Verifiable Safe Data Driven Control

T. Dai, J. Miller, M. Sznaier

Robust Systems Lab
Electrical and Computer Engineering
Northeastern University

Team:

Tianyu Dai

Jared Miller

Biel Roig-Solvas
Big Picture: Control Oriented Learning on the Fly

- **Three Phases:**
  - **Stayin’ Alive**
    - Maintain viability by avoiding a known bad set
    - Use this phase to learn about the plant
  - **Verifiable Data Driven Control**
    - Find a “stabilizing” control action
  - **Learning for Performance**
    - Keep learning while DD keeps you safe
    - Use these models to optimize performance
Big Picture: Control Oriented Learning on the Fly

- **Three Phases:**

- **Stayin’ Alive**
  - Maintain viability by avoiding a known bad set
  - Use this phase to learn about the plant

- **Verifiable Data Driven Control**
  - Find a “stabilizing” control action

- **Learning for Performance**
  - Keep learning while DD keeps you safe
  - Use these models to optimize performance

*All of these reduce to tractable SDPs via densities/measures*
Three Phases:

- **Stayin’ Alive**
  - Maintain viability by avoiding a known bad set
  - Use this phase to learn about the plant

- **Verifiable Data Driven Control**
  - Find a “stabilizing” control action

- **Learning for Performance**
  - Keep learning while DD keeps you safe
  - Use these models to optimize performance

All of these reduce to tractable SDPs via densities/measures...
But standard SDP methods scale horribly
Efficient Algorithms for large SDPs
Semi-definite programming

\[ A_i \cdot X = b_i \]

\[ \min_X C \cdot X \]

\[ A_i \cdot X = b_i \]

\[ X \succeq 0 \]
Interior point methods for SDPs

Few iterations, but costly \( (\text{time} \sim O(N^4), \text{memory} \sim O(N^4)) \)

\[
\min_X C \cdot X - \frac{1}{t} \log |X|
\]

\[
A_i \cdot X = b_i
\]

\[
X \succeq 0
\]

\[
C \cdot X(t) \geq C \cdot X^* \geq C \cdot X(t) - \frac{N}{t}
\]

Practically limited to \( N \sim 1000 \)
DD and SDD relaxations of SDPs

\[ \min_X C \cdot X \]
\[ A_i \cdot X = b_i \]
\[ X \geq 0 \]
\[ X \in DD \]
\[ X \in SDD \]

\[ C \cdot X^*_{DD} \geq C \cdot X^*_{SDD} \geq C \cdot X^*_{SDP} \]

- Much cheaper than SDPs
  - LP/SOCP (structured Hessian)
  - But generically suboptimal

A. Ahmadi et al.
Hall & Ahmadi proposed to solve a sequence of DD/SDD problems to improve the cost.

Closer to the SDP optimum, but still suboptimal in the general case and no theoretical guarantees.
Taking DD/SDD side-steps

Move away from the PSD boundary when stuck

\[
\begin{align*}
\max_X \log \det (X) \\
A_i \cdot X &= b_i \\
C \cdot X &= C \cdot X_k \\
X &\succeq 0 \\
X &\in DD/SDD?
\end{align*}
\]
Taking DD/SDD side-steps

\[ \max_X \phi_{DD/SDD}(X) \]
\[ A_i \cdot X = b_i \]
\[ C \cdot X = C \cdot X_k \]
\[ X \in DD/SDD \]
\[ \phi_{SDD}(X) = \sum_{i,j>i}^N \log |M_{i,j}| \]

- **Facts:**
  - LPs/SOCPs
  - Sparse, block diagonal Hessian with closed form inverse.
  - Guaranteed return to the central path
Globally Convergent DD/SDD Algorithm

- Alternate a sequence of cost decreasing & centering steps.
- Centering steps return to the central path (a maxdet problem)
- Guaranteed to reach $\varepsilon$-optimality in $\kappa$ iterations, with:
  \[
  \kappa \leq \frac{\log(t^*) - \log(t_0)}{\log(\chi)}
  \]
  $\chi$ depends only on problem size

First result proving that you can solve a generic SDP using LP/SOCP
Results

SDPLIB 1.2, A Library of Semidefinite Programming Test Problems

BRIAN BORCHERS

SDPLIB is a collection of semidefinite programming (SDP) test problems. The problems are drawn from a variety of applications, including truss topology design, control systems engineering, and relaxations of combinatorial optimization problems. The current version of the library contains a total of 92 SDP problems encoded in a standard format. It is hoped that SDPLIB will stimulate the development of improved software for the solution of SDP problems.

KEY WORDS: Semidefinite Programming
Examples

- Convergence guarantees are of theoretical interest, but bounds are worst-case. Empirical performance is much better than these bounds.

- Experiments on two SDP problems of the SDPLib dataset: a Lovász Theta-number SDP problem and a MaxCut relaxation.

- Both achieve global optimality up to the 4th decimal place.

- Theta2 (SDPLib). $N = 100$, $M = 498$

- Mcp250-2 (SDPLib). $N = 250$, $M = 250$
Decomposed subsets for structured SDPs

\[
\begin{bmatrix}
  x_{11} & x_{12} & \? & \? & x_{15} & x_{16} \\
  x_{12} & x_{22} & x_{23} & \? & x_{25} & \? \\
  \? & x_{23} & x_{33} & x_{34} & x_{35} & \? \\
  \? & \? & x_{34} & x_{44} & x_{45} & \? \\
  x_{15} & x_{25} & x_{35} & x_{45} & x_{55} & x_{56} \\
  x_{16} & \? & \? & \? & x_{56} & x_{66}
\end{bmatrix}
\]
Exploiting Structure in SDP

- SDPs arising from dynamical systems are typically structured:
  - Sparsity
  - Symmetry

- This structure can be exploited to reduce complexity

- Large PSD Cone → smaller cones
- Often renders problems tractable
Given the matrix:

\[
\begin{bmatrix}
  x_{11} & x_{12} & ? & ? & x_{15} & x_{16} \\
  x_{12} & x_{22} & x_{23} & ? & x_{25} & ? \\
  ? & x_{23} & x_{33} & x_{34} & x_{35} & ? \\
  ? & ? & x_{34} & x_{44} & x_{45} & ? \\
  x_{15} & x_{25} & x_{35} & x_{45} & x_{55} & x_{56} \\
  x_{16} & ? & ? & ? & x_{56} & x_{66}
\end{bmatrix}
\]

does there exist a PSD completion?
Example of a chordal sparse structure in SDP

- Given the matrix:

\[
\begin{bmatrix}
  x_{11} & x_{12} & ? & ? & x_{15} & x_{16} \\
  x_{12} & x_{22} & x_{23} & ? & x_{25} & ? \\
  ? & x_{23} & x_{33} & x_{34} & x_{35} & ? \\
  ? & ? & x_{34} & x_{44} & x_{45} & ? \\
  x_{15} & x_{25} & x_{35} & x_{45} & x_{55} & x_{56} \\
  x_{16} & ? & ? & ? & x_{56} & x_{66}
\end{bmatrix}
\]

- does there exist a PSD completion?
- The answer is related to the cliques of an associated graph
- One node per row/column
- Edge \((i,j)\) if \(x_{ij}\) is given
Example of a chordal sparse structure in SDP

Grone’s Theorem: $X$ is PSD completable iff $X(C_i) \succeq 0$, $i = 1, ..n_c$
Chordal decompositions of SDPs

- Decompose, then approximate
- Blocks are $DD$ vs. Matrix is $DD$

$DD \subset DD(\varepsilon, ?) \subset S_+$

$SDD \subset SDD(\varepsilon, ?) \subset S_+$

- Preserving structure yields tighter approximations
- Karush-Kuhn Tucker (KKT) certification of optimality
Decomposed Change of Basis

- Decomposing always improves objectives at iteration 1
- Anecdotally beats dense change-of-basis after
- May be combined with Decrease/Centering iterations
Mixing Cones

- Adds flexibility in optimization
- Useful if problem has few large cliques
Sea Star: Network $H_\infty$ norm estimation.

- $H_\infty$ norm: energy gain of bounded disturbance
- Each agent is an order $\leq 10$ linear system
- 310 agents in total, globally stable
- 70 agents in the head, 20 agents per arm
Sea Star: Sparse Clique Structure

- BRL: 1 giant clique of size 387 + many smaller ones
- Matlab runs out of memory even with chordal decompositions.
- Keep cliques below threshold PSD, approximate large cliques

Sea Star LMI Clique Sizes ($p = 1278$)
**Sea Star: Tight Upper Bounds**

- All marked entries achieved optimality (KKT)
- $B_k$: block factor-width 2 with block size $k$
- $B_1 = \text{SDD}$

<table>
<thead>
<tr>
<th>PSD threshold</th>
<th>$D_D$</th>
<th>$B_1$</th>
<th>$B_3$</th>
<th>$B_5$</th>
<th>$B_8$</th>
<th>$B_{15}$</th>
<th>$B_{30}$</th>
<th>$B_{55}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td>9.54</td>
<td>2.71</td>
<td>2.33</td>
<td>2.01</td>
<td>3.95</td>
<td>9.50</td>
<td>34.75</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>11.88</td>
<td>5.21</td>
<td>4.64</td>
<td>4.49</td>
<td>5.02</td>
<td>9.60</td>
<td>33.99</td>
<td></td>
</tr>
</tbody>
</table>

Upper bound time (min.) $\gamma = 1.137$
Symmetry and Sparsity

- Matrix group with sparsity and symmetry (90 x 90)

Permutation Orbits

Group-Invariant Matrix $X$

Block-Diagonalizer $P$

Block Diagonalization $X_k$
Symmetry Structure

- Sparsity alone destroys Symmetry
- Symmetry, then Sparsity
- Exploiting both structures yields the tightest bounds

\[ \text{\textit{SDD} Block Arrow with Symmetry} \]

<table>
<thead>
<tr>
<th></th>
<th>Cost</th>
<th>Full</th>
<th>Sym.</th>
<th>Time (s)</th>
<th>Full</th>
<th>Sym.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dense</td>
<td>12.96</td>
<td>10.86</td>
<td></td>
<td>Dense</td>
<td>124.5</td>
<td>19.3</td>
</tr>
<tr>
<td>Sparse</td>
<td>9.49</td>
<td>8.44</td>
<td></td>
<td>Sparse</td>
<td>38.2</td>
<td>12.0</td>
</tr>
</tbody>
</table>
Work in progress:

- Randomized algorithms to solve large SDPs
  - Combination with Frank-Wolfe type algorithms

- Exploiting structure + sparsity:
  - Cone and factor width k decompositions /inclusions
  - Non symmetric barrier algorithms that exploit these