ahead". In: *Phil. Trans. R. Soc. A* 372.2020 (2014), p. 20130350.

[153] D. Goluskin. "Bounding averages rigorously using semidefinite programming: mean moments of the Lorenz system". In: *Journal of Nonlinear Science* 28.2 (2018), pp. 621–651.

[154] G. Fantuzzi, D. Goluskin, D. Huang, and S. I. Chernyshenko. "Bounds for deterministic and stochastic dynamical systems using sum-of-squares optimization". In: *SIAM Journal on Applied Dynamical Systems* 15.4 (2016), pp. 1962–1988.

[155] T. Weisser, J. B. Lasserre, and K.-C. Toh. "Sparse-BSOS: a bounded degree SOS hierarchy for large scale polynomial optimization with sparsity". In: *Mathematical Programming Computation* 10.1 (2018), pp. 1–32.

[156] C. Josz and D. K. Molzahn. "Lasserre hierarchy for large scale polynomial optimization in real and complex variables". In: *SIAM Journal on Optimization* 28.2 (2018), pp. 1017–1048.

[157] A. A. Ahmadi, S. Dash, and G. Hall. "Optimization over structured subsets of positive semidefinite matrices via column generation". In: *Discrete Optimization* 24 (2017), pp. 129–151.

[158] M. M. Tobenkin, F. Permenter, and A. Megretski. *Spotless polynomial and conic optimization*. 2013.

[159] M. Dür. "Copositive programming–a survey". In: *Recent advances in optimization and its applications in engineering*. Springer, 2010, pp. 3–20.

[160] A. Themelis and P. Patrinos. "SuperMann: a superlinearly convergent algorithm for finding fixed points of nonexpansive operators". In: *arXiv preprint arXiv:1609.06955* (2016).

[161] H. F. Walker and P. Ni. "Anderson acceleration for fixed-point iterations". In: *SIAM Journal on Numerical Analysis* 49.4 (2011), pp. 1715–1735.

[162] M. Journée, F. Bach, P.-A. Absil, and R. Sepulchre. "Low-rank optimization on the cone of positive semidefinite matrices". In: *SIAM Journal on Optimization* 20.5 (2010), pp. 2327–2351.

[163] B. Recht, M. Fazel, and P. A. Parrilo. "Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization". In: *SIAM Review* 52.3 (2010), pp. 471–501.

[164] J. Dancis. "Positive semidefinite completions of partial hermitian matrices". In: *Linear Algebra and its Applications* 175 (1992), pp. 97–114.

[165] E. M. Aylward, S. M. Itani, and P. A. Parrilo. "Explicit SOS decompositions of univariate polynomial matrices and the Kalman-Yakubovich-Popov lemma". In: *Decision and Control, 46th IEEE Conference on*. IEEE. 2007, pp. 5660–5665.

[166] M. Laurent. "Sums of squares, moment matrices and optimization over polynomials". In: *Emerging Applications of Algebraic Geometry*. Springer, 2009, pp. 157–270.

[167] M. Putinar. "Positive polynomials on compact semi-algebraic sets". In: *Indiana University Mathematics Journal* 42.3 (1993), pp. 969–984.

[168] E. Kaszkurewicz and A. Bhaya. *Matrix diagonal stability in systems and computation*. Springer Science & Business Media, 2012.

[169] M. Arcak and E. D. Sontag. "Diagonal stability of a class of cyclic systems and its connection with the secant criterion". In: *Automatica* 42.9 (2006), pp. 1531–1537.

[170] M. Arcak. "Diagonal stability on cactus graphs and application to network stability analysis". In: *IEEE Transactions on Automatic Control* 56.12 (2011), pp. 2766–2777.

[171] D. Carlson, D. Hershkowitz, and D. Shasha. "Block diagonal semistability factors and Lyapunov semistability of block triangular matrices". In: *Linear Algebra and its Applications* 172 (1992), pp. 1–25.

[172] A. Sootla and J. Anderson. "On existence of solutions to structured lyapunov inequalities". In: *American Control Conference (ACC)*. IEEE. 2016, pp. 7013–7018.

[173] A. Berman, F. Goldberg, and R. Shorten. "Comments on Lyapunov α-stability with some extensions". In: *Variational and Optimal Control Problems on Unbounded Domains* (2014).

[174] A. Alavian and M. Rotkowitz. "Stabilizing decentralized systems with arbitrary information structure". In: *Decision and Control (CDC), IEEE 53rd Annual Conference on*. IEEE. 2014, pp. 4032–4038.

[175] B. D. Anderson and D. J. Clements. "Algebraic characterization of fixed modes in decentralized control". In: *Automatica* 17.5 (1981), pp. 703–712.