Exploiting Low-Rank Structure in Semidefinite Programming by Approximate Operator Splitting

Mario Souto, **Joaquim D. Garcia** and Álvaro Veiga

LAMPS LABORATORY OF APPLIED MATHEMATICAL PROGRAMMING AND STATISTICS | PUC RIO

June 26, 2018

Outline

Introduction

Algorithm

Exploiting low-rank structure

`SDP solver`

Case studies

Conclusion

Semidefinite Programming

- Primal:

$$\begin{array}{ll} \underset{X \in \mathbb{S}^n}{\text{minimize}} & \mathbf{tr}(CX) \\ \text{subject to} & \mathbf{tr}(A_i X) = b_i, \quad i = 1, \ldots, m, \\ & X \succeq 0, \end{array}$$

- Dual:

$$\begin{array}{ll} \underset{y \in \mathbb{R}^m}{\text{maximize}} & b^T y \\ \text{subject to} & \sum_{i=1}^{m} y_i A_i \preceq C. \end{array}$$

- Problem data:
  - $A_1, \ldots, A_p, C \in \mathbb{S}^n$;
  - $b_1, \ldots, b_m \in \mathbb{R}$.

Semidefinite Programming - *General* form

- Primal:

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

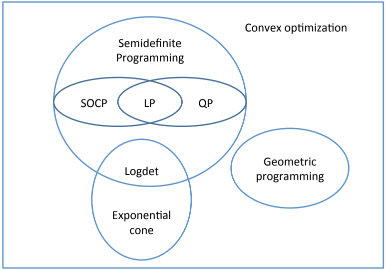
- Dual:

$$\begin{aligned} \underset{y' \in \mathbb{R}^m, y'' \in \mathbb{R}^p}{\text{maximize}} \quad & b^T y' + h^T y'' \\ \text{subject to} \quad & \mathcal{A}^*(y') + \mathcal{G}^*(y'') \preceq C, \\ & y'' \leq 0. \end{aligned}$$

- Where:
  - $\mathcal{A}(X) = [\mathbf{tr}(A_1 X), ..., \mathbf{tr}(A_m X)]^T$;
  - $b = [b_1, \ldots, b_m]^T$.

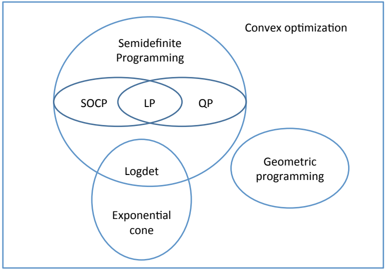
Why SDP matters?

- Subsumes most of convex optimization problems;



Why SDP matters?

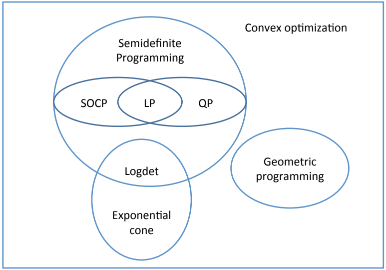
- Subsumes most of convex optimization problems;



- Several problems of interest lie in the SDP family;

Why SDP matters?

- Subsumes most of convex optimization problems;



- Several problems of interest lie in the SDP family;
- Establishes tight convex relaxations for several nonconvex problems.

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).

Why isn't SDP widely used?

- Problem size grows very fast (quadratically on matrix side);

Why isn't SDP widely used?

- Problem size grows very fast (quadratically on matrix side);
- Sparsity is not trivial to be exploited:
  - Changing with the adoption of chordal decomposition;

Why isn't SDP widely used?

- Problem size grows very fast (quadratically on matrix side);
- 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 modelling frameworks (JuMP.jl, cvxpy, Convex.jl);

Why isn't SDP widely used?

- Problem size grows very fast (quadratically on matrix side);
- 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 modelling frameworks (JuMP.jl, cvxpy, Convex.jl);
- 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.

Goals

- Propose a novel first-order method for solving SDPs:
  - Based on the *primal-dual hybrid gradient*,
  - Providing both optimal primal and dual variables,
  - Admits inequalities without the use of slack variables.

Goals

- Propose a novel first-order method for solving SDPs:
  - Based on the *primal-dual hybrid gradient*,
  - Providing both optimal primal and dual variables,
  - Admits inequalities without the use of slack variables.
- Exploit the **low-rank** structure within the convex optimization framework;

Goals

- Propose a novel first-order method for solving SDPs:
  - Based on the *primal-dual hybrid gradient*,
  - Providing both optimal primal and dual variables,
  - Admits inequalities without the use of slack variables.
- Exploit the **low-rank** structure within the convex optimization framework;
- Make available an open source SDP solver, called `ProxSDP`.

Outline

Algorithm

- The SDP problem can be solved by a Proximal Point Algorithm (Rockafellar 1976):
- More specifically, a particular case of the Primal-Dual Hybrid Gradient (Chambolle-Pock 2010):
- The resulting algorithm `PD-SDP` has a convergence rate of $\mathcal{O}(1/N)$, optimal for nonsmooth problems (Nesterov 2004).

Algorithm PD-SDP

Given: $\mathcal{M}$, $b \in \mathbb{R}^p$, $h \in \mathbb{R}^q$ and $C \in \mathbb{S}^n$.

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

$$X^{k+1} \leftarrow \textbf{proj}_{\mathbb{S}_+^n}(X^k - \alpha(\mathcal{M}^*(y^k) + C))$$

$$y^{k+1/2} \leftarrow y^k + \alpha\mathcal{M}(2X^{k+1} - X^k)$$

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

end while

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

- Spectral decomposition + Matrix multiplication + Truncations

Outline

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 $\mathcal{O}(n^3)$ due to spectral decomposition;
- 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:

$$\mathbf{proj}_{\mathbb{S}_+^n}(X, r) = \sum_{i=1}^{r} \mathsf{max}\{0, \lambda_i\} u_i u_i^T.$$

- The approximation error can be expressed as the sum of the eigenvalues that were left out by the truncated projection (Eckart–Young–Mirsky theorem 1936)

$$\left\| \mathbf{proj}_{\mathbb{S}_+^n}(X) - \mathbf{proj}_{\mathbb{S}_+^n}(X, r) \right\|_F^2 = \sum_{i=r+1}^{n} \max\{\lambda_i, 0\} \le (n-r) \max\{\lambda_r, 0\}.$$

- The approximate fixed-point iteration do converge as long as the error component is summable (Eckstein-Bertsekas 1992)

Algorithm `LR-PD-SDP`

Given: $\mathcal{M}$, $b \in \mathbb{R}^p$, $h \in \mathbb{R}^q$, $C \in \mathbb{S}^n$ and $r = 1$.

while $(n - r)\lambda_r > \epsilon_{\text{tol}}$ **do**

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

$X^{k+1} \leftarrow \textbf{proj}_{\mathbb{S}^n_+}(X^k - \alpha(\mathcal{M}^*(y^k) + C), r)$

$y^{k+1/2} \leftarrow y^k + \alpha\mathcal{M}(2X^{k+1} - X^k)$

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

end while

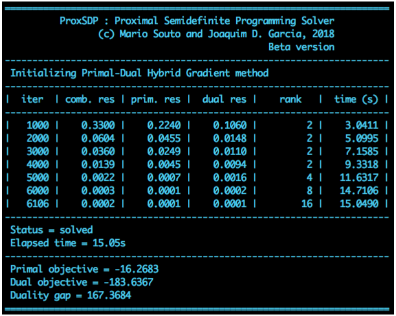
$r \leftarrow 2r$

end while

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

Outline

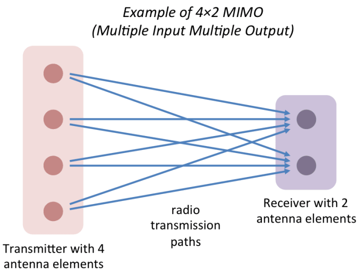
- Open source solver developed in the Julia language;
- Solves general SDP problems (currently can be called from JuMP.jl);
- Fast performance for problems with low-rank structure;
- Provides both optimal primal and dual solutions.



```
====================================================================
              ProxSDP : Proximal Semidefinite Programming Solver
                     (c) Mario Souto and Joaquim D. Garcia, 2018
                                                    Beta version
--------------------------------------------------------------------
 Initializing Primal-Dual Hybrid Gradient method
--------------------------------------------------------------------
|  iter  | comb. res | prim. res |  dual res |    rank   |  time (s) |
--------------------------------------------------------------------
|   1000 |    0.3300 |    0.2240 |    0.1060 |         2 |    3.0411 |
|   2000 |    0.0604 |    0.0455 |    0.0148 |         2 |    5.0995 |
|   3000 |    0.0360 |    0.0249 |    0.0110 |         2 |    7.1585 |
|   4000 |    0.0139 |    0.0045 |    0.0094 |         2 |    9.3318 |
|   5000 |    0.0022 |    0.0007 |    0.0016 |         4 |   11.6317 |
|   6000 |    0.0003 |    0.0001 |    0.0002 |         8 |   14.7106 |
|   6106 |    0.0002 |    0.0001 |    0.0001 |        16 |   15.0490 |
--------------------------------------------------------------------
 Status = solved
 Elapsed time = 15.05s
--------------------------------------------------------------------
 Primal objective = -16.2683
 Dual objective = -183.6367
 Duality gap = 167.3684
====================================================================
```

Outline

Massive MIMO



- Binary Multiple Input Multiple Output (MIMO):

$$y = Hx + \varepsilon,$$

  - Transmitted symbols $x \in \{-1, +1\}^n$;
  - Received signal $y \in \mathbb{R}^m$;
  - Matrix of channel coefficients $H \in \mathbb{R}^{m \times n}$;
  - Additive Gaussian noise $\varepsilon \in \mathbb{R}^m$ with variance σ^2.

MIMO detection SDR

- Write binary constraints using the SD relaxation
- Resulting formulation:

$$
\begin{array}{ll}
\underset{X \in \mathbb{S}_+^{n+1}}{\text{minimize}} & \mathbf{tr}(WX) \\
\text{subject to} & \mathbf{diag}(X) = 1, \\
& X \succeq 0, \\
& X_{n+1,n+1} = 1, \\
& -1 \leq X \leq 1, \\
& \mathbf{rank}(X) = 1.
\end{array}
$$

- De-noising is exact if signal to noise ratio is sufficiently large (Ottersten-Jalden 2006).

MIMO experiments

Table: MIMO detection with high SNR.

n	**m**	**p**	**CSDP**	**SCS**	**LR-PD-SDP**
100	101	10201	114.3	1.6	**0.3**
200	201	40401	timeout	8.9	**1.3**
300	301	90601	timeout	39.6	**3.1**
400	401	160801	timeout	101	**6.1**
500	501	251001	timeout	136.1	**8.8**

Graph equipartition problem from SDPLIB

n	Rank	Instance	SCS	CSDP	PD-SDP	LR-PD-SDP
124	4	gpp124-1	29.8	**0.6**	8.4	**0.6**
124	4	gpp124-2	11.1	**0.5**	6.6	**0.5**
124	6	gpp124-3	9.3	0.6	5.3	**0.5**
124	6	gpp124-4	13.5	**0.6**	18.4	0.8
250	5	gpp250-1	155.6	**2.4**	32.2	2.6
250	7	gpp250-2	76.9	**2.4**	23.7	2.7
250	8	gpp250-3	61.8	**2.2**	29.3	3.5
250	8	gpp250-4	70.5	**2.3**	40.3	2.5
500	7	gpp500-1	timeout	25.9	150.2	**9.7**
500	8	gpp500-2	634.2	22.3	156.8	**9.5**
500	11	gpp500-3	405.37	16.4	117.2	**11.9**
500	13	gpp500-4	429.7	13.4	129.4	**14.4**
801	31	equalG11	timeout	81.1	timeout	**29.4**
1001	16	equalG51	timeout	164.6	timeout	**55.6**

Table: Comparison of convergence time, in seconds, for SDPLIB's graph equipartition problem instances.

Outline

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 low-rank recovered solution;

Conclusion

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