Sieve: a simple preprocessing algorithm for semidefinite programming

Yuzixuan Zhu

Joint work with Gábor Pataki and Quoc Tran-Dinh

University of North Carolina at Chapel Hill

ISMP, July 2018

Outline

- Basic concepts
- Examples
- The Sieve Algorithm
- Computational Results

Semidefinite Program (SDP)

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

where

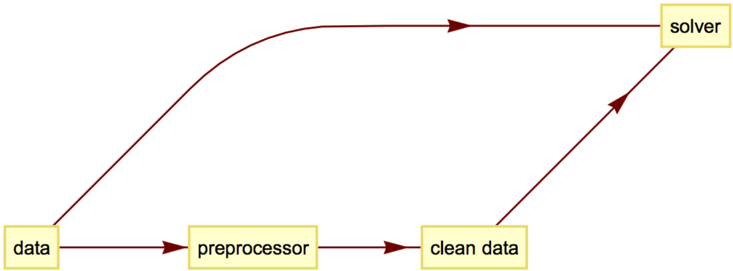
- $A_i, X \in \mathcal{S}^n,\ b_i \in \mathbb{R},\ i = 1, ..., m$
- $A \cdot X := \text{trace}(AX) = \sum_{i,j=1}^n a_{ij} x_{ij}$
- $X \succeq 0$: $X \in \mathcal{S}^n_+$, i.e. X is symmetric positive semidefinite (psd)

Motivation

We want to

- Reduce problem size by removing redundancy
- Detect lack of strict feasibility

in the preprocessing stage.



Example 1

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

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

$$X \succeq 0$$

Example 1

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

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

$$X \succeq 0$$

Suppose $X = (x_{ij})_{3\times 3}$ feasible $\Rightarrow x_{11} = 0$

$\Rightarrow x_{12} = x_{13} = 0$

Example 1

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

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

$$X \succeq 0$$

$$\text{Suppose } X = (x_{ij})_{3\times 3} \text{ feasible } \Rightarrow x_{11} = 0$$

$$\Rightarrow x_{12} = x_{13} = 0$$

$$\Rightarrow x_{22} = -1$$

$$\Rightarrow \text{Infeasible!}$$

Example 2

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

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

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

$$X \succeq 0$$

Example 2

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

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

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

$$X \succeq 0$$

Example 2

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

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

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

$$X \succeq 0$$

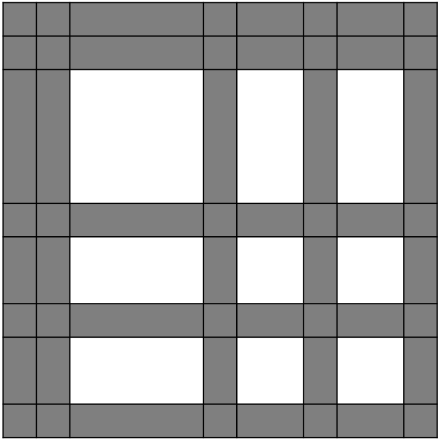
Example 2

Before preprocessing: $X \in \mathcal{S}^4_+$; 3 constraints

After preprocessing: $X \in \mathcal{S}^1_+$; 1 constraint: $1 \cdot X = 1$

The Sieve Structure

After reduction, the matrix looks like this:



Basic 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.

Safe mode

Fix $\epsilon = 2.2204 \times 10^{-16}$.

- $D_i \succ 0$? Check whether $D_i - \sqrt{\epsilon} I \succ 0$
- $b_i < 0$? Check whether $b_i < -\sqrt{\epsilon}\max\{||b_i||_\infty, 1\}$
- $b_i = 0$? Check whether $b_i > -\epsilon\max\{||b_i||_\infty, 1\}$

Permenter-Parrilo (PP) preprocessing methods[2]

- PP reduces the size of an SDP by solving linear programming (LP) subproblems
- Implemented for primal (p-) and dual (d-) SDPs
- Implemented using diagonal (-d1) and diagonally dominant (-d2) approximations
- We use Mosek[1] as the LP subproblems solver

[1] **Mosek:14**.
[2] **PerPar:14**.

Problem sets

21 datasets consisting of 197 SDP problems:

- 12 datasets from Permenter-Parrilo collection[3]
- 1 dataset from D. Henrion and K. C. Toh
- 8 datasets from http://plato.asu.edu/ftp/sdp/

[3] **PerPar:14**.

Computational setup

- Preprocess using Sieve and 4 PP methods (pd1, pd2, dd1, dd2)
- Use Mosek to solve each problem before and after preprocessing
- Matlab R2015a on MacBook Pro with OS X Yosemite 10.10.5
- Processor: 2.7 GHz Intel Core i5
- Memory: 8GB 1867 MHz DDR3

Comparison criteria

- Does preprocessing reduce a problem?
- Does it help to detect infeasibility?
- Does it reduce DIMACS errors[4]?
- Does it help to recover the true objective value?

[4] **DIMACS**.

Help codes

We set the help code to be

- **1**, if preprocessing detects (or help detect) infeasibility
- **-1**, if solver detects infeasibility before preprocessing, but does not detect infeasibility after preprocessing
- **2**, if

$$\text{Error}_{\text{before}} > 10^{-6} \quad \text{and} \quad \frac{\text{Error}_{\text{after}}}{\text{Error}_{\text{before}}} < 0.1$$

- **-2**, if

$$\text{Error}_{\text{after}} > 10^{-6} \quad \text{and} \quad \frac{\text{Error}_{\text{after}}}{\text{Error}_{\text{before}}} > 10$$

- **3**, if

$$\frac{|\text{obj}_{\text{after}} - \text{obj}_{\text{before}}|}{1 + |\text{obj}_{\text{before}}|} > 10^{-6}$$

Recover true objective values?

Problem set "Compact"[5]

problem	correct	w/o prep.	after pd1/pd2	after dd1/dd2	after Sieve
1	Infeas, $+\infty$	0.00, 0.00	0, 1	0.00, 0.00	Infeas, -
2	Infeas, $+\infty$	0.00, 0.00	Infeas, $+\infty$	0.00, 0.00	Infeas, -
3	Infeas, $+\infty$	1.5, 1.5	Infeas, $+\infty$	1.5, 1.5	Infeas, -
4	Infeas, $+\infty$	1.5, 1.5	Infeas, $+\infty$	1.5, 1.5	Infeas, -
5	Infeas, $+\infty$	1.5, 1.5	Infeas, $+\infty$	1.5, 1.5	Infeas, -
6	Infeas, $+\infty$	1.5, 1.5	Infeas, $+\infty$	1.5, 1.5	Infeas, -
7	Infeas, $+\infty$	1.5, 1.5	Infeas, $+\infty$	1.5, 1.5	Infeas, -
8	Infeas, $+\infty$	1.5, 1.5	Infeas, $+\infty$	1.5, 1.5	Infeas, -
9	Infeas, $+\infty$	1.5, 1.5	Infeas, $+\infty$	1.5, 1.5	Infeas, -
10	Infeas, $+\infty$	1.5, 1.5	Infeas, $+\infty$	1.5, 1.5	Infeas, -
correct%	100%, 100%	0%, 0%	90%, 90%	0%, 0%	100%, -

[5] **Waki:12**.

Recover true objective values?

Problem set "Example"[6]

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*	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%

[6] **CheWolkSchurr:12**.

Recover true objective values?

Problem set "unbounded"[7]

problem	correct	w/o prep.	after pd1/pd2	after dd1/dd2	after Sieve
1	0, 0	0.00, 0.00	0.00, 0.00	0.00, 0.00	0, 0
2	0, 0	0.00*, 0.00*	0, 0	0.00*, 0.00*	0, 0
3	0, 0	0.00*, 0.00*	0, 0	0.00*, 0.00*	0, 0
4	0, 0	0.00*, 0.00*	0, 0	0.00*, 0.00*	0, 0
5	0, 0	-1, -1	0, 0	-1, -1	0, 0
6	0, 0	-1, -1	0, 0	-1, -1	0, 0
7	0, 0	-1, -1	0, 0	-1, -1	0, 0
8	0, 0	-1, -1	0, 0	-1, -1	0, 0
9	0, 0	-1, -1	0, 0	-1, -1	0, 0
10	0, 0	-1, -1	0, 0	-1, -1	0, 0
correct%	100%, 100%	10%, 10%	100%, 100%	10%, 10%	100%, 100%

[7] **waki2012strange**.

Overall summary: reduction

197 problems in total

$$\text{reduction rate on } n\text{:}\ \frac{\sum n_{\text{before}} - \sum n_{\text{after}}}{\sum n_{\text{before}}}$$

$$\text{reduction rate on } m\text{:}\ \frac{\sum m_{\text{before}} - \sum m_{\text{after}}}{\sum m_{\text{before}}}$$

	# problems	reduction rate on n	reduction rate on m	added # free vars
pd1	54	1.57%	6.90%	0
pd2	75	1.75%	7.94%	0
dd1	14	11.02%	0.00%	2293495
dd2	21	11.08%	0.00%	2315849
Sieve	61	3.49%	13.63%	0

Overall summary: help or hurt

± 1: helped/hurt in detecting infeasibility
± 2: helped/hurt DIMACS error
$+3$: helped improve objective value

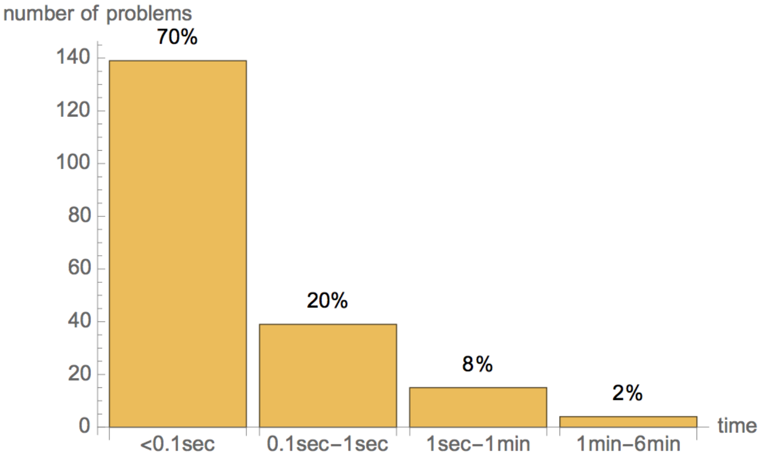
probems	1	-1	2	-2	3
pd1	12	0	8	0	13
pd2	12	0	11	0	17
dd1	0	2	3	1	5
dd2	0	2	6	1	6
Sieve	14	0	8	1	20

Overall summary: time

Solving time before preprocessing: 111904.02sec (31.09hrs)

	preprocessing time (sec)	$\frac{\text{preprocessing time}}{\text{solving time before preprocessing}} \times 100\%$
pd1	695.60	0.62%
pd2	8607.75	7.69%
dd1	488.03	0.44%
dd2	12069.64	10.78%
Sieve	869.03	0.80%

High speed of Sieve



Conclusion

Advantages of Sieve:

- Simple to understand and implement; 50 lines of Matlab code
- Machine precision using safe mode
- Reduces size of SDPs and detects infeasibility efficiently
- Does not depend on any optimization solver
- Very fast and stable

Paper and Code

- **zhu2017sieve**
- **github-sieve**

Thank you!