Sieve-SDP: A Simple Algorithm to Preprocess Semidefinite Programs

Yuzixuan Zhu

Joint work with Gábor Pataki and Quoc Tran-Dinh

University of North Carolina at Chapel Hill

SIAM Annual Meeting, July 2018

Outline

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

Semidefinite Program (SDP)

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

where

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

Motivation

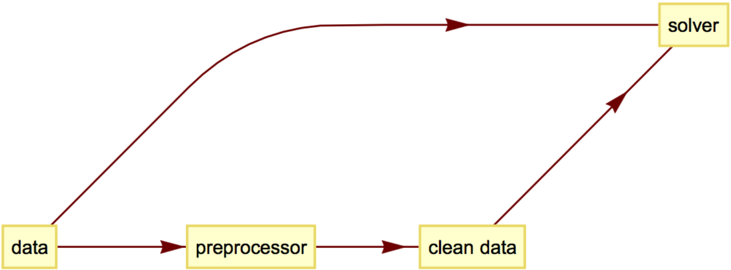
Softwares: SeDuMi, SDPT3, Mosek, ...

- Slow for problems that are large
- Error for problems without strict feasibility

We want to preprocess the problem to

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

before giving the problem to the solver.



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

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

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

$$X \succeq 0$$

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

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

Example 1

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

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

$$X \succeq 0$$

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

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

$\Rightarrow x_{22} = -1$

$\Rightarrow$ 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

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

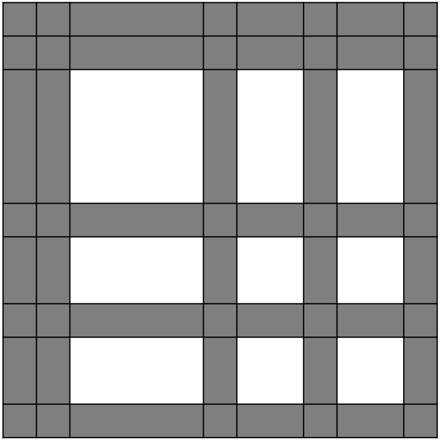
Example 2

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

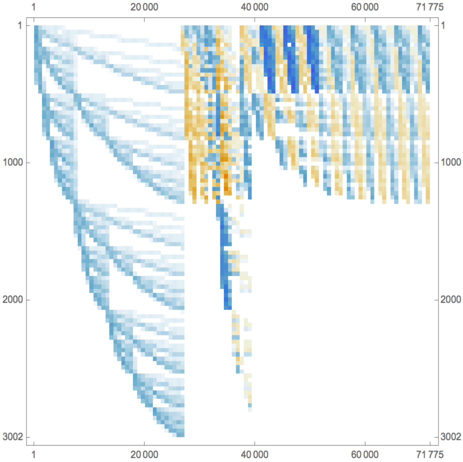
After preprocessing: $X \in \mathcal{S}_+^1$; 1 constraint: $1 \cdot X = 1$

The Sieve structure

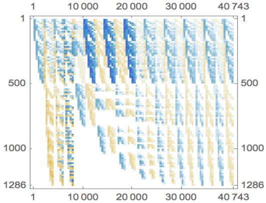
After reduction, the matrix looks like this:



A large example



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



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

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 $b_i > -\epsilon \max\{||b_i||_\infty, 1\}$

Sieve-SDP is a facial reduction algorithm (FDA)[1][2][3]

- The feasible region of an SDP is

$$\{X \in \mathcal{S}_+^n : \ A_i \cdot X = b_i, \ i = 1, ..., m\},$$

which is equivalent to

$$\{X \in F : \ A_i \cdot X = b_i, \ i = 1, ..., m\}$$

for some F face of $\mathcal{S}_+^n$.
- FDA iterates to reduce the cone ($F_{k+1} \subseteq F_k \subseteq \cdots \subseteq \mathcal{S}_+^n$).

[1] **borwein1981facial**.
[2] **waki2013facial**.
[3] **pataki2013strong**.

Permenter-Parrilo (PP) preprocessing methods[4]

- PP reduces the size of an SDP by solving linear programming subproblems
- Implemented for primal (p-) and dual (d-) SDPs
- Implemented using diagonal (-d1) and diagonally dominant (-d2) approximations

[4] **PerPar:14**.

Problem sets

Table: 5 datasets consisting of 771 SDP problems.

dataset	source	# problems
Permenter-Parrilo (PP)	[[**permenter2014partial**]]	68
Mittelmann	Mittelmann website	31
Dressler-Illiman-de Wolff (DIW)	[[**dressler2019approach**]]	155
Henrion-Toh	Didier Henrion and Kim-Chuan Toh	98
Toh-Sun-Yang	[[**sun2015convergent**, **yang2015sdpnal**]]	419
total		771

Computational setup

Computational setup

- Preprocess using Sieve-SDP and four PP methods (pd1, pd2, dd1, dd2).

Computational setup

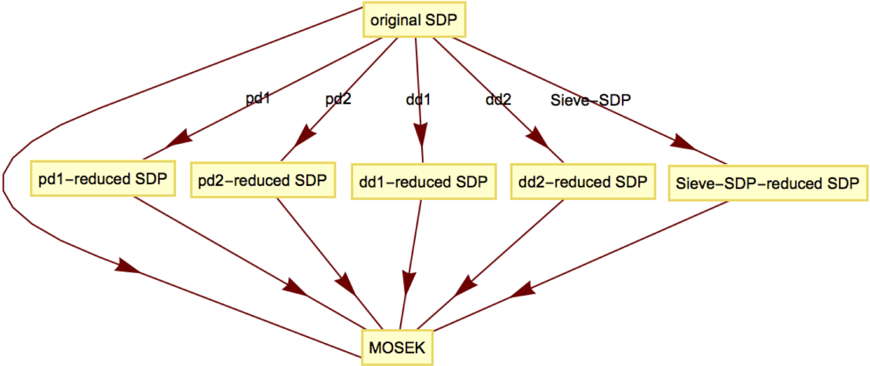
- Preprocess using Sieve-SDP and four PP methods (pd1, pd2, dd1, dd2).
- Use MOSEK [[**mosek2017mosek**]] to solve each problem before and after preprocessing.

Computational setup

- Preprocess using Sieve-SDP and four PP methods (pd1, pd2, dd1, dd2).
- Use MOSEK [[**mosek2017mosek**]] to solve each problem before and after preprocessing.
- MATLAB R2015a on MacBook Pro with 8GB of RAM.

Computational setup

- Preprocess using Sieve-SDP and four PP methods (pd1, pd2, dd1, dd2).
- Use MOSEK [[**mosek2017mosek**]] to solve each problem before and after preprocessing.
- MATLAB R2015a on MacBook Pro with 8GB of RAM.



Comparison criteria

[5] http://archive.dimacs.rutgers.edu/Challenges/Seventh/Instances/

Comparison criteria

- Does preprocessing reduce a problem?

[5] `http://archive.dimacs.rutgers.edu/Challenges/Seventh/Instances/`

Comparison criteria

- Does preprocessing reduce a problem?
- Does it help to detect infeasibility?

[5] `http://archive.dimacs.rutgers.edu/Challenges/Seventh/Instances/`

Comparison criteria

- Does preprocessing reduce a problem?
- Does it help to detect infeasibility?
- Does it help to recover the true objective value?

[5] `http://archive.dimacs.rutgers.edu/Challenges/Seventh/Instances/`

Comparison criteria

- Does preprocessing reduce a problem?
- Does it help to detect infeasibility?
- Does it help to recover the true objective value?
- Does it reduce solution inaccuracy defined by DIMACS errors[5]?

[5] `http://archive.dimacs.rutgers.edu/Challenges/Seventh/Instances/`

Comparison criteria

- Does preprocessing reduce a problem?
- Does it help to detect infeasibility?
- Does it help to recover the true objective value?
- Does it reduce solution inaccuracy defined by DIMACS errors[5]?
- Does it reduce solving time?

[5] `http://archive.dimacs.rutgers.edu/Challenges/Seventh/Instances/`

Recover true objective values?

Table: Objective values (P, D) on the “Compact” problems [[**waki2012generate**]].

problem	correct	w/o prep.	after pd1/pd2	after dd1/dd2	after Sieve-SDP
CompactDim2R1	Infeas, $+\infty$	3.79e+06, 4.20e+06	Infeas, 1	3.79e+06, 4.20e+06	Infeas, -
CompactDim2R2	Infeas, $+\infty$	6.41e-10, 6.81e-10	Infeas, 2	6.41e-10, 6.81e-10	Infeas, -
CompactDim2R3	Infeas, $+\infty$	1.5, 1.5	Infeas, 2	1.5, 1.5	Infeas, -
CompactDim2R4	Infeas, $+\infty$	1.5, 1.5	Infeas, 2	1.5, 1.5	Infeas, -
CompactDim2R5	Infeas, $+\infty$	1.5, 1.5	Infeas, 2	1.5, 1.5	Infeas, -
CompactDim2R6	Infeas, $+\infty$	1.5, 1.5	Infeas, 2	1.5, 1.5	Infeas, -
CompactDim2R7	Infeas, $+\infty$	1.5, 1.5	Infeas, 2	1.5, 1.5	Infeas, -
CompactDim2R8	Infeas, $+\infty$	1.5, 1.5	Infeas, 2	1.5, 1.5	Infeas, -
CompactDim2R9	Infeas, $+\infty$	1.5, 1.5	Infeas, 2	1.5, 1.5	Infeas, -
CompactDim2R10	Infeas, $+\infty$	1.5, 1.5	Infeas, 2	1.5, 1.5	Infeas, -
correctness %	100%, 100%	0%, 0%	100%, 0%	0%, 0%	100%, -

SDP relaxation for polynomial optimization

- Polynomial optimization:

$$\begin{aligned} \min_{x \in {}^N} \quad & f_0(x) \\ \text{s.t.} \quad & f_i(x) \geq 0, \quad i = 1, ..., r. \end{aligned} \tag{poly-opt}$$

SDP relaxation for polynomial optimization

- Polynomial optimization:

$$\begin{aligned} \min_{x \in ^N} \quad & f_0(x) \\ \text{s.t.} \quad & f_i(x) \geq 0, \quad i = 1, ..., r. \end{aligned} \qquad \text{(poly-opt)}$$

- It is equivalent to

$$\begin{aligned} \max_{\gamma \in} \quad & \gamma \\ \text{s.t.} \quad & f_0(x) + \textstyle\sum_{i=1}^r s_i(x) f_i(x) - \gamma = s_0(x), \quad \forall x \in^N, \end{aligned}$$

where $s_i(x),\ i = 0, 1, ..., r$ are sum-of-square polynomials [[**lasserre2001global**]].

SDP relaxation for polynomial optimization

- Polynomial optimization:

$$\begin{array}{ll} \min_{x \in^N} & f_0(x) \\ \text{s.t.} & f_i(x) \geq 0, \quad i = 1, ..., r. \end{array} \qquad \text{(poly-opt)}$$

- It is equivalent to

$$\begin{array}{ll} \max_{\gamma \in} & \gamma \\ \text{s.t.} & f_0(x) + \sum_{i=1}^{r} s_i(x) f_i(x) - \gamma = s_0(x), \quad \forall x \in^N, \end{array}$$

  where $s_i(x),\ i = 0, 1, ..., r$ are sum-of-square polynomials [[**lasserre2001global**]].

- It has SDP relaxation:

$$\begin{array}{ll} \min_{\gamma \in} & -\gamma, \\ \text{s.t.} & Q \succeq 0, \end{array} \qquad \text{(SDP-relaxation)}$$

  where $Q \in \mathcal{S}^n$ is based on γ and coefficients of $f_i,\ i = 0, 1, ..., r$.

SDP relaxation for polynomial optimization

- Polynomial optimization:

$$\begin{aligned} \min_{x\in^N} \quad & f_0(x) \\ \text{s.t.} \quad & f_i(x) \geq 0, \quad i = 1, ..., r. \end{aligned} \tag{poly-opt}$$

- It is equivalent to

$$\begin{aligned} \max_{\gamma\in} \quad & \gamma \\ \text{s.t.} \quad & f_0(x) + \textstyle\sum_{i=1}^r s_i(x) f_i(x) - \gamma = s_0(x), \quad \forall x \in^N, \end{aligned}$$

  where $s_i(x),\ i = 0, 1, ..., r$ are sum-of-square polynomials [[**lasserre2001global**]].
- It has SDP relaxation:

$$\begin{aligned} \min_{\gamma\in} \quad & -\gamma, \\ \text{s.t.} \quad & Q \succeq 0, \end{aligned} \tag{SDP-relaxation}$$

  where $Q \in \mathcal{S}^n$ is based on γ and coefficients of $f_i,\ i = 0, 1, ..., r$.
- Infeasibility of (SDP-relaxation) gives a useless lower bound $\gamma = -\infty$ to (poly-opt).

SDP relaxation for polynomial optimization

- Polynomial optimization:

$$\begin{aligned} \min_{x \in^N} \quad & f_0(x) \\ \text{s.t.} \quad & f_i(x) \geq 0, \quad i = 1, ..., r. \end{aligned} \tag{poly-opt}$$

- It is equivalent to

$$\begin{aligned} \max_{\gamma \in} \quad & \gamma \\ \text{s.t.} \quad & f_0(x) + \textstyle\sum_{i=1}^r s_i(x) f_i(x) - \gamma = s_0(x), \quad \forall x \in^N, \end{aligned}$$

  where $s_i(x), \; i = 0, 1, ..., r$ are sum-of-square polynomials [[**lasserre2001global**]].

- It has SDP relaxation:

$$\begin{aligned} \min_{\gamma \in} \quad & -\gamma, \\ \text{s.t.} \quad & Q \succeq 0, \end{aligned} \tag{SDP-relaxation}$$

  where $Q \in \mathcal{S}^n$ is based on γ and coefficients of $f_i, \; i = 0, 1, ..., r$.

- Infeasibility of (SDP-relaxation) gives a useless lower bound $\gamma = -\infty$ to (poly-opt).
- Without knowing the infeasibility of (SDP-relaxation), the effort to solving it could be tremendous.

Results of DIW dataset (polynomial optimization problems)

Table: Results of DIW dataset.

prep. method	# reduced	# infeas detected	n	m	t_{prep} (s)	t_{sol} (s)
w/o prep.	-	-	53,523	186,225		(39 hrs $\approx$) 139,493.56
pd1	155	56	1,450	3,278	1,615.02	128.46
pd2	155	56	1,450	3,278	10,831.32	124.44
dd1	0	0	53,523	186,225	48.32	139,493.56
dd2	0	0	53,523	186,225	22,135.71	139,493.56
Sieve-SDP	155	59	1,385	3,204	1,232.27	(1.5 min $\approx$) 87.53

- Increased the solving speed by more than 100 times!
- Infeasibility has been double-checked manually.

An example from DIW dataset

Figure: Size and sparsity before and after Sieve-SDP.

Overall summary on all 771 problems: size reduction

Table: Overall size reduction.

method	# reduced	red. on n	red. on m	extra free vars
pd1	209	15.47%	17.79%	0
pd2	230	15.59%	18.23%	0
dd1	14	6.74%	0.00%	2,293,495
dd2	21	9.28%	0.00%	2,315,849
Sieve-SDP	216	16.55%	20.66%	0

red. on n: $\frac{\sum n_{\text{before}} - \sum n_{\text{after}}}{\sum n_{\text{before}}}$

red. on m: $\frac{\sum m_{\text{before}} - \sum m_{\text{after}}}{\sum m_{\text{before}}}$

Overall summary on all 771 problems: helpfulness

Table: Overall helpfulness.

method	# reduced	# infeas detected	# DIMACS error improved	# out of memory
pd1	209	67	74	0
pd2	230	67	78	6
dd1	14	0	2	0
dd2	21	0	4	4
Sieve-SDP	216	73	74	0

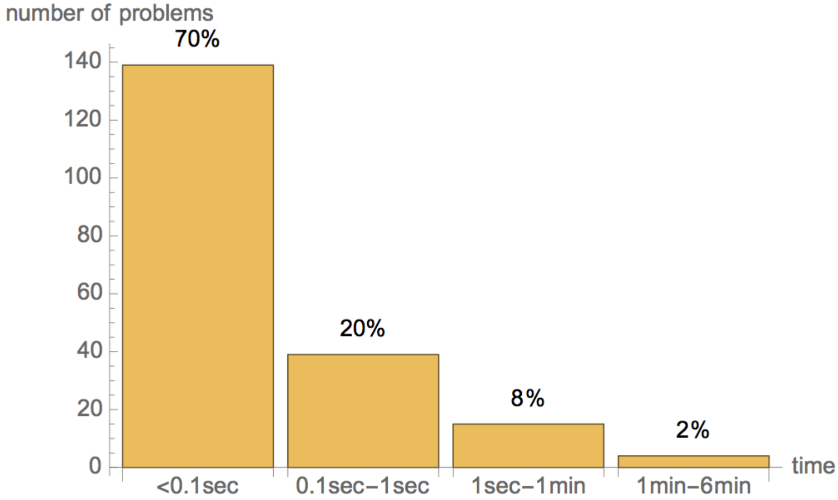
Overall summary on all 771 problems: time

Table: Preprocessing and solving times.

method	t_{prep} (hr)	t_{sol} (hr)	$t_{prep}/t_{sol\_w/o}$	time reduction
w/o	-	75.67	-	-
pd1	0.69	36.77	0.91%	50.50%
pd2	6.48	36.57	8.56%	43.12%
dd1	0.16	75.62	0.22%	-0.15%
dd2	10.00	75.56	13.21%	-13.16%
Sieve-SDP	0.60	36.62	0.80%	51.81%

$$\text{time reduction: } \frac{t_{sol\_w/o} - (t_{prep} + t_{sol})}{t_{sol\_w/o}} \times 100\%.$$

High speed of Sieve-SDP



Highlights of Sieve-SDP

- Simple to understand and implement
- Run in machine precision under safe mode
- Reduces size of SDPs and detects infeasibility efficiently
- Does not depend on any optimization solver
- Very fast and stable

Overall summary: reduction

197 problems in total

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

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

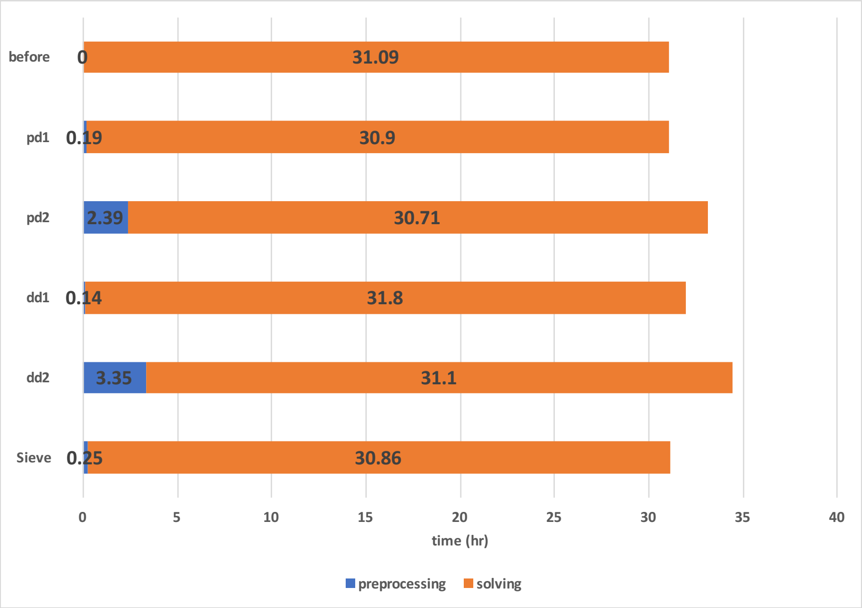
	reduction rate on n	reduction rate on m	added # free vars
pd1	1.57%	6.90%	0
pd2	1.75%	7.94%	0
dd1	11.02%	0.00%	2293495
dd2	11.08%	0.00%	2315849
Sieve	3.49%	13.63%	0

Overall summary: help or hurt

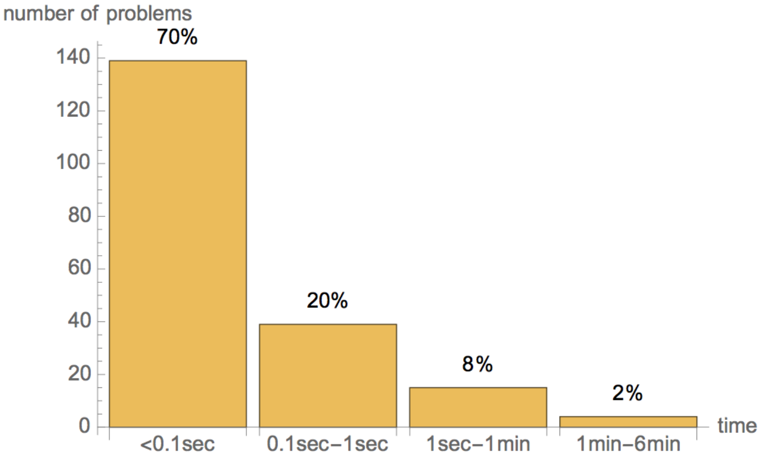
- +1: detected infeasibility
- −1: did not detect infeasibility even though solver detected infeasibility
- +2: reduced DIMACS error
- −2: increased DIMACS error
- +3: improved objective value
- −4: ran out of memory

methods	reduced	1	-1	2	-2	3	-4
pd1	54	12	0	8	0	13	0
pd2	75	12	0	11	0	17	6
dd1	14	0	2	3	1	5	0
dd2	21	0	2	6	1	6	4
Sieve	61	14	0	8	1	20	0

Overall summary: time



High speed of Sieve-SDP



Advantages of Sieve-SDP

- Simple to understand and implement
- 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

- **Paper**: **zhu2017sieve**
- **Code**: **github-sieve**
- Try Sieve-SDP in your research, and share your experience with me: zyzx@live.unc.edu.

Thank you for your attention!