ProxSDP.jl: New developments on Semidefinite Programming in Julia/JuMP

Mario Souto and Joaquim Dias Garcia

LAMPS LABORATORY OF APPLIED MATHEMATICAL PROGRAMMING AND STATISTICS | PUC RIO

March 19, 2019

Unique games conjecture

- Unique Games Conjecture: For a large class of problems, even finding an approximate solution is NP-hard.
- If the UGC is true, for a large class of problems, no polynomial-time algorithm can be better than **????**

Unique games conjecture

COMPUTATIONAL COMPLEXITY

First Big Steps Toward Proving the Unique Games Conjecture

1

The latest in a new series of proofs brings theoretical computer scientists within striking distance of one of the great conjectures of their discipline.

Unique games conjecture

COMPUTATIONAL COMPLEXITY

First Big Steps Toward Proving the Unique Games Conjecture

1

The latest in a new series of proofs brings theoretical computer scientists within striking distance of one of the great conjectures of their discipline.

4 Do you think the unique games conjecture is true or false? (en.m.wikipedia.org)
submitted 1 year ago by 744196884
3 comments share save hide give award report crosspost

all 3 comments
sorted by: best

[–] Rioghasarig 6 points 1 year ago
Yes
permalink embed save give award

[–] GNULinuxProgrammer 1 point 1 year ago
It seems to me that the same philosophical reasons as to why P is probably != NP can be applied to this conjecture?
permalink embed save give award

> [–] 744196884 [S] 2 points 1 year ago
> But if that were true then the academic community wouldn't be evenly divided on whether the unique games conjecture were true or false. Compare it to p vs np where most people think p =/= np

Applications

- Control problems;
- Robust structural design (e.g. truss topology);
- Eigenvalue optimization problems;
- Relaxations for combinatorial problems (e.g. Max-Cut, graph coloring, traveling salesman, Max-Sat, . . .);
- Optimal power flow relaxation;
- Machine Learning (matrix completion, robust PCA, kernel learning).

SDP latest news

ALGORITHMS

A Classical Math Problem Gets Pulled Into the Modern World

13

A century ago, the great mathematician David Hilbert posed a probing question in pure mathematics. A recent advance in optimization theory is bringing Hilbert's work into a world of self-driving cars.

MATHEMATICS

A New Tool to Help Mathematicians Pack

Improvements in how densely spheres and other shapes can be packed together could lead to advances in materials science, deep space communication and theoretical physics.

Quantamagazine

Why isn't SDP widely used?

- Problem size grows quadratically;

Why isn't SDP widely used?

- Problem size grows quadratically;
- Sparsity is not trivial to be exploited:
  - Changing with the adoption of chordal decomposition;

Why isn't SDP widely used?

- Problem size grows quadratically;
- Sparsity is not trivial to be exploited:
  - Changing with the adoption of chordal decomposition;
- Formulating the problem as a SDP may not always be straightforward:
  - Solved by modern modeling frameworks (**JuMP.jl** and others);

Why isn't SDP widely used?

- Problem size grows quadratically;
- Sparsity is not trivial to be exploited:
  - Changing with the adoption of chordal decomposition;
- Formulating the problem as a SDP may not always be straightforward:
  - Solved by modern modeling frameworks (**JuMP.jl** and others);
- State-of-the-art solvers are yet unable to solve large SDP problems.

Motivation - Low-rank structure

- Any SDP with m constraints admits a solution with rank at most $\sqrt{2m}$ (Barvinok-Pataki 1995/98);

Motivation - Low-rank structure

- Any SDP with m constraints admits a solution with rank at most $\sqrt{2m}$ (Barvinok-Pataki 1995/98);

- In practice, several SDP problems admits even lower rank solutions;

Motivation - Low-rank structure

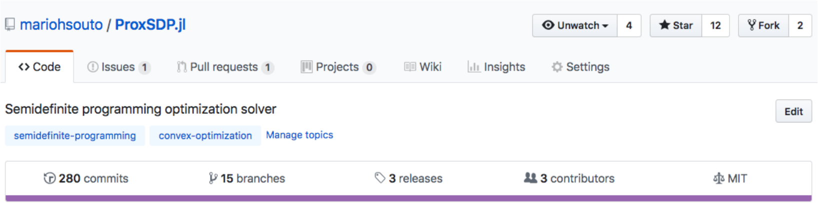
- Any SDP with m constraints admits a solution with rank at most $\sqrt{2m}$ (Barvinok-Pataki 1995/98);
- In practice, several SDP problems admits even lower rank solutions;
- Interior points methods frequently compute the full rank solution;

Motivation - Low-rank structure

- Any SDP with m constraints admits a solution with rank at most $\sqrt{2m}$ (Barvinok-Pataki 1995/98);
- In practice, several SDP problems admits even lower rank solutions;
- Interior points methods frequently compute the full rank solution;
- Low-rank structure is usually exploited as a matrix factorization (Burer-Monteiro 2003):

  $X = V^\intercal V$ where $V \in \mathbb{R}^{k \times n}$ and k is the target rank.

Recap from JuMPdev 2018...



https://github.com/mariohsouto/ProxSDP.jl

Semidefinite Programming

- Primal:

$$\begin{aligned} \underset{X \in \mathbb{S}^n}{\text{minimize}} \quad & \mathbf{tr}(CX) \\ \text{subject to} \quad & \mathcal{M}(X) = b, \\ & X \succeq 0. \end{aligned}$$

  where

$$\mathcal{M}(X) = \begin{bmatrix} \mathbf{tr}(M_1 X) \\ \mathbf{tr}(M_2 X) \\ \vdots \\ \mathbf{tr}(M_m X) \end{bmatrix}.$$

- Problem data: $M_1, \ldots, M_m, C \in \mathbb{S}^n$, $b \in \mathbb{R}^m$ and $h \in \mathbb{R}^p$.

Optimality condition

$$0 \in \partial\, \textbf{tr}(CX) + \partial\, I_{\mathbb{S}^n_+}(X) + \mathcal{M}^T(\partial\, I_{\substack{=b \\ \leq h}}(\mathcal{M}(X))).$$

- Introducing an auxiliary variable $y \in \mathbb{R}^{p+m}$:

$$\begin{aligned} &0 \in \partial\, \textbf{tr}(CX) + \partial\, I_{\mathbb{S}^n_+}(X) + \mathcal{M}^T(y), \\ &y \in \partial\, I_{\substack{=b \\ \leq h}}(\mathcal{M}(X)). \end{aligned}$$

- By definition, y is the dual variable associated with the linear constraints;
- If strong duality holds, any (X^*, y^*) satisfying the inclusion above is the optimal primal-dual pair.

Algorithm PD-SDP

while $\epsilon^k_{\text{comb}} > \epsilon_{\text{tol}}$ **do**

$X^{k+1} \leftarrow \textbf{proj}_{\mathbb{S}^n_+}(X^k - \tau(\mathcal{M}^T(y^k) + C))$ ▷ Primal step

$y^{k+1/2} \leftarrow y^k + \sigma\mathcal{M}((1+\theta)X^{k+1} - \theta X^k)$ ▷ Dual step part 1

$y^{k+1} \leftarrow y^{k+1/2} - \sigma\,\textbf{proj}_{=b}(y^{k+1/2}/\sigma)$ ▷ Dual step part 2

end while

return (X^{k+1}, y^{k+1})

Computational bottleneck

- The computational complexity of each iteration of `PD-SDP` is $\mathcal{O}(n^3)$;

Computational bottleneck

- The computational complexity of each iteration of `PD-SDP` is $\mathcal{O}(n^3)$;
- The spectral decomposition can be prohibitive even for medium scale problems;

Computational bottleneck

- The computational complexity of each iteration of `PD-SDP` is $\mathcal{O}(n^3)$;
- The spectral decomposition can be prohibitive even for medium scale problems;
- Can be reduced to $\mathcal{O}(n^2 r)$, if one knows the target rank r *a priori* to each iteration.

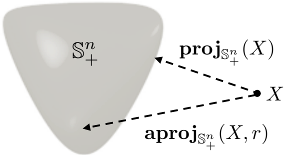
Computational bottleneck

- The computational complexity of each iteration of `PD-SDP` is $\mathcal{O}(n^3)$;
- The spectral decomposition can be prohibitive even for medium scale problems;
- Can be reduced to $\mathcal{O}(n^2 r)$, ~~if one knows the target rank r *a priori* to each iteration.~~

Low-rank approximation

- Truncated projection onto the positive semidefinite cone:

$$\textbf{aproj}_{\mathbb{S}^n_+}(X, r) = \sum_{i=1}^{r} \max\{0, \lambda_i\} u_i u_i^T,$$



- From (Eckart–Young–Mirsky theorem 1936), the approximation error can be bounded as

$$\left\| \textbf{proj}_{\mathbb{S}^n_+}(X) - \textbf{aproj}_{\mathbb{S}^n_+}(X, r) \right\|_F^2 \leq (n - r) \max\{\lambda_r, 0\}.$$

Algorithm LR-PD-SDP

while $(n - \boldsymbol{r})\lambda_r > \epsilon_\lambda$ **do**

  while $\epsilon^k_{\text{comb}} > \epsilon_{\text{tol}}$ **and** $\epsilon^k_{\text{comb}} < \epsilon^{k-\ell}_{\text{comb}}$ **do**

    $X^{k+1} \leftarrow \textbf{aproj}_{\mathbb{S}^n_+}(X^k - \tau(\mathcal{M}^T(y^k) + C), \boldsymbol{r})$ ▷ Approx. primal step

    $y^{k+1/2} \leftarrow y^k + \sigma\mathcal{M}((1+\theta)X^{k+1} - \theta X^k)$ ▷ Dual step part 1

    $y^{k+1} \leftarrow y^{k+1/2} - \sigma\, \textbf{proj}_{=b}(y^{k+1/2}/\sigma)$ ▷ Dual step part 2

  end while

  $\boldsymbol{r} \leftarrow 2\,\boldsymbol{r}$ ▷ *Target-rank* update

end while

return (X^{k+1}, y^{k+1})

Street-fighting optimization

- Algorithmic
  - Use adaptive step size for primal and dual update. Use **heuristic** for balance residuals;
  - **Linesearch** for selecting over-relaxation parameter as large as possible.
- Computational
  - Arpack `eig` function might fail. Limit the number of iterations, choose tolerance accordingly;
  - Can use MKL if available.

Adding other cones and inequalities

Algorithm `LR-PD-SDP`

while $(n - \boldsymbol{r})\lambda_r > \epsilon_\lambda$ **do**

while $\epsilon^k_{\text{comb}} > \epsilon_{\text{tol}}$ **and** $\epsilon^k_{\text{comb}} < \epsilon^{k-\ell}_{\text{comb}}$ **do**

$X^{k+1} \leftarrow \textbf{aproj}_{\mathcal{K}}(X^k - \tau(\mathcal{M}^T(y^k) + C), \boldsymbol{r})$ ▷ Approx. primal step

$y^{k+1/2} \leftarrow y^k + \sigma\mathcal{M}((1+\theta)X^{k+1} - \theta X^k)$ ▷ Dual step part 1

$y^{k+1} \leftarrow y^{k+1/2} - \sigma\, \textbf{proj}_{\substack{=b \\ \leq h}}(y^{k+1/2}/\sigma)$ ▷ Dual step part 2

end while

$\boldsymbol{r} \leftarrow 2\,\boldsymbol{r}$ ▷ *Target-rank* update

end while

return (X^{k+1}, y^{k+1})

Graph equipartition problem

n	sdplib	SCS	CSDP	MOSEK	PD-SDP	LR-PD-SDP
124	gpp124-1	1.6	0.4	**0.2**	0.7	0.9
124	gpp124-2	1.5	0.4	0.3	0.5	**0.2**
124	gpp124-3	1.6	0.3	**0.2**	0.6	**0.2**
124	gpp124-4	1.7	0.5	0.3	0.6	**0.2**
250	gpp250-1	21.4	2.9	**0.9**	3.7	1.4
250	gpp250-2	7.8	2.2	**1.1**	4.1	1.2
250	gpp250-3	12.6	2.1	**0.9**	3.4	**0.9**
250	gpp250-4	16.4	2.2	0.9	3.8	**0.6**
500	gpp500-1	134.2	59.1	8.2	22.7	**5.6**
500	gpp500-2	97.4	12.2	8.6	21.5	**6.1**
500	gpp500-3	64.4	12.1	8.9	15.5	**4.4**
500	gpp500-4	71.4	13.4	8.7	15.4	**6.5**
801	equalG11	324.2	47.3	32.4	84.3	**11.3**
1001	equalG51	425.1	98.7	83.4	113.5	**22.5**

Table: Comparison of running times (seconds) for the SDPLIB's graph equipartition problem instances.

Sensor network localization

n	SCS	CSDP	MOSEK	PD-SDP	LR-PD-SDP
50	0.2	0.2	**0.1**	0.5	0.6
100	**0.8**	4.5	0.9	6.1	1.6
150	**2.6**	28.1	3.2	14.4	3.6
200	6.4	89.8	11.2	32.3	**6.1**
250	12.1	239.2	36.4	52.9	**7.9**
300	28.7	timeout	85.2	96.6	**13.5**

Table: Comparison of running times (seconds) for randomized network localization problem instances.

MIMO experiments

n	SCS	CSDP*	MOSEK	PD-SDP	LR-PD-SDP
100	1.5	1.2	**0.1**	**0.1**	**0.1**
500	277.8	27.4	2.3	3.1	**1.1**
1000	timeout	97.2	15.6	16.5	**4.7**
2000	timeout	473.6	117.5	115.9	**38.9**
3000	timeout	timeout	418.2	350.6	**122.1**
4000	timeout	timeout	976.8	906.5	**258.3**
5000	timeout	timeout	timeout	timeout	**472.4**

Table: Running times (seconds) for MIMO detection with high SNR.

Conclusion

- Achievements:
  - Primal-dual method for solving SDP;

Conclusion

- Achievements:
  - Primal-dual method for solving SDP;
  - Low-rank structure is efficiently exploited;

Conclusion

- Achievements:
  - Primal-dual method for solving SDP;
  - Low-rank structure is efficiently exploited;
  - Open-source SDP solver [`ProxSDP`] is readly available, https://github.com/mariohsouto/ProxSDP.jl

Conclusion

- Achievements:
  - Primal-dual method for solving SDP;
  - Low-rank structure is efficiently exploited;
  - Open-source SDP solver [`ProxSDP`] is readly available, https://github.com/mariohsouto/ProxSDP.jl
- Future ideas:
  - Explore properties of intermediate low-rank feasible solution;

Conclusion

- Achievements:
  - Primal-dual method for solving SDP;
  - Low-rank structure is efficiently exploited;
  - Open-source SDP solver [`ProxSDP`] is readly available,
    https://github.com/mariohsouto/ProxSDP.jl
- Future ideas:
  - Explore properties of intermediate low-rank feasible solution;
  - Combine proposed method with chordal sparsity techniques;

Conclusion

- Achievements:
  - Primal-dual method for solving SDP;
  - Low-rank structure is efficiently exploited;
  - Open-source SDP solver [`ProxSDP`] is readly available, https://github.com/mariohsouto/ProxSDP.jl
- Future ideas:
  - Explore properties of intermediate low-rank feasible solution;
  - Combine proposed method with chordal sparsity techniques;
  - Exploit low rank structure of other problems (SOS, AC relaxation...)