

Dealing with inequalities in large scale Semidefinite Programs

A computational study

Federico Battista, Marianna De Santis

DIAG, Università di Roma Sapienza

ODS2021



SAPIENZA
UNIVERSITÀ DI ROMA

Semidefinite Programs

We focus on solving large scale SDPs in the following form:

$$\begin{aligned} \min \quad & \langle C, X \rangle \\ \text{s.t.} \quad & \langle A^i, X \rangle \leq b_i, \quad \forall i = 1, \dots, l \\ & \langle A^j, X \rangle = b_j, \quad \forall j = l + 1, \dots, m \\ & X \in \mathcal{S}_n^+ \end{aligned} \tag{1}$$

where:

- $\langle M, N \rangle = \text{trace}(MN)$ is the standard inner product in \mathcal{S}_n
- $C \in \mathcal{S}_n$,
- $A^i \in \mathcal{S}_n$, $i = 1, \dots, m + l$,
- $b \in \mathbb{R}^{m+l}$.

Reduction of SDPs in standard form

One can always reduce (1) into standard form. Let $s \in \mathbb{R}^l$ be a vector of slack variables, with $s_i \geq 0$, $i = 1, \dots, l$. Then we can expand matrix X to $\bar{X} \in \mathcal{S}_{n+l}$

$$\bar{X} := \begin{pmatrix} X & 0_{n,l} \\ 0_{l,n} & \text{Diag}(s) \end{pmatrix}$$

Remark: $\bar{X} \succeq 0 \iff X \succeq 0, s \geq 0$.

By the same arguments, we expand matrices A_i, A_j, C to $\bar{A}^i, \bar{A}^j, \bar{C} \in \mathcal{S}_{n+l}^+$

$$\bar{A}^i := \begin{pmatrix} A^i & 0_{n,l} \\ 0_{l,n} & e_i^T e_i \end{pmatrix}, \quad \bar{A}^j := \begin{pmatrix} A^j & 0_{n,l} \\ 0_{l,n} & 0_{l,l} \end{pmatrix}, \quad \bar{C} := \begin{pmatrix} C & 0_{n,l} \\ 0_{l,n} & 0_{l,l} \end{pmatrix}$$

Reduction of SDPs in standard form

Then (1) can be written as

$$\begin{aligned} \min \quad & \langle \bar{C}, \bar{X} \rangle \\ \text{s.t.} \quad & \bar{A}\bar{X} = b \\ & \bar{X} \in \mathcal{S}_{n+l}^+ \end{aligned} \tag{2}$$

where

- $b := (b_1, \dots, b_m) \in \mathbb{R}^m$
- $\bar{A} : \mathcal{S}_{n+l} \rightarrow \mathbb{R}^m$ is the linear operator $(\bar{A}\bar{X})_i = \langle \bar{A}^i, \bar{X} \rangle$ with $\bar{A}^i \in \mathcal{S}_{n+l}$, $i = 1, \dots, m$.

Duality in Semidefinite Programs

The dual problem of (2) is defined as

$$\begin{aligned} \max \quad & b^T y \\ \text{s.t.} \quad & \bar{\mathcal{A}}^T y + \bar{Z} = \bar{C} \\ & \bar{Z} \in \mathcal{S}_{n+l}^+, \end{aligned} \tag{3}$$

where $\bar{\mathcal{A}}^T : \mathbb{R}^m \rightarrow \mathcal{S}_{n+l}$ is the adjoint operator of $\bar{\mathcal{A}}$, i.e

$$\bar{\mathcal{A}}^T y = \sum_i y_i \bar{A}^i \quad \text{for } y \in \mathbb{R}^m.$$

Note that the matrix $\bar{Z} \in \mathcal{S}_{n+l}$ is a “slack” matrix variable that can be written as

$$\bar{Z} := \begin{pmatrix} Z & 0_{n,l} \\ 0_{n,l} & \text{diag}(p) \end{pmatrix},$$

with $p \in \mathbb{R}^l$.

Duality in Semidefinite Programs

Some assumptions:

- both the primal (2) and the dual (3) problems have strictly feasible points (i.e. Slater's condition is satisfied)
- then *strong duality* holds and (y, \bar{Z}, \bar{X}) is optimal iff

$$\begin{aligned} \bar{A}\bar{X} &= b & \bar{A}^\top y + \bar{Z} &= \bar{C} & \bar{Z}\bar{X} &= 0 \\ \bar{X} &\in \mathcal{S}_{n+1}^+ & \bar{Z} &\in \mathcal{S}_{n+1}^+ & & \end{aligned} \quad (4)$$

ADAL: Alternating Direction Augmented Lagrangian

ADAL¹ is an ADMM capable to address SDPs in **standard form** (2), based on the augmented Lagrangian built over the dual (3):

$$L_{\sigma}(y, \bar{Z}; \bar{X}) = b^T y - \langle \bar{A}^T y + \bar{Z} - \bar{C}, \bar{X} \rangle - \frac{\sigma}{2} \|\bar{A}^T y + \bar{Z} - \bar{C}\|^2.$$

$L_{\sigma}(y, \bar{Z}; \bar{X})$ is optimized by “alternating” between the y -variables and the \bar{Z} -variables.

At each iteration, the new point $(y^{k+1}, \bar{Z}^{k+1}, \bar{X}^{k+1})$ is computed by the following steps

$$y^{k+1} = \operatorname{argmax}_{y \in \mathbb{R}^m} L_{\sigma^k}(y, \bar{Z}^k; \bar{X}^k), \quad (5)$$

$$\bar{Z}^{k+1} = \operatorname{argmax}_{Z \in \mathcal{S}_n^+} L_{\sigma^k}(y^{k+1}, \bar{Z}; \bar{X}^k), \quad (6)$$

$$\bar{X}^{k+1} = \bar{X}^k + \sigma^k (\bar{A}^T y^{k+1} + \bar{Z}^{k+1} - \bar{C}). \quad (7)$$

¹Povh, Rendl, and Wiegele 2006; Wen, Goldfarb, and Yin 2010

ADAL: Alternating Direction Augmented Lagrangian

Algorithm 1 Scheme of ADAL²

- 1: Choose $\sigma > 0$, $\bar{X} \in \mathcal{S}_{n+l}^+$, $\bar{Z} \in \mathcal{S}_{n+l}^+$
 - 2: **repeat**
 - 3: $y = (\bar{A}\bar{A}^\top)^{-1} \left(\frac{1}{\sigma} b - \bar{A} \left(\frac{1}{\sigma} \bar{X} - \bar{C} + \bar{Z} \right) \right)$
 - 4: $\bar{W} := \bar{X}/\sigma - \bar{C} + \bar{A}^\top y$
 - 5: $\bar{Z} = -(\bar{W})_-$
 - 6: $\bar{X} = \sigma(\bar{W})_+$
 - 7: Update σ
 - 8: **until** convergence
-

²Wen, Goldfarb, and Yin 2010

Memory issues when dealing with inequalities

One can use ADAL “as it is” to solve problems in form of (1) reducing it to standard form (2).

The memory required to store the matrices \bar{C} , \bar{A}_i , \bar{A}_j , \bar{Z} and \bar{X} gets large with the number l of inequalities.

It is *computationally infeasible* to deal with large scale problems, even using efficient sparse matrix implementations.

Idea: rewrite the steps of ADAL in terms of the matrices C , A^i , A^j and X , without performing the reduction to standard form explicitly.

Extending ADAL to deal with inequality constraints

y-update

$$y = (\bar{\mathcal{A}}\bar{\mathcal{A}}^\top)^{-1} \left(\frac{1}{\sigma} b - \bar{\mathcal{A}} \left(\frac{1}{\sigma} \bar{X} - \bar{C} + \bar{Z} \right) \right)$$

The following on the linear map $\bar{\mathcal{A}}$ applied to \bar{X} holds:

$$\bar{\mathcal{A}}(\bar{X}) = \begin{pmatrix} \langle \bar{A}^1, \bar{X} \rangle \\ \vdots \\ \langle \bar{A}^l, \bar{X} \rangle \\ \langle \bar{A}^{l+1}, \bar{X} \rangle \\ \vdots \\ \langle \bar{A}^m, \bar{X} \rangle \end{pmatrix} = \begin{pmatrix} \langle A^1, X \rangle + s_1 \\ \vdots \\ \langle A^l, X \rangle + s_l \\ \langle A^{l+1}, X \rangle \\ \vdots \\ \langle A^m, X \rangle \end{pmatrix} = \mathcal{A}(X) + \begin{pmatrix} s^T \\ 0_{m-l} \end{pmatrix}$$

Extending ADAL to deal with inequality constraints

y-update

$$y = (\bar{\mathcal{A}}\bar{\mathcal{A}}^\top)^{-1} \left(\frac{1}{\sigma} b - \mathcal{A} \left(\frac{1}{\sigma} X - C + Z \right) + \begin{pmatrix} \frac{1}{\sigma} s^\top + p^\top \\ 0_{m-l} \end{pmatrix} \right)$$

The operator $\bar{\mathcal{A}}\bar{\mathcal{A}}^\top : \mathbb{R}^m \rightarrow \mathbb{R}^m$ can be rewritten as

$$\bar{\mathcal{A}}\bar{\mathcal{A}}^\top = \mathcal{A}\mathcal{A}^\top + \text{diag} \begin{pmatrix} 1_l \\ 0_{m-l} \end{pmatrix},$$

since the 1 in position $(n+i, n+i)$ of \bar{A}^i , $i = 1, \dots, l$ contributes in the row-by-column product only in position (i, i) .

Extending ADAL to deal with inequality constraints

y-update

$$y = \left(\mathcal{A}\mathcal{A}^\top + \text{diag} \begin{pmatrix} \mathbf{1}_l \\ \mathbf{0}_{m-l} \end{pmatrix} \right)^{-1} \left(\frac{1}{\sigma} b - \mathcal{A} \left(\frac{1}{\sigma} X - C + Z \right) + \begin{pmatrix} \frac{1}{\sigma} s^\top + \rho^\top \\ \mathbf{0}_{m-l} \end{pmatrix} \right)$$

The operator $\bar{\mathcal{A}}\bar{\mathcal{A}}^\top : \mathbb{R}^m \rightarrow \mathbb{R}^m$ can be rewritten as

$$\bar{\mathcal{A}}\bar{\mathcal{A}}^\top = \mathcal{A}\mathcal{A}^\top + \text{diag} \begin{pmatrix} \mathbf{1}_l \\ \mathbf{0}_{m-l} \end{pmatrix},$$

since the 1 in position $(n+i, n+i)$ of \bar{A}^i , $i = 1, \dots, l$ contributes in the row-by-column product only in position (i, i) .

Extending ADAL to deal with inequality constraints

\bar{X} and \bar{Z} -update

$$\bar{W} = \frac{\bar{X}}{\sigma} - \bar{C} + \bar{\mathcal{A}}^T y$$

The adjoint operator $\bar{\mathcal{A}}^T : \mathbb{R}^m \rightarrow \mathcal{S}_{n+l}$ of $\bar{\mathcal{A}}$ is defined as

$$\bar{\mathcal{A}}^T y = \sum_m^{i=1} y_i \bar{\mathcal{A}}^i = \begin{pmatrix} \mathcal{A}^T y & & 0_{n,l} \\ & y_1 & \\ 0_{n,l} & & \ddots \\ & & & y_l \end{pmatrix}$$

Extending ADAL to deal with inequality constraints

\bar{X} and \bar{Z} -update

$$\bar{W} = \begin{pmatrix} \frac{X}{\sigma} - C + \mathcal{A}^T y & 0_{n,l} \\ 0_{n,l} & \text{Diag} \left(\frac{s^T}{\sigma} + \begin{pmatrix} y_1 \\ \vdots \\ y_l \end{pmatrix} \right) \end{pmatrix}$$

The projection of \bar{W} is computed via the spectral decomposition.

In order to compute eigenvectors and eigenvalues of \bar{W} we can process it “block-wise”.

Then update \bar{Z} and \bar{X} accordingly.

On the convergence of ADAL

The correctness and convergence of the scheme introduced is inherited by the convergence of ADAL³.

In particular, Algorithm 1 can be interpreted as a fixed point method and we can state the following result:

Theorem

The sequence $\{(\bar{X}^k, y^k, \bar{Z}^k)\}$ generated by Algorithm 1 from any starting point $(\bar{X}^0, y^0, \bar{Z}^0)$ converges to a solution $(\bar{X}^, y^*, \bar{Z}^*) \in \Omega^*$, where Ω^* is the set of primal and dual solutions of (2) and (3).*

³Wen, Goldfarb, and Yin 2010

Obtaining valid dual bounds: a post-processing procedure

Every dual feasible solution of an SDP relaxation yields a valid bound on the optimal value of the related CO problem.

Following ideas developed in Cerulli et al. 2020, let $\tilde{Z} \in \mathcal{S}_n^+$. If the LP

$$\begin{aligned} \max \quad & -b_{ineq}^T \lambda + b_{eq}^T \mu \\ \text{s.t.} \quad & C + \mathcal{A}_{ineq}^T \lambda - \mathcal{A}_{eq}^T \mu = \tilde{Z} \\ & \lambda \geq 0 \end{aligned} \tag{8}$$

has an optimal solution $\tilde{y} = (\tilde{\lambda}, \tilde{\mu}) \in \mathbb{R}^m$, then (\tilde{y}, \tilde{Z}) is a feasible solution for the dual.

Idea: To approximately solve the primal (1) with ADAL to get $\tilde{Z} \in \mathcal{S}_n^+$, then try to get a dual feasible solution (and then a valid bound) by addressing problem (8).

Computational Experience

Comparison with SDPNAL+⁴, the state-of-the-art method for solving large-scale SDPs that has been awarded with the Beale-Orchard-Hays Prize in 2018.

Implementation of Extended ADAL in MATLAB and GUROBI as linear programming solver for the post-processing procedure (8).

Benchmarks performed on:

- Random SDPs instances
- **Maximum Stable Set** and **Graph Coloring** SDP relaxations on DIMACS⁵ graphs

Performance of the algorithms compared using performance profiles as proposed by Dolan and J.Moré 2002.

⁴Yang, Sun, and Toh 2015

⁵Johnson and Trick 1996

Computational Experience: Random Instances

The random instances are obtained from the instance generator used in Malick et al. 2009.

- n : dimension of the matrices
- m : total number of constraints
- p : percentage of inequality constraints

CPU Time limit: 1800 seconds

Accuracy: 10^{-5}

Computational Experience: Random Instances

n	m	p (%)	ADAL		SDPNAL+	
			#sol	CPU time	#sol	CPU time
250	25000	25	5	838.04	0	-
		50	5	1166.45	0	-
		75	5	1114.52	0	-
500	50000	25	5	217.61	5	106.28
		50	5	260.43	5	221.66
		75	5	325.71	5	250.97
1000	10000	25	5	136.63	5	49.52
		50	5	157.21	5	58.22
		75	5	242.63	5	71.38
	50000	25	5	57.19	5	60.96
		50	5	94.09	5	109.48
		75	5	110.00	5	111.29
	100000	25	5	83.15	5	136.53
		50	5	127.37	5	181.13
		75	5	155.05	5	184.21

Table: Results on random instances

Computational Experience: Random Instances

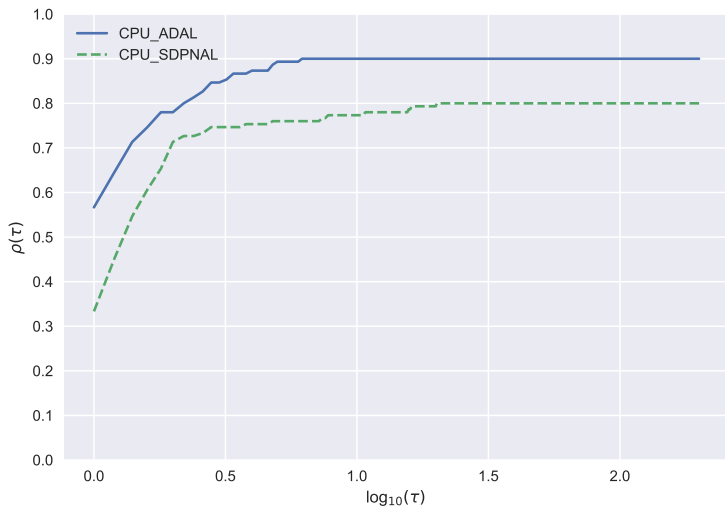


Figure: Performance profiles on CPU time. Comparison between ADAL and SDPNAL+ on random instances (Higher is better)

Computational Experience: Maximum Stable Set

Given an undirected graph $G = (V, E)$, where V is the set of vertices and E is the set of edges, a subset of $S \subseteq V$ is called *stable* if no two vertices in S are adjacent. The **stability number** $\alpha(G)$ denotes the maximum cardinality of a stable set in G .

We considered the following strengthening of the *theta number*⁶

$$\begin{aligned} \vartheta_+(G) = \quad & \max \quad \langle J, X \rangle \\ & \text{s.t.} \quad \text{trace}(X) = 1 \\ & \quad X_{ij} = 0 \quad \{i, j\} \in E(G) \\ & \quad X \geq 0 \\ & \quad X \in \mathcal{S}_n^+ \end{aligned}$$

⁶Lovász 1979

Computational Experience: Maximum Stable Set

Instances: from the Second DIMACS Implementation challenge⁷

Post-processing procedure called every 200 iterations of ADAL and at the very last iteration.

Keep in memory the best dual bound found by the post-processing procedure (and then a valid bound on $\alpha(G)$)

CPU time limit: 3600 seconds

Tolerance: 10^{-6}

⁷Johnson and Trick 1996

Computational Experience: Maximum Stable Set

Graph	$\vartheta_+(G)$			CPU times		
	ADAL	SDPNAL+	BestBound	ADAL	SDPNAL+	BestBound
DSJC1000-5	31.67	31.67	31.67	41.60	34.32	43.69
C500-9	83.58	83.58	83.58	30.55	8.32	31.03
C2000-5	44.56	44.56	44.56	389.59	534.67	398.86
brock800_1	41.87	41.87	41.87	24.31	11.67	20.96
brock800_3	41.88	41.88	41.88	24.52	13.02	25.87
p_hat300-2	26.71	26.71	26.71	211.40	161.90	28.17
p_hat500-2	38.56	38.56	38.56	580.86	537.38	92.52
p_hat700-2	48.44	48.44	48.44	1161.67	295.99	218.26
p_hat1000-2	54.84	54.84	54.84	1815.65	697.11	487.41
p_hat1500-2	-	-	76.46	-	-	1826.66
p_hat1500-3	113.65	113.65	113.65	3014.42	879.45	1886.51
keller4	13.47	13.47	13.47	3.35	1.46	2.69
sanr400_0.5	20.18	20.18	20.18	6.60	3.80	6.19
sanr400_0.7	33.97	33.97	33.97	7.21	4.05	7.49
hamming8-4	16.00	16.00	16.00	2.62	1.13	2.60
hamming10-4	42.67	42.67	42.67	97.36	31.77	93.90

Table: Results on $\vartheta_+(G)$, graphs from the second DIMACS implementation challenge.

Computational Experience: Maximum Stable Set

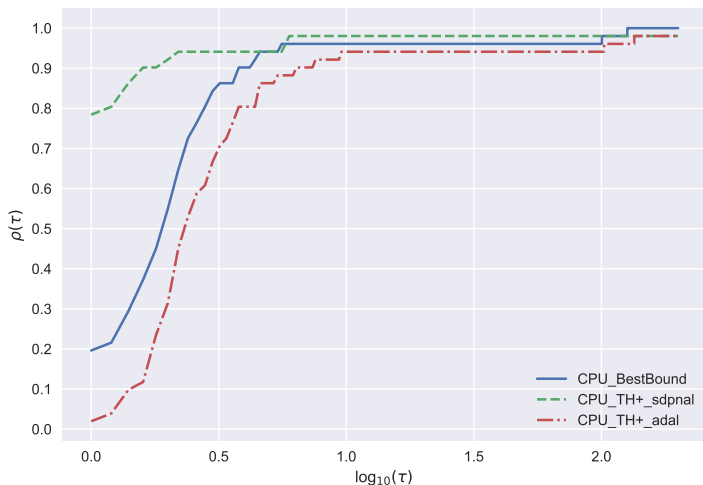


Figure: Performance profiles on CPU time. Comparison among ADAL, BestBound and SDPNAL+ on the computation of $\vartheta_+(G)$. (Higher is better)

Computational Experience: Graph Coloring

Given an undirected graph $G = (V, E)$, a k -coloring is a partition of V into k stable sets. The *chromatic number* $\chi(G)$ is the smallest integer k for which G has a k -coloring.

We consider the following formulation for $\bar{\vartheta}_+(G)$ reported in Laurent and Rendl 2005:

$$\begin{aligned} \bar{\vartheta}_+(G) = \quad & \min \quad t \\ & \text{s.t.} \quad X_{ii} = t - 1 \quad i \in V(G) \\ & \quad \quad X_{ij} = -1 \quad \{i, j\} \in \bar{E}(G) \\ & \quad \quad X_{ij} \geq -1 \quad \{i, j\} \in E(G) \\ & \quad \quad X \in \mathcal{S}_n^+. \end{aligned}$$

Computational Experience: Graph Coloring

Instances: from the Second DIMACS Implementation challenge⁷

Post-processing procedure called every 200 iterations of ADAL and at the very last iteration.

Keep in memory the best dual bound found by the post-processing procedure (and then a valid bound on $\chi(G)$)

CPU time limit: 3600 seconds

Tolerance: 10^{-6}

⁷Johnson and Trick 1996

Computational Experience: Graph Coloring

Graph	$\bar{\nu}_+(G)$			CPU times		
	ADAL	SDPNAL+	BestBound	ADAL	SDPNAL+	BestBound
DSJC125.1	4.14	4.14	4.14	69.01	22.88	1.72
DSJR500.1c	-	83.75	83.75	-	1231.74	190.31
inithx.i.3	31.00	31.00	30.23	341.12	13.64	32.57
mulsol.i.1	49.00	49.00	-	18.89	3.48	-
school1	14.00	14.00	14.00	14.74	65.03	8.08
myciel7	2.82	2.82	2.82	7.35	7.60	1.34
mug100_1	3.00	3.00	3.00	19.59	84.51	0.46
mug100_25	3.00	3.00	3.00	26.20	84.97	0.46
ash608GPIA	3.33	3.33	3.31	265.72	41.34	129.25
ash958GPIA	3.33	3.33	-	529.68	124.35	-
1-Insertions_6	2.31	2.31	2.31	337.22	100.65	22.80
2-Insertions_5	2.16	2.16	2.16	544.91	109.90	52.67
3-Insertions_4	2.09	2.09	2.09	125.48	29.79	8.39
4-Insertions_4	2.06	2.06	2.06	563.58	130.23	8.89
2-FullIns_5	4.08	4.08	4.08	2670.31	184.26	381.59
5-FullIns_4	-	7.01	7.01	-	207.83	137.49
wap03a	40.00	40.00	40.00	1594.69	2668.31	1507.80
wap07a	40.00	40.00	40.00	309.93	426.97	145.89

Table: Results on $\bar{\nu}_+(G)$, graphs from the second DIMACS implementation challenge.

Computational Experience: Graph Coloring

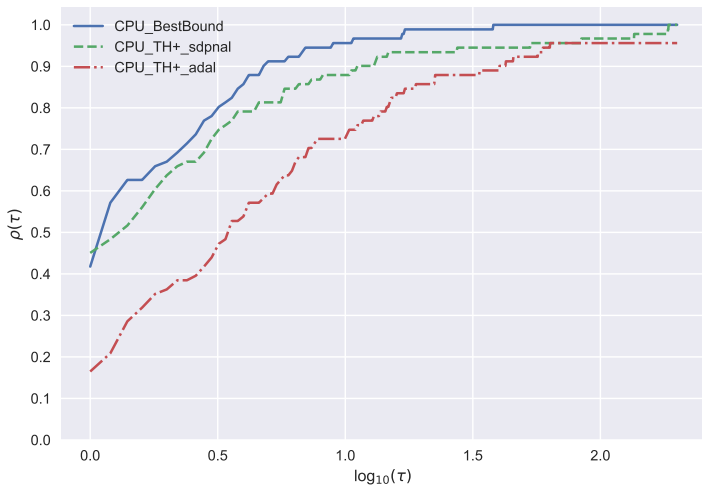


Figure: Performance profiles on CPU time. Comparison among ADAL, BestBound and SDPNAL+ on the computation of $\bar{\nu}_+(G)$. (Higher is better)

Conclusions

We proposed a numerical comparison between ADAL, an ADMM method for SDPs in general form and SDPNAL+, the state-of-the-art method for solving large-scale SDPs.

Despite SDPNAL+ is more robust than ADAL, we could detect classes of instances where our proposal is competitive.

The post-processing procedure allows to find dual feasible solutions, which give a valid bound on optimal value of the primal.





When solving SDP relaxations of CO problems, it allows to stop the execution of the ADMM as soon as a “good” bound is obtained.

Furthermore, since a dual feasible solution is detected, it allows to use reoptimization techniques within branch-and-bound frameworks.





Thanks for your attention!
Questions?

Preprint <https://arxiv.org/abs/2106.12411>

References I

-  Cerulli, Martina et al. (2020). “Improving ADMMs for solving doubly nonnegative programs through dual factorization”. In: *4OR*, pp. 1–34.
-  Dolan, E. and J.Moré (2002). “Benchmarking optimization software with performance profiles”. In: *Mathematical Programming 91*, pp. 201–213.
-  Johnson, David J. and Michael A. Trick, eds. (1996). *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, Workshop, October 11-13, 1993*. American Mathematical Society. ISBN: 0821866095.
-  Laurent, Monique and Franz Rendl (2005). “Semidefinite Programming and Integer Programming”. In: *Discrete Optimization*. Ed. by K. Aardal, G.L. Nemhauser, and R. Weismantel. Vol. 12. Handbooks in Operations Research and Management Science. Amsterdam, The Netherlands: Elsevier. Chap. 8, pp. 393–514.

References II

-  Lovász, László (1979). “On the Shannon capacity of a graph”. In: *IEEE Transactions on Information theory* 25.1, pp. 1–7.
-  Malick, Jérôme et al. (2009). “Regularization methods for Semidefinite Programming”. In: *SIAM J. Optim.* 20.1, pp. 336–356.
-  Povh, Janez, Franz Rendl, and Angelika Wiegele (2006). “A Boundary Point Method to solve Semidefinite Programs”. In: *Computing* 78, pp. 277–286.
-  Wen, Zaiwen, Donald Goldfarb, and Wotao Yin (2010). “Alternating direction augmented Lagrangian methods for semidefinite programming”. In: *Mathematical Programming Computation* 2.3, pp. 203–230.

References III



Yang, Liuqin, Defeng Sun, and Kim-Chuan Toh (2015).
“SDPNAL+: a majorized semismooth Newton-CG augmented
Lagrangian method for semidefinite programming with
nonnegative constraints”. In: *Mathematical Programming
Computation* 7.3, pp. 331–366.

Performance Profiles

Given a set of solvers \mathcal{S} and a set of problems \mathcal{P} , the performance of a solver $s \in \mathcal{S}$ on problem $p \in \mathcal{P}$ is compared against the best performance obtained by any solver in \mathcal{S} on the same problem. The performance ratio is defined as

$$r_{p,s} = t_{p,s} / \min\{t_{p,s'} \mid s' \in \mathcal{S}\},$$

where $t_{p,s}$ is the measure we want to compare, and we consider a cumulative distribution function

$$\rho_s(\tau) = |\{p \in \mathcal{P} \mid r_{p,s} \leq \tau\}| / |\mathcal{P}|.$$

The performance profile for $s \in \mathcal{S}$ is the plot of the function ρ_s .