Sieve-SDP: A Simple Algorithm to Preprocess Semidefinite Programs[1]

Yuzixuan Zhu, Gábor Pataki and Quoc Tran-Dinh
University of North Carolina at Chapel Hill

THE UNIVERSITY *of* NORTH CAROLINA *at* CHAPEL HILL

Introduction

- Semidefinite Program (SDP)

$$\begin{aligned} &\inf\ C \cdot X \\ &\text{s.t. } A_i \cdot X = b_i\ (i = 1, ..., m) \\ &\quad X \succeq 0 \end{aligned}$$

  where
  - $C, A_i, X \in \mathcal{S}^n,\ b_i \in \mathbb{R},\ i = 1, ..., m$
  - $A \cdot X := \operatorname{trace}(AX) = \sum_{i,j=1}^n a_{ij} x_{ij}$
  - $X \succeq 0$: $X \in \mathcal{S}_+^n$, i.e. X is symmetric positive semidefinite
- Motivation: Solvers (SeDuMi, SDPT3, Mosek, etc.) are often
  - slow for large SDPs
  - erroneous for SDPs that are not strictly feasible
- Want an SDP preprocessor to
  - reduce problem size
  - detect lack of strict feasibility
  - improve solution accuracy

$\boxed{\text{data}} \rightarrow \boxed{\text{preprocessor}} \rightarrow \boxed{\text{clean data}} \rightarrow \boxed{\text{solver}}$

Basic Sieve-SDP Steps

Step 1. Find a constraint of the form

$$\begin{pmatrix} D_i & 0 \\ 0 & 0 \end{pmatrix} \cdot X = b_i,$$

where $b_i \leq 0$ and $D_i \succ 0$ (checked by Cholesky factorization).

Step 2. If $b_i < 0$, stop. The SDP is infeasible.

Step 3. If $b_i = 0$, delete rows and columns corresponding to D_i; remove this constraint.

Machine Precision

Sieve-SDP achieves machine precision due to the accuracy of Cholesky factorization.

Example

Before Preprocessing

$$\begin{pmatrix} 1&1&0&0\\1&2&0&0\\0&0&0&0\\0&0&0&0 \end{pmatrix} \cdot X = 0$$

$$\begin{pmatrix} 0&0&0&0\\0&0&0&1\\0&0&0&0\\0&1&0&1 \end{pmatrix} \cdot X = 0$$

$$\begin{pmatrix} 0&0&0&0\\0&0&2&0\\0&2&1&0\\0&0&0&0 \end{pmatrix} \cdot X = 1$$

$$X \in \mathcal{S}_+^4$$

$\begin{pmatrix} 1&1\\1&2 \end{pmatrix} \succ 0$ in constraint 1 $\Longrightarrow$

1st Reduction

$$\begin{pmatrix} 1&1&0&0\\1&2&0&0\\0&0&0&0\\0&0&0&0 \end{pmatrix} \cdot X = 0, \quad \text{removed}$$

$$\begin{pmatrix} 0&0&0&0\\0&0&0&1\\0&0&0&0\\0&1&0&1 \end{pmatrix} \cdot X = 0$$

$$\begin{pmatrix} 0&0&0&0\\0&0&2&0\\0&2&1&0\\0&0&0&0 \end{pmatrix} \cdot X = 1$$

$$X \in \mathcal{S}_+^4$$

$1 \succ 0$ in constraint 2 $\Downarrow$

2nd Reduction

$$\begin{pmatrix} 0&0\\0&1 \end{pmatrix} \cdot X = 0, \quad \text{removed}$$

$$\begin{pmatrix} 1&0\\0&0 \end{pmatrix} \cdot X = 1$$

$$X \in \mathcal{S}_+^2$$

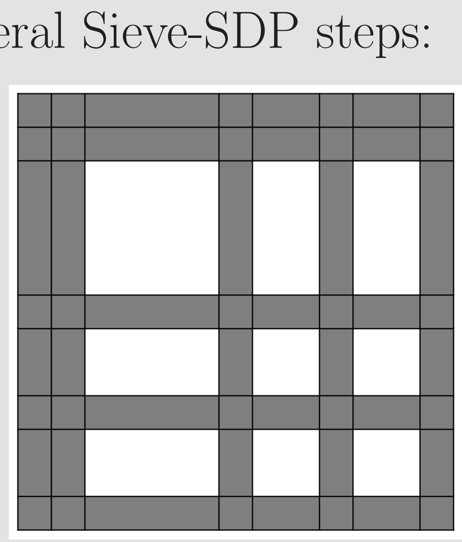
$\Longleftarrow$

After Preprocessing

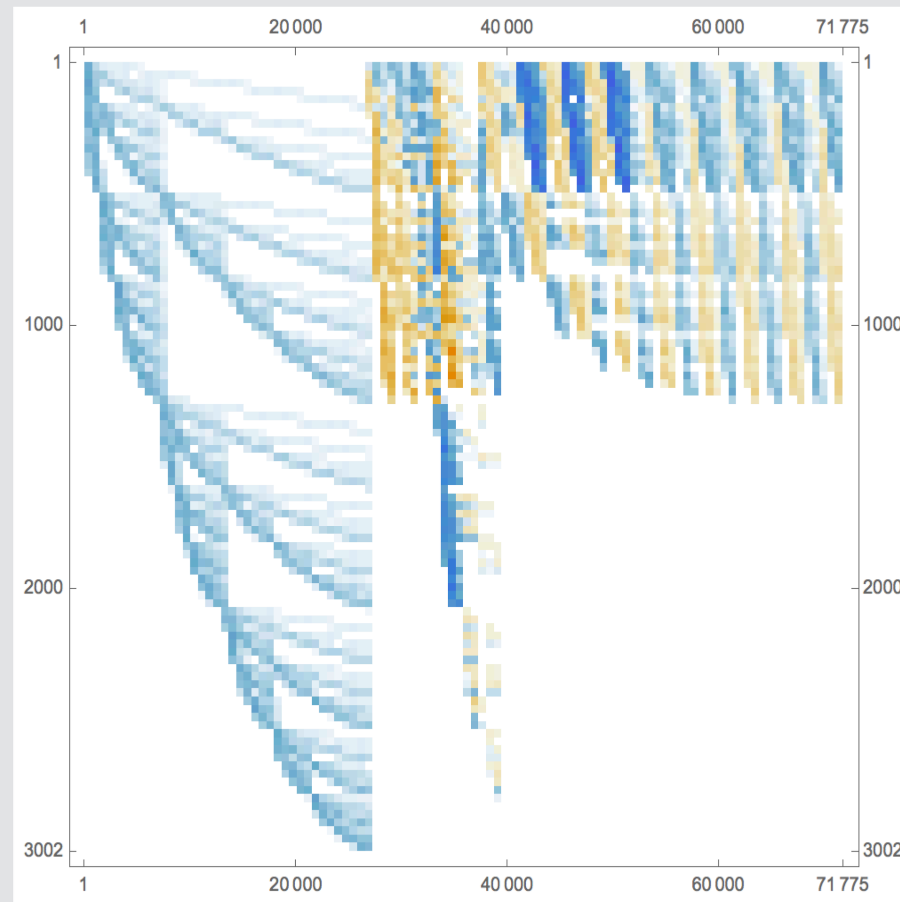
$$1 \cdot X = 1$$
$$X \geq 0$$

The Sieve Structure

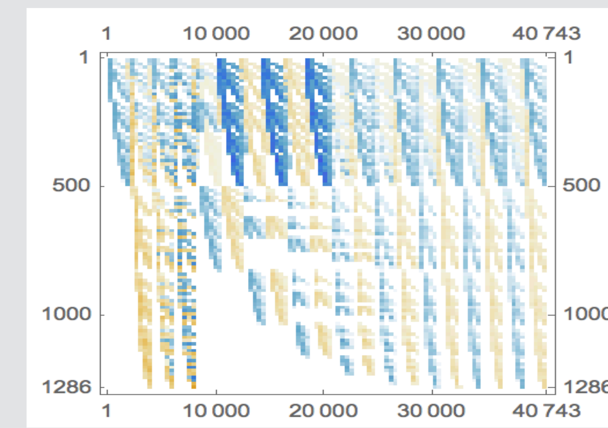
Matrix structure after several Sieve-SDP steps:



Large Example



(a) An SDP with $\sum n_i^2 = 71775$ and $m = 3002$



(b) Reduced SDP with $\sum n_i^2 = 40743$ and $m = 1286$

Computational Experiments: Setup

- 12 datasets from Permenter-Parrilo[2], 1 dataset from Henrion-Toh, and 8 datasets from Mittelmann[3]: 197 problems in total.
- Each problem is preprocessed by Sieve-SDP and each of the four Permenter-Parrilo preprocessing methods (pd1, pd2, dd1 and dd2)[2].
- Each problem is solved by Mosek 8.0 before and after each preprocessing method, then their solution qualities are compared.[a]

[a]Matlab R2015a on MacBook Pro with 2.7 GHz Intel Core and 8GB RAM

Computational Experiments: Summary

Problem Size Reduction and Preprocessing Time

method	n	red$_n$	m	red$_m$	time$_{\text{prep}}$ (s)	time$_{\text{prep}}$/time$_{\text{sol}}$ [a]
w/o prep.	219671	0.00%	522603	0.00%	0.00	0.00%
pd1	216227	1.57%	486554	6.90%	695.60	0.62%
pd2	215823	1.75%	481126	7.94%	8607.75	7.69%
dd1	195471	11.02%	522603	0.00%	488.03	0.44%
dd2	195330	11.08%	522603	0.00%	12069.64	10.78%
Sieve	212002	3.49%	451350	13.63%	869.03	0.80%

Solution Quality Improvement

method	# reduced	# infeas. detected	# err. improved	# obj. corrected
pd1	54	12	8	11, 11
pd2	75	12	11	11, 11
dd1	14	0	3	1, 5
dd2	21	0	6	1, 5
Sieve	61	14	8	11, 11

[a]solving time before preprocessing is time$_{\text{sol}}$ = 31.09 hrs

Computational Experiments: Case Study on "Example"[4]

Primal and Dual Objective Values Improvements after Preprocessing

problem	correct	w/o prep.	after pd1/pd2	after dd1/dd2	after Sieve
1	0, 0	0, 0	0, 0	0, 0	0, 0
2	1, 0	0.33, 0.33	1, 1	0.00, 0.00	1, 1
3	0, 0	0.33, 0.33	0.00, 0.00	0.00, 0.00	0.00, 0.00
4	infeas, 0	0, 0.00[a]	0, 1	0, 0	infeas, -
6	1, 1	1, 1	1, 1	1, 1	1, 1
7	0, 0	0, 0	0, 0	0, 0	0, 0
9a	infeas, 0	0, 0.34	0, 1	0, 0	infeas, -
9b	infeas, 0	0, 0.34	0, 1	0, 0	infeas, -
correct%	100%, 100%	38%, 38%	63%, 50%	50%, 100%	100%, 50%

[a]This solution has too large a DIMACS error to be regarded as correct.

Conclusions: Advantages of Sieve-SDP

- Reduces size of SDPs and detects infeasibility efficiently
- Reduces solving effort and improve solution accuracy
- Simple to understand and easy to implement (60 lines of Matlab code)
- Within machine precision
- Does not depend on any optimization solver

[1] Yuzixuan Zhu, Gábor Pataki, and Quoc Tran-Dinh. "Sieve-SDP: a simple facial reduction algorithm to preprocess SDPs". *arXiv preprint arXiv:1710.08954* (2017). url: `https://github.com/unc-optimization/SieveSDP`
[2] Permenter, Frank, and Pablo Parrilo. "Partial facial reduction: simplified, equivalent SDPs via approximations of the PSD cone." *Mathematical Programming* (2014): 1-54. `www.mit.edu/~fperment`
[3] `http://plato.asu.edu/ftp/sdp/`
[4] Cheung, Yuen-Lam, Simon Schurr, and Henry Wolkowicz. "Preprocessing and regularization for degenerate semidefinite programs" *Computational and analytical mathematics*. Springer, New York, NY, 2013. 251-303.