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)

\[ \inf \ C \cdot X \]
\[ \text{s.t. } A_i \cdot X = b_i \ (i = 1, \ldots, m) \]
\[ X \succeq 0 \]

where
- \( C, A_i, X \in S^n \), \( b_i \in \mathbb{R} \), \( i = 1, \ldots, m \)
- \( A \cdot X := \text{trace}(AX) = \sum_{i,j=1}^n a_{ij} x_{ij} \)
- \( X \succeq 0: X \in S^n_+ \), 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

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

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

\[X \succeq 0\]

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

\[\Rightarrow x_{12} = x_{13} = 0\]
Example 1

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

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

\[X \succeq 0\]

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

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

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

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

\[X \geq 0\]
Example 2

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

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

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

\[X \succeq 0\]
Example 2

Before preprocessing: \( X \in S^4_+; \) 3 constraints

After preprocessing: \( X \in S^1_+; \) 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)\textsuperscript{123}

- The feasible region of an SDP is

\[
\{ X \in S^n_+ : A_i \cdot X = b_i, \ i = 1, \ldots, m \},
\]

which is equivalent to

\[
\{ X \in F : A_i \cdot X = b_i, \ i = 1, \ldots, m \}
\]

for some $F$ face of $S^n_+$.

- FDA iterates to reduce the cone ($F_{k+1} \subseteq F_k \subseteq \cdots \subseteq S^n_+$).

\textsuperscript{1}borwein1981facial.
\textsuperscript{2}waki2013facial.
\textsuperscript{3}pataki2013strong.
Permenter-Parrilo (PP) preprocessing methods

- 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
## Problem sets

Table: 5 datasets consisting of 771 SDP problems.

<table>
<thead>
<tr>
<th>dataset</th>
<th>source</th>
<th># problems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permenter-Parrilo (PP)</td>
<td>[[permenter2014partial]]</td>
<td>68</td>
</tr>
<tr>
<td>Mittelmann</td>
<td>Mittelmann website</td>
<td>31</td>
</tr>
<tr>
<td>Dressler-Illman-de Wolff (DIW)</td>
<td>[[dressler2019approach]]</td>
<td>155</td>
</tr>
<tr>
<td>Henrion-Toh</td>
<td>Didier Henrion and Kim-Chuan Toh</td>
<td>98</td>
</tr>
<tr>
<td>Toh-Sun-Yang</td>
<td>[[sun2015convergent, yang2015sdpnal]]</td>
<td>419</td>
</tr>
<tr>
<td>total</td>
<td></td>
<td>771</td>
</tr>
</tbody>
</table>
 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 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.
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

▶ 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/
Comparison criteria

- Does preprocessing reduce a problem?

\(^5\text{http://archive.dimacs.rutgers.edu/Challenges/Seventh/Instances/}\)
Comparison criteria

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

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

\(^5\text{http://archive.dimacs.rutgers.edu/Challenges/Seventh/Instances/}\)
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

<table>
<thead>
<tr>
<th>problem</th>
<th>correct</th>
<th>w/o prep.</th>
<th>after pd1/pd2</th>
<th>after dd1/dd2</th>
<th>after Sieve-SDP</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompactDim2R1</td>
<td>Infeas, +∞</td>
<td>3.79e+06, 4.20e+06</td>
<td>Infeas, 1</td>
<td>3.79e+06, 4.20e+06</td>
<td>Infeas, -</td>
</tr>
<tr>
<td>CompactDim2R2</td>
<td>Infeas, +∞</td>
<td>6.41e-10, 6.81e-10</td>
<td>Infeas, 2</td>
<td>6.41e-10, 6.81e-10</td>
<td>Infeas, -</td>
</tr>
<tr>
<td>CompactDim2R3</td>
<td>Infeas, +∞</td>
<td>1.5, 1.5</td>
<td>Infeas, 2</td>
<td>1.5, 1.5</td>
<td>Infeas, -</td>
</tr>
<tr>
<td>CompactDim2R4</td>
<td>Infeas, +∞</td>
<td>1.5, 1.5</td>
<td>Infeas, 2</td>
<td>1.5, 1.5</td>
<td>Infeas, -</td>
</tr>
<tr>
<td>CompactDim2R5</td>
<td>Infeas, +∞</td>
<td>1.5, 1.5</td>
<td>Infeas, 2</td>
<td>1.5, 1.5</td>
<td>Infeas, -</td>
</tr>
<tr>
<td>CompactDim2R6</td>
<td>Infeas, +∞</td>
<td>1.5, 1.5</td>
<td>Infeas, 2</td>
<td>1.5, 1.5</td>
<td>Infeas, -</td>
</tr>
<tr>
<td>CompactDim2R7</td>
<td>Infeas, +∞</td>
<td>1.5, 1.5</td>
<td>Infeas, 2</td>
<td>1.5, 1.5</td>
<td>Infeas, -</td>
</tr>
<tr>
<td>CompactDim2R8</td>
<td>Infeas, +∞</td>
<td>1.5, 1.5</td>
<td>Infeas, 2</td>
<td>1.5, 1.5</td>
<td>Infeas, -</td>
</tr>
<tr>
<td>CompactDim2R9</td>
<td>Infeas, +∞</td>
<td>1.5, 1.5</td>
<td>Infeas, 2</td>
<td>1.5, 1.5</td>
<td>Infeas, -</td>
</tr>
<tr>
<td>CompactDim2R10</td>
<td>Infeas, +∞</td>
<td>1.5, 1.5</td>
<td>Infeas, 2</td>
<td>1.5, 1.5</td>
<td>Infeas, -</td>
</tr>
</tbody>
</table>

**correctness %**

| 100%, 100% | 0%, 0% | 100%, 0% | 0%, 0% | 100%, - |
SDP relaxation for polynomial optimization

- Polynomial optimization:

\[
\begin{aligned}
\min_{x \in \mathbb{N}} & \quad f_0(x) \\
\text{s.t.} & \quad f_i(x) \geq 0, \quad i = 1, \ldots, r.
\end{aligned}
\]  

(poly-opt)
SDP relaxation for polynomial optimization

- Polynomial optimization:
  \[
  \min_{x \in \mathbb{N}} f_0(x) \\
  \text{s.t. } f_i(x) \geq 0, \quad i = 1, ..., r. \\
  \]  
  \[(\text{poly-opt})\]

- It is equivalent to
  \[
  \max_{\gamma \in \mathbb{R}} \gamma \\
  \text{s.t. } f_0(x) + \sum_{i=1}^{r} s_i(x) f_i(x) - \gamma = s_0(x), \quad \forall x \in \mathbb{N}, \\
  \]  
  where \(s_i(x), i = 0, 1, ..., r\) are sum-of-square polynomials \cite{lasserre2001global}.
SDP relaxation for polynomial optimization

- Polynomial optimization:
  \[
  \min_{x \in \mathbb{N}} \quad f_0(x) \\
  \text{s.t.} \quad f_i(x) \geq 0, \quad i = 1, \ldots, r. \\
  \text{(poly-opt)}
  \]

- It is equivalent to
  \[
  \max_{\gamma} \quad \gamma \\
  \text{s.t.} \quad f_0(x) + \sum_{i=1}^{r} s_i(x)f_i(x) - \gamma = s_0(x), \quad \forall x \in \mathbb{N},
  \]
  where \(s_i(x), \ i = 0, 1, \ldots, r\) are sum-of-square polynomials [[lasserre2001global]].

- It has SDP relaxation:
  \[
  \min_{\gamma} \quad -\gamma, \\
  \text{s.t.} \quad Q \succeq 0, \\
  \text{(SDP-relaxation)}
  \]
  where \(Q \in \mathbb{S}^n\) is based on \(\gamma\) and coefficients of \(f_i, \ i = 0, 1, \ldots, r\).
SDP relaxation for polynomial optimization

- Polynomial optimization:
  \[
  \min_{x \in \mathbb{N}} f_0(x) \quad \text{s.t.} \quad f_i(x) \geq 0, \quad i = 1, \ldots, r.
  \]  
  (poly-opt)

- It is equivalent to
  \[
  \max_{\gamma \in \mathbb{R}} \gamma \quad \text{s.t.} \quad f_0(x) + \sum_{i=1}^{r} s_i(x)f_i(x) - \gamma = s_0(x), \quad \forall x \in \mathbb{N},
  \]
  where \(s_i(x), \ i = 0, 1, \ldots, r\) are sum-of-square polynomials \([\text{lasserre2001global}]\).

- It has SDP relaxation:
  \[
  \min_{\gamma \in \mathbb{R}} -\gamma \quad \text{s.t.} \quad Q \succeq 0,
  \]  
  (SDP-relaxation)

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

- Infeasibility of (SDP-relaxation) gives a useless lower bound \(\gamma = -\infty\) to (poly-opt).
SDP relaxation for polynomial optimization

- Polynomial optimization:

\[
\begin{align*}
\min_{x \in \mathbb{N}} & \quad f_0(x) \\
\text{s.t.} & \quad f_i(x) \geq 0, \quad i = 1, \ldots, r.
\end{align*}
\]

(poly-opt)

- It is equivalent to

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

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

- It has SDP relaxation:

\[
\begin{align*}
\min_{\gamma} & \quad -\gamma, \\
\text{s.t.} & \quad Q \succeq 0,
\end{align*}
\]

(SDP-relaxation)

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

- 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.

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

- 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.

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

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.

<table>
<thead>
<tr>
<th>method</th>
<th># reduced</th>
<th># infeas detected</th>
<th># DIMACS error improved</th>
<th># out of memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>pd1</td>
<td>209</td>
<td>67</td>
<td>74</td>
<td>0</td>
</tr>
<tr>
<td>pd2</td>
<td>230</td>
<td>67</td>
<td>78</td>
<td>6</td>
</tr>
<tr>
<td>dd1</td>
<td>14</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>dd2</td>
<td>21</td>
<td>0</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Sieve-SDP</td>
<td>216</td>
<td>73</td>
<td>74</td>
<td>0</td>
</tr>
</tbody>
</table>
Overall summary on all 771 problems: time

**Table: Preprocessing and solving times.**

<table>
<thead>
<tr>
<th>method</th>
<th>(t_{\text{prep}}) (hr)</th>
<th>(t_{\text{sol}}) (hr)</th>
<th>(t_{\text{prep}} / t_{\text{sol_w/o}})</th>
<th>time reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>w/o</td>
<td>-</td>
<td>75.67</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>pd1</td>
<td>0.69</td>
<td>36.77</td>
<td>0.91%</td>
<td>50.50%</td>
</tr>
<tr>
<td>pd2</td>
<td>6.48</td>
<td>36.57</td>
<td>8.56%</td>
<td>43.12%</td>
</tr>
<tr>
<td>dd1</td>
<td>0.16</td>
<td>75.62</td>
<td>0.22%</td>
<td>-0.15%</td>
</tr>
<tr>
<td>dd2</td>
<td>10.00</td>
<td>75.56</td>
<td>13.21%</td>
<td>-13.16%</td>
</tr>
<tr>
<td>Sieve-SDP</td>
<td>0.60</td>
<td>36.62</td>
<td>0.80%</td>
<td>51.81%</td>
</tr>
</tbody>
</table>

\[
\text{time reduction: } \frac{t_{\text{sol\_w/o}} - (t_{\text{prep}} + t_{\text{sol}})}{t_{\text{sol\_w/o}}} \times 100\%.
\]
High speed of Sieve-SDP

- <0.1sec: 70%
- 0.1sec–1sec: 20%
- 1sec–1min: 8%
- 1min–6min: 2%
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

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

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

<table>
<thead>
<tr>
<th></th>
<th>reduction rate on $n$</th>
<th>reduction rate on $m$</th>
<th>added # free vars</th>
</tr>
</thead>
<tbody>
<tr>
<td>pd1</td>
<td>1.57%</td>
<td>6.90%</td>
<td>0</td>
</tr>
<tr>
<td>pd2</td>
<td>1.75%</td>
<td>7.94%</td>
<td>0</td>
</tr>
<tr>
<td>dd1</td>
<td>11.02%</td>
<td>0.00%</td>
<td>2293495</td>
</tr>
<tr>
<td>dd2</td>
<td>11.08%</td>
<td>0.00%</td>
<td>2315849</td>
</tr>
<tr>
<td>Sieve</td>
<td>3.49%</td>
<td>13.63%</td>
<td>0</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>methods</th>
<th>reduced</th>
<th>1</th>
<th>-1</th>
<th>2</th>
<th>-2</th>
<th>3</th>
<th>-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>pd1</td>
<td>54</td>
<td>12</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>13</td>
<td>0</td>
</tr>
<tr>
<td>pd2</td>
<td>75</td>
<td>12</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>17</td>
<td>6</td>
</tr>
<tr>
<td>dd1</td>
<td>14</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>dd2</td>
<td>21</td>
<td>0</td>
<td>2</td>
<td>6</td>
<td>1</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>Sieve</td>
<td>61</td>
<td>14</td>
<td>0</td>
<td>8</td>
<td>1</td>
<td>20</td>
<td>0</td>
</tr>
</tbody>
</table>
## Overall summary: time

<table>
<thead>
<tr>
<th>Method</th>
<th>Before</th>
<th>Processing</th>
<th>Solving</th>
</tr>
</thead>
<tbody>
<tr>
<td>dd2</td>
<td>30.71</td>
<td>31.1</td>
<td>31.1</td>
</tr>
<tr>
<td>dd1</td>
<td>31.8</td>
<td>31.1</td>
<td>31.1</td>
</tr>
<tr>
<td>pd1</td>
<td>30.9</td>
<td>30.9</td>
<td>30.9</td>
</tr>
<tr>
<td>Sieve</td>
<td>30.86</td>
<td>30.86</td>
<td>30.86</td>
</tr>
<tr>
<td>pd2</td>
<td>31.09</td>
<td>31.09</td>
<td>31.09</td>
</tr>
</tbody>
</table>
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!