Scalable Semidefinite and Sum-of-square Optimization via Matrix Decomposition

Yang Zheng, PhD

Postdoc, School of Engineering and Applied Sciences
Harvard University

Seminar at EECS, KTH Royal Institute of Technology

Dec 18, 2019

Acknowledgments

UNIVERSITY OF OXFORD

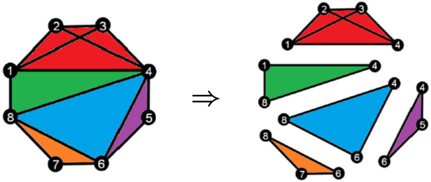
Imperial College London

HARVARD

Outline

1. Introduction: Matrix decomposition and chordal graphs
2. Part I - Decomposition in sparse semidefinite optimization
3. Part II - Decomposition in sparse sum-of-squares optimization
4. Part III - Beyond chordal decomposition
5. Conclusion

Introduction: Matrix decomposition and chordal graphs



Matrix decomposition and chordal graphs

Matrix decomposition:

- A simple example

$$A = \underbrace{\begin{bmatrix} 3 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 3 \end{bmatrix}}_{\succeq 0} = \underbrace{\begin{bmatrix} 3 & 1 & 0 \\ 1 & 0.5 & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{\succeq 0} + \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0.5 & 1 \\ 0 & 1 & 3 \end{bmatrix}}_{\succeq 0}$$

- This is true for any PSD matrix with such pattern, *i.e.*, sparse cone decomposition

$$\underbrace{\begin{bmatrix} * & * & 0 \\ * & * & * \\ 0 & * & * \end{bmatrix}}_{\succeq 0} = \underbrace{\begin{bmatrix} * & * & 0 \\ * & * & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{\succeq 0} + \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & * & * \\ 0 & * & * \end{bmatrix}}_{\succeq 0}$$

  where $*$ denotes a real scalar number.

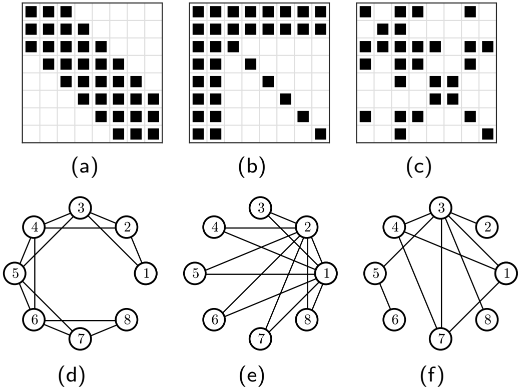
Benefits:

- Reduce computational complexity, and thus improve efficiency! $(3 \times 3 \rightarrow 2 \times 2)$

Matrix decomposition and chordal graphs

Matrix decomposition:

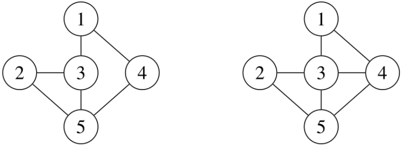
- Many other patterns admit similar decompositions, *e.g.*



- They can be commonly characterized by **chordal graphs**.

Matrix decomposition and chordal graphs

Chordal graphs: An undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is called *chordal* if every cycle of length greater than three has a chord.

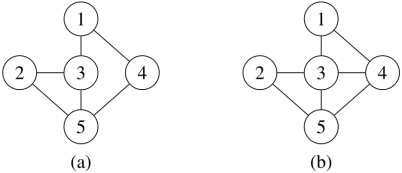


(a) (b)

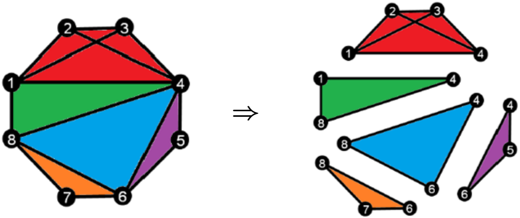
Notation: (Vandenberghe, & Andersen, 2014)

- *Chordal extension:* Any non-chordal graph can be chordal extended;
- *Maximal clique:* A clique is a set of nodes that induces a complete subgraph;
- *Clique decomposition:* A chordal graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ can be decomposed into a set of maximal cliques $\{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_p\}$.

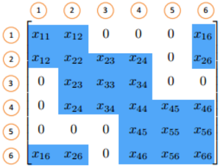
Chordal graphs: An undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is called *chordal* if every cycle of length greater than three has a chord.

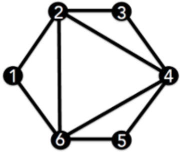


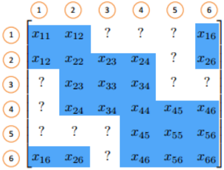
Clique decomposition:



Matrix decomposition and chordal graphs







Sparse positive semidefinite (PSD) matrices

$$\mathbb{S}^n(\mathcal{E},0) = \{X \in \mathbb{S}^n \mid X_{ij} = X_{ji} = 0, \forall (i,j) \notin \mathcal{E}\},$$
$$\mathbb{S}^n_+(\mathcal{E},0) = \{X \in \mathbb{S}^n(\mathcal{E},0) \mid X \succeq 0\}.$$

Positive semidefinite completable matrices

$$\mathbb{S}^n(\mathcal{E},?) = \{X \in \mathbb{S}^n \mid X_{ij} = X_{ji}, \text{ given if } (i,j) \in \mathcal{E}\},$$
$$\mathbb{S}^n_+(\mathcal{E},?) = \{X \in \mathbb{S}^n(\mathcal{E},?) \mid \exists M \succeq 0, M_{ij} = X_{ij}, \forall (i,j) \in \mathcal{E}\}.$$

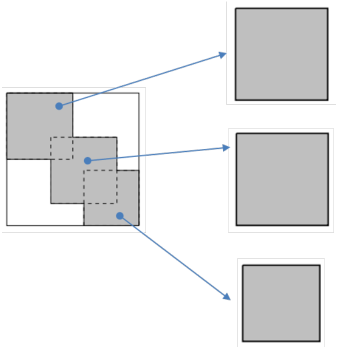
$\mathbb{S}^n_+(\mathcal{E},0)$ and $\mathbb{S}^n_+(\mathcal{E},?)$ are dual to each other.

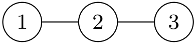
Matrix decomposition and chordal graphs

Clique decomposition for PSD completable matrices (Grone, *et al.*, 1984)

Let $\mathcal{G}(\mathcal{V}, \mathcal{E})$ be a chordal graph with maximal cliques $\{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_p\}$. Then,

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





$$\begin{bmatrix} X_{11} & X_{12} & ? \\ X_{21} & X_{22} & X_{23} \\ ? & X_{32} & X_{33} \end{bmatrix} \in \mathbb{S}_+^3(\mathcal{E}, ?)$$

$$\Updownarrow$$

$$\begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix} \succeq 0$$

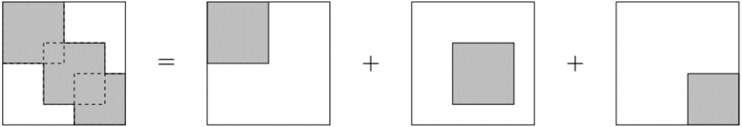
$$\begin{bmatrix} X_{22} & X_{23} \\ X_{32} & X_{33} \end{bmatrix} \succeq 0$$

Matrix decomposition and chordal graphs

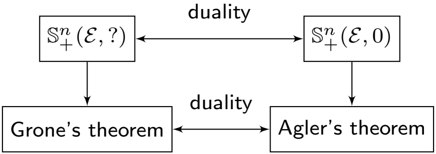
Clique decomposition for PSD matrices (Agler, *et al.*, 1988; Griewank and Toint, 1984)

Let $\mathcal{G}(\mathcal{V},\mathcal{E})$ be a chordal graph with maximal cliques $\{\mathcal{C}_1,\mathcal{C}_2,\ldots,\mathcal{C}_p\}$. Then,

$$Z \in \mathbb{S}^n_+(\mathcal{E},0) \Leftrightarrow Z = \sum_{k=1}^{p} E_{\mathcal{C}_k}^T Z_k E_{\mathcal{C}_k},\ Z_k \in \mathbb{S}_+^{|\mathcal{C}_k|}$$



Sparse Cone Decomposition



Matrix decomposition and chordal graphs

Applications (a partial and incomplete list)

- **Sparse semidefinite programs** → Part I of the talk
  - Fukuda, Kojima, Murota, Nakata, 2001; Andersen, Dahl, Vandenberghe, 2010; Sun, Andersen, Vandenberghe, 2014; Madani, Kalbat, Lavaei, 2015; Zheng, Fantuzzi, Papachristodoulou, Goulart, Wynn, 2017;
- Analysis and control of sparse networked systems
  - Andersen, Pakazad, Hansson, Rantzer, 2014; Mason, Papachristodoulou, 2014; Zheng, Mason, Papachristodoulou, 2018; Pakazad, Hansson, Andersen, Rantzer, 2018; Zheng, Kamgarpour, Sootla, Papachristodoulou, 2018.
- Power systems (OPF problems)
  - Dall'Anese, Zhu, Giannakis, 2013; Andersen, Hansson, Vandenberghe, 2014
- **Polynomial optimization** → Part II of the talk
  - Waki, Kim, Kojima, Muramatsu, 2006; Lasserre, 2006; Fawzi, Saunderson, Parrilo, 2016.

A survey paper

- Vandenberghe, Lieven, and Martin S. Andersen. "Chordal graphs and semidefinite optimization." Foundations and Trends in Optimization 1.4 (2015): 241-433.

Part I: Decomposition in sparse semidefinite optimization

Sparse semidefinite programs (SDPs)

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

$$\begin{aligned} \max_{y,\, Z} \quad & \langle b, y \rangle \\ \text{subject to} \quad & Z + \sum_{i=1}^{m} A_i\, y_i = C, \\ & Z \succeq 0. \end{aligned}$$

where $X \succeq 0$ means X is positive semidefinite.

- **Applications:** Control theory, fluid dynamics, polynomial optimization, *etc*.
- **Interior-point solvers:** SeDuMi, SDPA, SDPT3 (suitable for small and medium-sized problems); *Modelling package:* YALMIP, CVX
- **Large-scale cases:** it is important to exploit the inherent structure
  - Low rank;
  - Algebraic symmetry;
  - **Chordal sparsity**
    - Second-order methods: Fukuda *et al.*, 2001; Nakata *et al.*, 2003; Burer 2003; Andersen *et al.*, 2010.
    - **First-order methods**: Madani *et al.*, 2015; Sun, Andersen, and Vandenberghe, 2014.

Aggregate sparsity pattern of matrices

$$C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, A_1 = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 2 \end{bmatrix}, A_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix} \Longrightarrow \begin{bmatrix} * & * & 0 \\ * & * & * \\ 0 & * & * \end{bmatrix}$$

Primal SDP

$$\begin{aligned} \min \quad & \langle C, X \rangle \\ \text{subject to} \quad & \langle A_1, X \rangle = b_1 \\ & \langle A_2, X \rangle = b_2 \\ & X \succeq 0. \end{aligned}$$

Dual SDP

$$\begin{aligned} \max_{y,\, Z} \quad & \langle b, y \rangle \\ \text{subject to} \quad & y_1 A_1 + y_2 A_2 + Z = C, \\ & Z \succeq 0. \end{aligned}$$

$X \in \begin{bmatrix} * & * & ? \\ * & * & * \\ ? & * & * \end{bmatrix}$	Patterns of feasible solutions	$Z \in \begin{bmatrix} * & * & 0 \\ * & * & * \\ 0 & * & * \end{bmatrix}$
$X \in \mathbb{S}^3_+(\mathcal{E}, ?)$	Cone replacement	$Z \in \mathbb{S}^3_+(\mathcal{E}, 0)$

Apply the clique decomposition on $\mathbb{S}^3_+(\mathcal{E}, ?)$ and $\mathbb{S}^3_+(\mathcal{E}, 0)$

- Fukuda *et al.*, 2001; Nakata *et al.*, 2003; Andersen *et al.*, 2010; Madani *et al.*, 2015; Sun, Andersen, and Vandenberghe, 2014.

Cone decomposition of sparse SDPs

Primal SDP

$$\begin{aligned} \min \quad & \langle C, X\rangle \\ \text{subject to} \quad & \langle A_i, X\rangle = b_i, i = 1, \ldots, m \\ & \boxed{X \succeq 0}. \end{aligned}$$

Dual SDP

$$\begin{aligned} \max_{y,\, Z} \quad & \langle b, y\rangle \\ \text{subject to} \quad & \sum_{i=1}^{m} y_i A_i + Z = C, \\ & \boxed{Z \succeq 0}. \end{aligned}$$

$X \in \mathbb{S}^n_+(\mathcal{E}, ?)$

Cone replacement

(Assuming an aggregate sparsity pattern $\mathcal{E}$)

$Z \in \mathbb{S}^n_+(\mathcal{E}, 0)$

$\Downarrow$ $\Downarrow$

$$\begin{aligned} \min \quad & \langle C, X\rangle \\ \text{s.t.} \quad & \langle A_i, X\rangle = b_i, i = 1, \ldots, m \\ & \boxed{E_{\mathcal{C}_k} X E_{\mathcal{C}_k}^T \succeq 0, k = 1, \ldots, p}. \end{aligned}$$

$$\begin{aligned} \max_{y,\, Z} \quad & \langle b, y\rangle \\ \text{s.t.} \quad & \sum_{i=1}^{m} y_i A_i + \sum_{k=1}^{p} E_{\mathcal{C}_k}^T Z_k E_{\mathcal{C}_k} = C, \\ & \boxed{Z_k \succeq 0, k = 1, \ldots, p} \end{aligned}$$

- **A big sparse PSD cone** is equivalently replaced by a set of **coupled small** PSD cones;
- Our idea: **consensus** variables $\Rightarrow$ decouple the coupling constraints;

Decomposed SDPs for operator-splitting algorithms

Primal decomposed SDP

$$\begin{aligned} \min_{X, X_k} \quad & \langle C, X \rangle \\ \text{s.t.} \quad & \langle A_i, X \rangle = b_i, \qquad i = 1, \ldots, m, \\ & \boxed{X_k = E_{\mathcal{C}_k} X E_{\mathcal{C}_k}^T, k = 1, \ldots, p,} \\ & X_k \in \mathbb{S}_+^{|\mathcal{C}_k|}, \qquad k = 1, \ldots, p. \end{aligned}$$

Dual decomposed SDP

$$\begin{aligned} \max_{y, Z_k, V_k} \quad & \langle b, y \rangle \\ \text{s.t.} \quad & \sum_{i=1}^{m} A_i \, y_i + \sum_{k=1}^{p} E_{\mathcal{C}_k}^T V_k E_{\mathcal{C}_k} = C, \\ & \boxed{Z_k - V_k = 0, \ k = 1, \ldots, p,} \\ & Z_k \in \mathbb{S}_+^{|\mathcal{C}_k|}, \quad k = 1, \ldots, p. \end{aligned}$$

- A set of slack consensus variables has been introduced;
- The slack variables allow one to **separate the conic and the affine constraints** when using operator-splitting algorithms $\Rightarrow$ fast iterations

Vectorization

$$\begin{aligned} \min_{x, x_k} \quad & \langle c, x \rangle \\ \text{s.t.} \quad & Ax = b, \\ & \boxed{x_k = H_k x}, \quad k = 1, \ldots, p, \\ & x_k \in \mathcal{S}_k, \qquad k = 1, \ldots, p, \end{aligned}$$

$$\begin{aligned} \max_{y, z_k, v_k} \quad & \langle b, y \rangle \\ \text{s.t.} \quad & A^T y + \sum_{k=1}^{p} H_k^T v_k = c, \\ & \boxed{z_k - v_k = 0}, k = 1, \ldots, p, \\ & z_k \in \mathcal{S}_k, k = 1, \ldots, p. \end{aligned}$$

Alternating Direction Method of Multipliers (ADMM)

The ADMM algorithm solves the optimization problem (Bertsekas and Tsitsiklis, 1989; Boyd, *et al.*, 2011)

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

where f and g are convex functions.

- **Augmented Lagrangian**

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

- **ADMM steps**

$$\begin{aligned} x^{(n+1)} &= \arg\min_x \; \mathcal{L}_\rho(x, y^{(n)}, z^{(n)}), && \rightarrow x\text{-minimization step} \\ y^{(n+1)} &= \arg\min_y \; \mathcal{L}_\rho(x^{(n+1)}, y, z^{(n)}), && \rightarrow y\text{-minimization step} \\ z^{(n+1)} &= z^{(n)} + \rho \left(Ax^{(n+1)} + By^{(n+1)} - c\right). && \rightarrow \text{dual variable update} \end{aligned}$$

ADMM is particularly suitable when the subproblems have closed-form expressions, or can be solved efficiently.

ADMM for primal decomposed SDPs

$$\begin{aligned} \min_{x, x_k} \quad & \langle c, x \rangle \\ \text{s.t.} \quad & Ax = b, \\ & \boxed{x_k = H_k x}, \quad k = 1, \ldots, p, \\ & x_k \in \mathcal{S}_k, \qquad k = 1, \ldots, p, \end{aligned}$$

Reformulation using indicator functions

$$\begin{aligned} \min_{x, x_1, \ldots, x_p} \quad & \langle c, x \rangle + \delta_0 \left(Ax - b\right) + \sum_{k=1}^{p} \delta_{\mathcal{S}_k}(x_k) \\ \text{s.t.} \quad & x_k = H_k x, \quad k = 1, \ldots, p. \end{aligned}$$

- *x-minimization step:* QP with linear constraints, KKT condition

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

- *y-minimization step:* Parallel projections onto **small PSD cones**

$$\begin{aligned} \min_{x_k} \quad & \left\| x_k - H_k x^{(n+1)} + \rho^{-1} \lambda_k^{(n)} \right\|^2 \\ \text{s.t.} \quad & x_k \in \mathcal{S}_k. \end{aligned}$$

- Update multipliers

ADMM for dual decomposed SDPs

$$\begin{aligned}\max_{y, z_k, v_k} \quad & \langle b, y \rangle \\ \text{s.t.} \quad & A^T y + \sum_{k=1}^{p} H_k^T v_k = c, \\ & \boxed{z_k - v_k = 0}, k = 1, \ldots, p, \\ & z_k \in \mathcal{S}_k, k = 1, \ldots, p.\end{aligned}$$

Reformulation using indicator functions

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

- *x-minimization step:* QP with linear constraints, KKT condition

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

- *y-minimization step:* Parallel projections onto **small PSD cones**

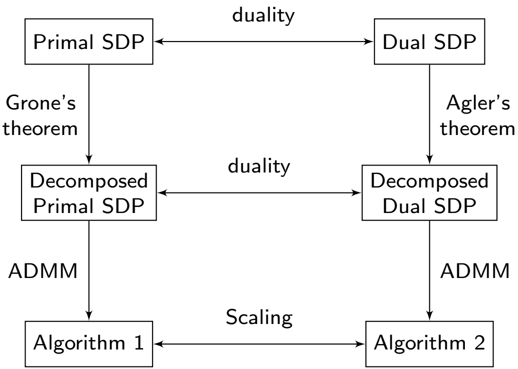
$$\begin{aligned}\min_{z_k} \quad & \left\| z_k - v_k^{(n)} + \rho^{-1} \lambda_k^{(n)} \right\|^2 \\ \text{s.t.} \quad & z_k \in \mathcal{S}_k.\end{aligned}$$

- Update multipliers

ADMM for primal and dual decomposed SDPs

Equivalence between the primal and dual cases

- ADMM steps in the dual form are scaled versions of those in the primal form.
- Extension to the homogeneous self-dual embedding exists.



Both algorithms only require conic projections onto small PSD cones. **Complexity depends on the largest maximal cliques, instead of the original dimension!**

CDCS

Cone decomposition conic solver

- An open source MATLAB solver for sparse conic programs;
- CDCS supports constraints on the following cones:
  - Free variables
  - non-negative orthant
  - second-order cone
  - the positive semidefinite cone.
- Input-output format is in accordance with SeDuMi; Interface via YALMIP.
- Syntax: `[x,y,z,info] = cdcs(At,b,c,K,opts);`

Download from `https://github.com/OxfordControl/CDCS`

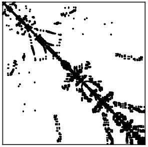
Numerical comparison

- SeDuMi (interior-point solver): default parameters, and low-accuracy solution 10^{-3}
- SCS (first-order solver)
- CDCS and SCS: stopping condition 10^{-3} (max. iterations 2000)
- All simulations were run on a PC with a 2.8 GHz Intel Core i7 CPU and 8GB of RAM.

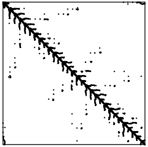
Large-scale sparse SDPs

Instances from Andersen, Dahl, Vandenberghe, 2010

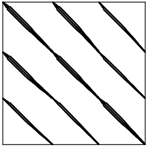
	rs35	rs200	rs228	rs365	rs1555	rs1907
Original cone size, n	2003	3025	1919	4704	7479	5357
Affine constraints, m	200	200	200	200	200	200
Number of cliques, p	588	1635	783	1244	6912	611
Maximum clique size	418	102	92	322	187	285
Minimum clique size	5	4	3	6	2	7



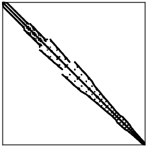
rs35



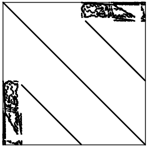
rs200



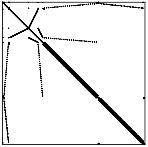
rs228



rs365



rs1555



rs1907

Large-scale sparse SDPs: Numerical results

	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	208	198	25.64	54	214	99.77

	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	38	165	65.02	151	175	63.75

	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	361	448	66.38	190	187	63.15

***: the problem could not be solved due to memory limitations.

†: maximum number of iterations reached.

Large-scale sparse SDPs: Numerical results

Average CPU time per iteration

	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	0.733	0.665	0.891

- $20\times, 21\times, 26\times$, and $75\times$ faster than SCS, respectively, for problems rs200, rs365, rs1907, and rs1555.
- The computational benefit comes form the cone decomposition (projections onto small PSD cones)
- CDCS enables us to solve large, sparse conic problems with moderate accuracy that are beyond the reach of standard interior-point and/or other first-order methods

The conic projections in all Algorithms require $\mathcal{O}(\sum_{k=1}^{p} |\mathcal{C}_k|^3)$ flops. **Complexity is dominated by the largest maximal clique!**

Part II: Decomposition in sparse SOS optimization

— bridging the gap between DSOS/SDSOS optimization and SOS optimization

Checking nonnegativity and Sum-of-squares

Checking whether a given polynomial is nonnegative has applications in many areas.

$$p(x) = \sum p_\alpha x^\alpha \geq 0, \qquad \text{e.g.,} \quad p(x) = x_1^2 + 2x_1x_2 + 2x_2^2 = (x_1 + x_2)^2 + x_2^2 \geq 0.$$

- **Application**: unconstrained polynomial optimization

$$\min_{x \in \mathbb{R}^n} p(x) \quad \Longleftrightarrow \quad \begin{aligned} \max \quad & \gamma \\ \text{subject to} \quad & p(x) - \gamma \geq 0. \end{aligned}$$

- **Sum-of-squares (SOS) relaxation**: $p(x)$ can be represented as a sum of finite squared polynomials $f_i(x), i = 1, \ldots, m$

$$p(x) = \sum_{i=1}^{m} f_i(x)^2,$$

- **SDP characterization (Parrilo 2000):** $p(x)$ is SOS if and only if there exists $Q \succeq 0$,

$$p(x) = v_d(x)^T Q v_d(x).$$

where $v_d(x) = [1, x_1, x_2, \ldots, x_n, x_1^2, x_1x_2, \ldots, x_n^d]^T$ is the standard monomial basis.

Checking nonnegativity and Sum-of-squares

Sum-of-square matrices

- Consider a symmetric matrix-valued polynomial

$$P(x) = \begin{bmatrix} p_{11}(x) & p_{12}(x) & \dots & p_{1r}(x) \\ p_{21}(x) & p_{22}(x) & \dots & p_{2r}(x) \\ \vdots & \vdots & \ddots & \vdots \\ p_{r1}(x) & p_{r2}(x) & \dots & p_{rr}(x) \end{bmatrix} \succeq 0, \forall x \in \mathbb{R}^n.$$

- Similar to the scalar case, the problem of checking whether $P(x)$ is positive semidefinite is NP-hard in general.
- **SOS relaxation:** We call $P(x)$ is an SOS matrix if

$$p(x,y) = y^T P(x) y \text{ is SOS in } [x;y]$$

- **SDP characterization (similar to the scalar case) (Parrilo et al.):** $P(x)$ is an SOS matrix if and only if there exists $Q \succeq 0$, such that

$$P(x) = (I_r \otimes v_d(x))^T Q (I_r \otimes v_d(x)).$$

where Q is called the Gram matrix.

SOS optimization

A general optimization problem:

- **Scalar version:** Consider the following real-valued SOS program

$$\begin{aligned} \min_{u} \quad & w^T u \\ \text{subject to} \quad & p_0(x) + \sum_{h=1}^{t} u_h p_h(x) \ \text{ is SOS}, \end{aligned} \tag{1}$$

  where $p_0(x), p_h(x), h = 1, \ldots, t$ are given polynomials.

- **Matrix version:** Consider the following matrix-valued SOS program

$$\begin{aligned} \min_{u} \quad & w^T u \\ \text{subject to} \quad & P_0(x) + \sum_{h=1}^{t} u_h P_h(x) \ \text{ is SOS}, \end{aligned} \tag{2}$$

  where $P_0(x), P_h(x), h = 1, \ldots, t$ are given symmetric polynomial matrices .

- Both (1) and (2) can be equivalently reformulated into SDPs;
- One fundamental problem is the poor scalability to large-scale instances, since

$$\binom{n+d}{d} = \mathcal{O}(n^d).$$

Scaled-diagonally dominant SOS (SDSOS) and DSOS

A new concept of (S)DSOS by Ahmadi and Majumdar, 2017

- *Diagonally dominant (dd) matrix:* a symmetric matrix $A = [a_{ij}]$ is dd if

$$a_{ii} \geq \sum_{j \neq i} |a_{ij}|, \forall i = 1, \ldots, n.$$

- *Scaled-diagonally dominant (sdd) matrix:* a symmetric matrix $A = [a_{ij}]$ is sdd if there exists a PSD diagonal matrix D, such that

$$DAD \text{ is dd.}$$

- *DSOS polynomials:* $p(x) = v_d(x)^T Q v_d(x)$, where the Gram matrix Q is dd.
- *SDSOS polynomials:* $p(x) = v_d(x)^T Q v_d(x)$, where the Gram matrix Q is sdd.

LP and SOCP-based optimization (Ahmadi and Majumdar, 2017)

- Optimization over dd matrices or DSOS polynomials is a linear program (LP).
- Optimization over sdd matrices or SDSOS polynomials is a second-order cone program (SOCP).

The gap between DSOS/SDSOS and SOS

A brief summary

- **SOS:** $p(x) = v_d(x)^T Q v_d(x)$, where the Gram matrix Q is PSD $\longrightarrow$ SDP
- **SDSOS:** $p(x) = v_d(x)^T Q v_d(x)$, where the Gram matrix Q is sdd $\longrightarrow$ SOCP
- **DSOS:** $p(x) = v_d(x)^T Q v_d(x)$, where the Gram matrix Q is dd $\longrightarrow$ LP

Another viewpoint

- **SDP** is an optimization problem involving PSD constraints of dimension $N \times N$
- **SOCP** is an optimization problem involving PSD constraints of dimension 2×2
- **LP** is an optimization problem involving PSD constraints of dimension 1×1

What is missing? How about problems that involve PSD constraints of dimension $k \times k$, where $1 \leq k \leq N$

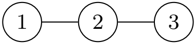
- One approach: factor-width k matrices (Boman, et al. 2005) $\longrightarrow$ Not practical $\binom{n}{k} = \mathcal{O}(n^k)$
- **Chordal decomposition**, considering sparsity and equivalent to sparse factor-width k matrices $\longrightarrow$ the main topic today.

Sparse polynomial matrix (similar to sparse real matrix)

- Given a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, we define a sparse polynomial matrix $P(x)$ where

$$p_{ij}(x) = 0, \text{ if } (i,j) \notin \mathcal{E}^*$$

- For example, for a line graph of three nodes



$$P(x) = \begin{bmatrix} p_{11}(x) & p_{12}(x) & \\ p_{21}(x) & p_{22}(x) & p_{23}(x) \\ & p_{32}(x) & p_{33}(x) \end{bmatrix}.$$

- Define a set of sparse polynomial matrices

$$\mathbb{R}^{r\times r}_{n,2d}(\mathcal{E},0) = \left\{ P(x) \in \mathbb{R}[x]^{r\times r}_{n,2d} \middle| p_{ij}(x) = p_{ji}(x) = 0, \text{ if } (i,j) \notin \mathcal{E}^* \right\}.$$

- SOS/SDSOS/DSOS matrices with a sparsity pattern $\mathcal{E}$

$$\begin{aligned} SOS^r_{n,2d}(\mathcal{E},0) &= SOS^r_{n,2d} \cap \mathbb{R}^{r\times r}_{n,2d}(\mathcal{E},0), \\ SDSOS^r_{n,2d}(\mathcal{E},0) &= SDSOS^r_{n,2d} \cap \mathbb{R}^{r\times r}_{n,2d}(\mathcal{E},0), \\ DSOS^r_{n,2d}(\mathcal{E},0) &= DSOS^r_{n,2d} \cap \mathbb{R}^{r\times r}_{n,2d}(\mathcal{E},0). \end{aligned}$$

Sparsity in SOS optimization

Sparsity in $P(x)$ does not necessarily lead to sparsity in the Gram matrix Q !!

For example

$$P(x) = \begin{bmatrix} p_{11}(x) & p_{12}(x) & \\ p_{21}(x) & p_{22}(x) & p_{23}(x) \\ & p_{32}(x) & p_{33}(x) \end{bmatrix} = \begin{bmatrix} v(x)^T Q_{11} v(x) & v(x)^T Q_{12} v(x) & v(x)^T Q_{13} v(x) \\ v(x)^T Q_{21} v(x) & v(x)^T Q_{22} v(x) & v(x)^T Q_{23} v(x) \\ v(x)^T Q_{31} v(x) & v(x)^T Q_{32} v(x) & v(x)^T Q_{33} v(x) \end{bmatrix}$$

$$= (I_3 \otimes v(x))^T \begin{bmatrix} Q_{11} & Q_{12} & Q_{13} \\ Q_{21} & Q_{22} & Q_{23} \\ Q_{31} & Q_{32} & Q_{33} \end{bmatrix} (I_3 \otimes v(x))$$

- If we make **a restriction that** $Q_{ij} = 0$**, if** $p_{ij}(x) = 0$, then the Gram matrix Q has the same pattern with $P(x)$. Now, **chordal decomposition** leads to

$$Q = \underbrace{\begin{bmatrix} * & * & 0 \\ * & * & * \\ 0 & * & * \end{bmatrix}}_{\succeq 0} = \underbrace{\begin{bmatrix} * & * & 0 \\ * & * & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{\succeq 0} + \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & * & * \\ 0 & * & * \end{bmatrix}}_{\succeq 0}$$

- We have the same chordal decomposition for polynomial matrix $P(x)$.

Sparse SOS matrix decomposition

Sparse version of SOS matrices

$$SSOS^r_{n,2d}(\mathcal{E},0) = \left\{ P(x) \in SOS^r_{n,2d}(\mathcal{E},0) \middle| P(x) \text{ admits a Gram matrix } Q \succeq 0, \text{ with } Q_{ij} = 0 \text{ when } p_{ij}(x) = 0 \right\}.$$

Theorem (Sparse SOS matrix decomposition)

If $\mathcal{E}$ is chordal with a set of maximal cliques $\mathcal{C}_1, \ldots, \mathcal{C}_t$, then

$$P(x) \in SSOS^r_{n,2d}(\mathcal{E},0) \Leftrightarrow P(x) = \sum_{k=1}^{t} E_k^T P_k(x) E_k,$$

where $P_k(x)$ is an SOS matrix of dimension $|\mathcal{C}_k| \times |\mathcal{C}_k|$.

Proof: apply the **Agler's theorem** to the sparse block matrix Q.

$$P(x) = (I_r \otimes v_d(x))^T \, Q \, (I_r \otimes v_d(x)) = (I_r \otimes v_d(x))^T \left(\sum_{k=1}^{t} E_{\widetilde{\mathcal{C}}_k}^T Q_k E_{\widetilde{\mathcal{C}}_k} \right) (I_r \otimes v_d(x))$$

$$= \sum_{k=1}^{t} \left[(I_r \otimes v_d(x))^T \, E_{\widetilde{\mathcal{C}}_k}^T Q_k E_{\widetilde{\mathcal{C}}_k} \, (I_r \otimes v_d(x)) \right] = \sum_{k=1}^{t} E_{\mathcal{C}_k}^T P_k(x) E_{\mathcal{C}_k},$$

LP/SOCP/SDP

We have the following inclusion relationship

$$DSOS^r_{n,2d}(\mathcal{E},0) \subseteq SDSOS^r_{n,2d}(\mathcal{E},0) \subseteq SSOS^r_{n,2d}(\mathcal{E},0) \subseteq SOS^r_{n,2d}(\mathcal{E},0) \subseteq \mathcal{P}^r_{n,2d}(\mathcal{E},0)$$

Key idea: if a matrix Q is (scaled) diagonally dominant, then it is still (scaled) diagonally dominant when replacing any off-diagonal elements with zeros.

- A brief summary (scalability):

$$\begin{aligned}
\mathcal{P}^r_{n,2d}(\mathcal{E},0) &\longrightarrow \text{NP-hard} \\
DSOS^r_{n,2d}(\mathcal{E},0) &\longrightarrow \text{LP (PSD cones: } 1\times 1) \\
SDSOS^r_{n,2d}(\mathcal{E},0) &\longrightarrow \text{SOCP (PSD cones: } 2\times 2) \\
SSOS^r_{n,2d}(\mathcal{E},0) &\longrightarrow \textbf{SDP with smaller PSD cones of } k\times k \\
SOS^r_{n,2d}(\mathcal{E},0) &\longrightarrow \text{SDP with a PSD cone of } N\times N
\end{aligned}$$

Solution quality: $\mathcal{P}_{\text{dsos}}, \mathcal{P}_{\text{sdsos}}$ and $\mathcal{P}_{\text{ssos}}$ are a sequence of inner approximations with increasing accuracy to the SOS problem $\mathcal{P}_{\text{sos}}$, meaning that

$$f^*_{\text{dsos}} \geq f^*_{\text{sdsos}} \geq f^*_{\text{ssos}} \geq f^*_{\text{sos}},$$

- Similar results can be shown for scalar sparse SOS optimization, which rely on the notion of *correlative sparsity pattern* (Waki *et al.*, 2006).

Implementations and numerical comparison

Packages

- SOS optimization: SOSTOOLS, YALMIP
- DSOS/SDSOS optimization: SPOTLESS
- Chordal decomposition: YALMIP (we adapted the option of correlative sparsity technique)
- SDP solver: Mosek

Numerical examples and applications

- Polynomial optimization problems
- Copositive optimization
- Control application: finding Lyapunov functions

Example 1: Polynomial optimization problems

Eigenvalue bounds on matrix polynomials

$$\begin{array}{rl} \min\limits_{\gamma} & \gamma \\ \text{subject to} & P(x) + \gamma I \succeq 0, \end{array}$$

where $n = 2, 2d = 2$, the polynomial is randomly generated. $P(x)$ has an arrow pattern.

Table: CPU time (in seconds) required by Mosek

Dimension r	10	20	30	40	50	60	70	80
SOS	0.30	1.33	6.64	27.3	108.1	308.7	541.3	1 018.6
SSOS	0.34	0.34	0.35	0.35	0.33	0.32	0.32	0.33
SDSOS	0.47	0.63	1.09	1.29	2.67	3.70	4.40	6.02
DSOS	**	**	**	**	**	**	**	**

**: The program is infeasible.

Example 1: Polynomial optimization problems

Eigenvalue bounds on matrix polynomials

$$\begin{array}{rl} \min_{\gamma} & \gamma \\ \text{subject to} & P(x) + \gamma I \succeq 0, \end{array}$$

where $n = 2, 2d = 2$, the polynomial is randomly generated. $P(x)$ has an arrow pattern.

Table: Optimal value γ

Dimension r	10	20	30	40	50	60	70	80
SOS	1.447	4.813	5.917	4.154	21.61	10.09	7.364	10.19
SSOS	1.454	4.878	5.917	4.498	21.64	12.71	7.558	11.39
SDSOS	40.1	279.3	1 254.4	145.5	762.8	1 521.1	1 217.3	598.0
DSOS	**	**	**	**	**	**	**	**

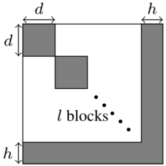
**: The program is infeasible.

Example 2: Copositive optimization

Consider the following copositive program

$$\begin{aligned} \min_{\gamma} \quad & \gamma \\ \text{subject to} \quad & Q + \gamma I \in \mathcal{C}^n, \end{aligned}$$

where Q is a random symmetric matrix with a block-arrow sparsity pattern.



Numerical results

In the simulation, the block size is $d = 3$; arrow head is $h = 2$; we vary the number of blocks l

Table: CPU time (in seconds) required by Mosek

l	2	4	6	8	10
SOS	0.45	7.34	248.9	*	*
SSOS	0.39	0.41	0.38	0.49	0.40
SDSOS	0.54	1.22	4.99	11.07	32.18
DSOS	0.59	0.76	2.19	5.72	17.11

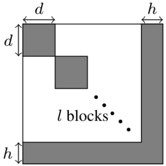
*: Out of memory.

Example 2: Copositive optimization

Consider the following copositive program

$$\begin{aligned} \min_{\gamma} \quad & \gamma \\ \text{subject to} \quad & Q + \gamma I \in \mathcal{C}^n, \end{aligned}$$

where Q is a random symmetric matrix with a block-arrow sparsity pattern.



Numerical results

In the simulation, the block size is $d = 3$; arrow head is $h = 2$; we vary the number of blocks l

Table: Optimal value γ

l	2	4	6	8	10
SOS	1.137	4.197	2.836	*	*
SSOS	1.137	4.197	2.836	4.043	4.718
SDSOS	1.184	4.500	3.282	4.562	5.146
DSOS	2.551	7.775	6.452	12.057	15.203

*: Out of memory.

Example 3: Finding Lyapunov functions

Control application: finding Lyapunov functions

- Consider a dynamical system with a banded pattern

$$\begin{aligned} \dot{x}_1 &= f_1(x_1, x_2), & g_1(x) &= \gamma - x_1^2 \geq 0 \\ \dot{x}_2 &= f_2(x_1, x_2, x_3), & g_2(x) &= \gamma - x_2^2 \geq 0 \\ &\vdots \\ \dot{x}_n &= f_n(x_{n-1}, x_n), & g_2(x) &= \gamma - x_n^2 \geq 0 \end{aligned}$$

- Generate locally stable systems of degree three;
- Consider a polynomial Lyapunov function of degree two with a banded pattern

$$V(x) = V_1(x_1, x_2) + V_2(x_1, x_2, x_3) + \ldots + V_n(x_{n-1}, x_n)$$

- Then, we consider the following SOS program

$$\begin{aligned} \text{Find} \quad & V(x), r_i(x) \\ \text{subject to} \quad & V(x) - \epsilon(x^T x) \text{ is SOS} \\ & -\langle \nabla V(x), f(x) \rangle - \sum_{i=1}^{n} r_i(x) g_i(x) \text{ is SOS} \\ & r_i(x) \text{ is SOS}, i = 1, \ldots, n. \end{aligned}$$

Example 3: Finding Lyapunov functions

Control application: finding Lyapunov functions

Table: CPU time (in seconds) required by Mosek

n	10	15	20	30	40	50
SOS	1.29	18.44	247.84	*	*	*
SSOS	0.55	0.68	0.71	0.83	1.04	1.17
SDSOS	0.71	1.76	4.47	32.21	85.99	257.20
DSOS	0.70	1.42	3.58	35.12	73.64	324.32

*: Out of memory.

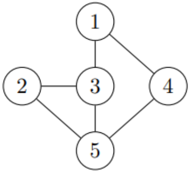
Part III - Beyond chordal decomposition

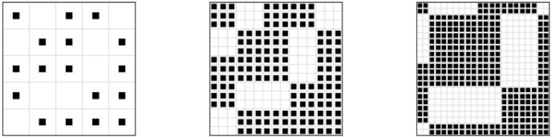
Extension 1: Block chordal decomposition

Classical chordal decomposition:

$$\underbrace{\begin{bmatrix} * & * & 0 \\ * & * & * \\ 0 & * & * \end{bmatrix}}_{\succeq 0} = \underbrace{\begin{bmatrix} * & * & 0 \\ * & * & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{\succeq 0} + \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & * & * \\ 0 & * & * \end{bmatrix}}_{\succeq 0}$$

where $*$ denotes a **real scalar number**.
Question: Does the decomposition still hold if $*$ denotes a block of arbitrary size?





(a) (b) (c)

Figure: Sparse block matrices: (a) $\alpha_1 = \{1, 1, 1, 1, 1\}$, (b) $\alpha_2 = \{3, 3, 3, 3, 3\}$, (c) $\alpha_3 = \{2, 8, 4, 6, 2\}$

Theorem: Both the decomposition and completion results hold for **sparse block matrices** with a chordal pattern!

Extension 1: Block chordal decomposition

Theorem: Both the decomposition and completion results hold for **sparse block matrices** with a chordal pattern!

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

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

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

Extension 2: PSD polynomial matrices

Question: Decomposition of PSD polynomial matrices?

$$P(x) = \begin{bmatrix} p_{11}(x) & p_{12}(x) & \\ p_{21}(x) & p_{22}(x) & p_{23}(x) \\ & p_{32}(x) & p_{33}(x) \end{bmatrix} \succeq 0, \forall x \in \mathbb{R}^n.$$

Restriction to SOS matrix: $P(x) \in SSOS^r_{n,2d}(\mathcal{E}, 0) \Leftrightarrow P(x) = \sum_{k=1}^{t} E_k^T P_k(x) E_k$, where $P_k(x)$ is an SOS matrix.

Generalization: The same chordal decomposition result holds if $P_k(x)$ is allowed to have rational function entries.

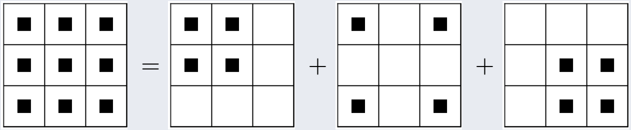
$$\underbrace{\begin{bmatrix} * & * & 0 \\ * & * & * \\ 0 & * & * \end{bmatrix}}_{\succeq 0} = \underbrace{\begin{bmatrix} * & * & 0 \\ * & * & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{\succeq 0} + \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & * & * \\ 0 & * & * \end{bmatrix}}_{\succeq 0}.$$

where $*$ denotes a rational function (i.e., a ratio of two polynomials).

Extension 3: Block factor-width two matrices

Question: How to deal with dense PSD matrices with no chordal sparsity?
Answer: One useful solution is to use an inner approximation of the PSD cone, *e.g.*, DD and SDD matrices.

(Block) factor-width two matrices via a block partition



- A new hierarchy of inner approximations of the PSD cone by varying the block partition.

  Given three partitions $\gamma = \{k_1, \ldots, k_p\}, \beta = \{l_1, \ldots, l_p\}$ and $\alpha = \{n_1, n_2\}$, where $\sum_{i=1}^{p} k_i = \sum_{i=1}^{q} l_i = n_1 + n_2 = n$ and $\alpha \sqsupseteq \beta \sqsupseteq \gamma$, we have the following inclusion:

$$SDD = \mathcal{FW}^n_{\mathbf{1},2} \subseteq \mathcal{FW}^n_{\gamma,2} \subseteq \mathcal{FW}^n_{\beta,2} \subseteq \mathcal{FW}^n_{\alpha,2} = \mathbb{S}^n_+$$

Extension 3: Block factor-width two matrices

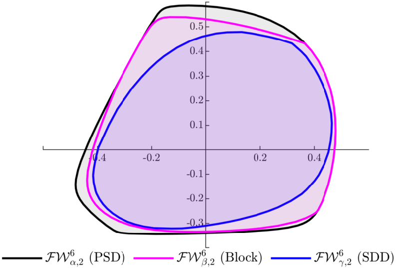


Figure: Boundary of x and y for which the 6×6 symmetric matrix $I_6 + xA + yB$ belongs to $\mathcal{FW}^6_{\alpha,2}, \mathcal{FW}^6_{\beta,2}$, and $\mathcal{FW}^6_{\gamma,2}$, where $\alpha = \{4, 2\}$, $\beta = \{2, 2, 2\}$, $\gamma = \{1, 1, 1, 1, 1, 1\}$.

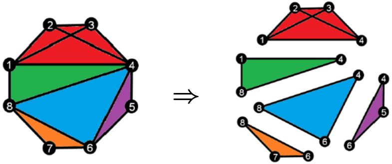
A new method to balance a trade-off between the **computation scalability** and **solution quality**!

- **More examples**: Y. Zheng, A. Sootla, and A. Papachristodoulou. "Block factor-width-two matrices and their applications to semidefinite and sum-of-squares optimization." arXiv:1909.11076 (2019).

Conclusion

Take-home message

- **Message 1: Chordal decomposition:** leading to sparse PSD cone decompositions



- **Message 2: Sparse SDPs can be solved 'fast'**

$$\begin{aligned} \min_{x, x_k} \quad & \langle c, x \rangle \\ \text{s.t.} \quad & Ax = b, \\ & \boxed{x_k = H_k x}, \quad k = 1, \ldots, p, \\ & x_k \in \mathcal{S}_k, \quad k = 1, \ldots, p, \end{aligned}$$

$$P(x) \in SSOS^r_{n,2d}(\mathcal{E}, 0)$$

$$\Longleftrightarrow P(x) = \sum_{k=1}^{t} E_k^T P_k(x) E_k,$$

CDCS: an open-source first-order conic solver;

Download from `https://github.com/OxfordControl/CDCS`

- **Message 3: Sparse SOS optimization can be solved 'fast':** Bridging the gap between DSOS/SDSOS optimization and SOS optimization.

Thank you for your attention!

Q & A

- Zheng, Y., Fantuzzi, G., Papachristodoulou, A., Goulart, P., & Wynn, A. (2019). Chordal decomposition in operator-splitting methods for sparse semidefinite programs. Mathematical Programming, 1-44.
- Zheng, Y., Fantuzzi, G., & Papachristodoulou, A. (2018, December). Decomposition and completion of sum-of-squares matrices. In 2018 IEEE Conference on Decision and Control (CDC) (pp. 4026-4031). IEEE.
- Zheng, Y., Fantuzzi, G., & Papachristodoulou, A. (2019, July). Sparse sum-of-squares (SOS) optimization: A bridge between DSOS/SDSOS and SOS optimization for sparse polynomials. In 2019 American Control Conference (ACC) (pp. 5513-5518). IEEE.
- Zheng, Y., Sootla, A., & Papachristodoulou, A. (2019). Block factor-width-two matrices and their applications to semidefinite and sum-of-squares optimization. arXiv preprint arXiv:1909.11076.