Solving SDPs by using sparse PCA

Santanu S. Dey[1] Aleksandr M. Kazachkov[2] Andrea Lodi[3]
Gonzalo Muñoz[4]

[1]Georgia Institute of Technology.

[2]University of Florida. [3]École Polytechnique de Montréal.

[4]Universidad de O'Higgins.

Thanks to A. M. Kazachkov for all the figures in the slides.

Feb 2021

$$\underbrace{\text{Solving SDPs}}_{\text{Related to Comb. Opt.}} \quad \text{by using} \quad \underbrace{\text{sparse PCA}}_{\text{Mach. Learning/Statistics}}$$

Santanu S. Dey[1] Aleksandr M. Kazachkov[2] Andrea Lodi[3]
Gonzalo Muñoz[4]

[1]Georgia Institute of Technology.

[2]University of Florida. [3]École Polytechnique de Montréal.

[4]Universidad de O'Higgins.

Thanks to A. M. Kazachkov for all the figures in the slides.

Feb 2021

1 The challenge – creating lightweight outer approximation of SDPs

Semi-definite programming

$$\begin{array}{ll} \min & \langle C, X\rangle \\ \text{s.t.} & \langle A^i, X\rangle \le b_i \quad \forall i \in \{1, \dots, m\} \\ & X \in \mathcal{S}^n_+, \end{array} \tag{SDP}$$

where C and the A^i's are $n \times n$ matrices, $\langle M, N\rangle := \sum_{i,j} M_{ij}N_{ij}$, and

Semi-definite programming

$$\begin{aligned} \min \quad & \langle C, X \rangle \\ \text{s.t.} \quad & \langle A^i, X \rangle \le b_i \quad \forall i \in \{1, \ldots, m\} \\ & X \in \mathcal{S}^n_+, \end{aligned} \tag{SDP}$$

where C and the A^i's are $n \times n$ matrices, $\langle M, N \rangle := \sum_{i,j} M_{ij} N_{ij}$, and

$$\mathcal{S}^n_+ = \{X \in \mathbb{R}^{n \times n} \,|\, X = X^T,\ u^\top X u \ge 0,\ \forall u \in \mathbb{R}^n\}.$$

Semi-definite programming

$$\begin{array}{lll} \min & \langle C, X \rangle & \\ \text{s.t.} & \langle A^i, X \rangle \leq b_i \quad \forall i \in \{1, \ldots, m\} & \\ & X \in \mathcal{S}_+^n, & \end{array} \tag{SDP}$$

where C and the A^i's are $n \times n$ matrices, $\langle M, N \rangle := \sum_{i,j} M_{ij} N_{ij}$, and

$$\mathcal{S}_+^n = \{X \in \mathbb{R}^{n \times n} \,|\, X = X^T,\ u^\top X u \geq 0,\ \forall u \in \mathbb{R}^n\}.$$

- Polynomial-time algorithm – but often challenging to solve in practice.

Semi-definite programming

$$\begin{array}{lll} \min & \langle C, X\rangle & \\ \text{s.t.} & \langle A^i, X\rangle \leq b_i \quad \forall i \in \{1, \dots, m\} & \\ & X \in \mathcal{S}^n_+, & \end{array} \qquad \text{(SDP)}$$

where C and the A^i's are $n \times n$ matrices, $\langle M, N\rangle := \sum_{i,j} M_{ij} N_{ij}$, and

$$\mathcal{S}^n_+ = \{X \in \mathbb{R}^{n \times n} \,|\, X = X^T,\ u^\top X u \geq 0,\ \forall u \in \mathbb{R}^n\}.$$

- Polynomial-time algorithm – but often challenging to solve in practice.
- In many applications (Combinatorial optimization, non-convex quadratic problems), the SDP is a relaxation of a more challenging nonconvex problem, which needs to be solved using the (spacial) branch-and-bound algorithm. We do not want to maintain a SDP relaxation – since it would have to solved at each node — but maintain a linear programming relaxation.

Semi-definite programming: solving via linear cutting-planes

1. Construct a linear programming relaxation:

$$
\begin{array}{lll}
\min & \langle C, X \rangle & \\
\text{s.t.} & \langle A^i, X \rangle \leq b_i \quad \forall i \in \{1, \dots, m\} & \text{(LP-Relax-of-SDP)} \\
& \cancel{X \in \mathcal{S}^n_+} &
\end{array}
$$

Semi-definite programming: solving via linear cutting-planes

1. Construct a linear programming relaxation:

$$\begin{array}{ll} \min & \langle C, X \rangle \\ \text{s.t.} & \langle A^i, X \rangle \le b_i \quad \forall i \in \{1, \ldots, m\} \\ & \cancel{X \in \mathcal{S}^n_+} \end{array} \qquad \text{(LP-Relax-of-SDP)}$$

2. Let optimal solution be $\tilde{X}$. If $\tilde{X} \in \mathcal{S}^n_+$, then we have solved the SDP.

Semi-definite programming: solving via linear cutting-planes

1. Construct a linear programming relaxation:

$$\begin{array}{ll} \min & \langle C, X\rangle \\ \text{s.t.} & \langle A^i, X\rangle \leq b_i \quad \forall i \in \{1, \dots, m\} \\ & \cancel{X \in \mathcal{S}^n_+} \end{array} \qquad \text{(LP-Relax-of-SDP)}$$

2. Let optimal solution be $\tilde{X}$. If $\tilde{X} \in \mathcal{S}^n_+$, then we have solved the SDP.
3. Else,

$$\tilde{X} = \sum_{i=1}^{n} \lambda_i v^i (v^i)^T \text{ and, say } \lambda_1 < 0,$$

Semi-definite programming: solving via linear cutting-planes

1. Construct a linear programming relaxation:

$$\begin{array}{ll} \min & \langle C, X \rangle \\ \text{s.t.} & \langle A^i, X \rangle \leq b_i \quad \forall i \in \{1, \ldots, m\} \\ & \cancel{X \in \mathcal{S}^n_+} \end{array} \qquad \text{(LP-Relax-of-SDP)}$$

2. Let optimal solution be $\tilde{X}$. If $\tilde{X} \in \mathcal{S}^n_+$, then we have solved the SDP.
3. Else,

$$\tilde{X} = \sum_{i=1}^{n} \lambda_i v^i (v^i)^T \text{ and, say } \lambda_1 < 0,$$

4. Let $V = v^1 (v^1)^\top$. Then

$$\boxed{\langle V, \tilde{X} \rangle = \lambda_1 < 0, \text{ but } \langle V, X \rangle \geq 0 \ \text{ for all } \ X \in \mathcal{S}^n_+}$$

Semi-definite programming: solving via linear cutting-planes

1. Construct a linear programming relaxation:

$$\begin{array}{ll} \min & \langle C, X\rangle \\ \text{s.t.} & \langle A^i, X\rangle \leq b_i \quad \forall i \in \{1, \dots, m\} \\ & \cancel{X \in \mathcal{S}^n_+} \end{array} \qquad \text{(LP-Relax-of-SDP)}$$

2. Let optimal solution be $\tilde{X}$. If $\tilde{X} \in \mathcal{S}^n_+$, then we have solved the SDP.
3. Else,

$$\tilde{X} = \sum_{i=1}^{n} \lambda_i v^i (v^i)^T \text{ and, say } \lambda_1 < 0,$$

4. Let $V = v^1 (v^1)^\top$. Then

$$\boxed{\langle V, \tilde{X}\rangle = \lambda_1 < 0, \text{ but } \langle V, X\rangle \geq 0 \ \text{ for all } \ X \in \mathcal{S}^n_+}$$

5. Update, (LP-Relax-of -SDP) as:

$$\begin{array}{ll} \min & \langle C, X\rangle \\ \text{s.t.} & \langle A^i, X\rangle \leq b_i \quad \forall i \in \{1, \dots, m\} \\ & \langle V, X\rangle \geq 0. \end{array}$$

Go to Step 2.

Lets see an example of this at work...

- Box QP:

$$\begin{array}{ll} \min & x^\top Q x + c^\top x \\ \text{s.t.} & x \in [0,1]^n \end{array}$$

Lets see an example of this at work...

- Box QP:

$$\begin{aligned} \min \quad & x^\top Qx + c^\top x \\ \text{s.t.} \quad & x \in [0,1]^n \end{aligned}$$

- SDP (+ McCormick) relaxation:

$$\begin{aligned} \min \quad & \langle Q, xx^T \rangle + c^\top x \\ \text{s.t.} \quad & x \in [0,1]^n \end{aligned}$$

Lets see an example of this at work...

- Box QP:

$$\begin{array}{ll} \min & x^\top Q x + c^\top x \\ \text{s.t.} & x \in [0,1]^n \end{array}$$

- SDP (+ McCormick) relaxation:

$$\begin{array}{ll} \min & \langle Q, xx^T \rangle + c^\top x \\ \text{s.t.} & x \in [0,1]^n \end{array}$$

$$\begin{array}{ll} \min & \langle Q, X \rangle + c^\top x \\ \text{s.t.} & x \in [0,1]^n, {\color{red} X = xx^T} \end{array}$$

Lets see an example of this at work...

- Box QP:

$$\begin{aligned} \min \quad & x^\top Q x + c^\top x \\ \text{s.t.} \quad & x \in [0,1]^n \end{aligned}$$

- SDP (+ McCormick) relaxation:

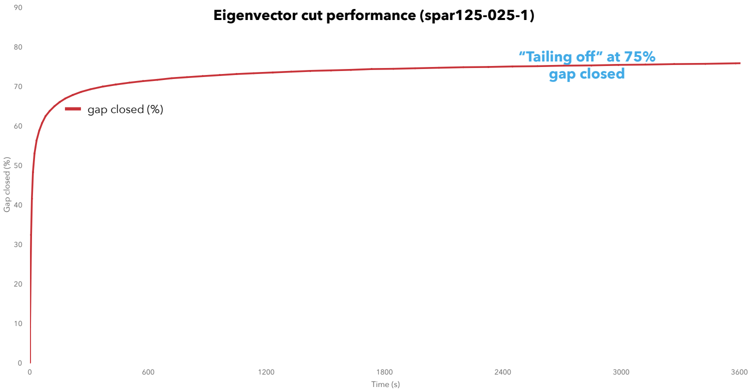
$$\begin{aligned} \min \quad & \langle Q, xx^T \rangle + c^\top x \\ \text{s.t.} \quad & x \in [0,1]^n \end{aligned}$$

$$\begin{aligned} \min \quad & \langle Q, X \rangle + c^\top x \\ \text{s.t.} \quad & x \in [0,1]^n, X = xx^T \end{aligned}$$

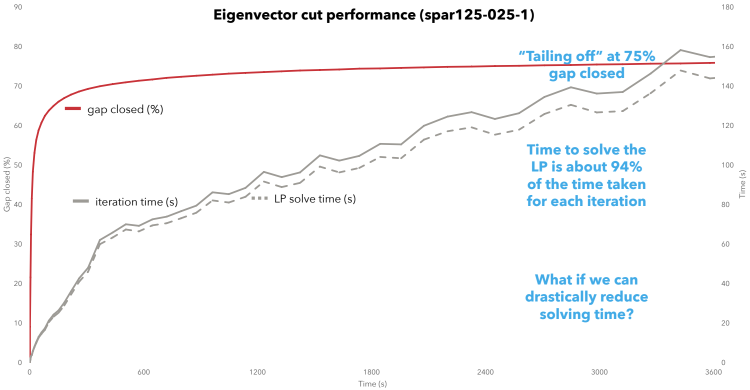
$$\begin{aligned} \min \quad & \langle Q, X \rangle + c^\top x \\ \text{s.t.} \quad & \max\{0, x_i + x_j - 1\} \le X_{ij} \le \min\{x_i, x_j\} \; i,j \in [n] \\ & \begin{bmatrix} 1 & x^\top \\ x & X \end{bmatrix} \in \mathcal{S}_+^n. \end{aligned}$$

- *spar125-025-1* (https://github.com/sburer/BoxQPinstances): $n = 125$ and Q has 25% non-zeros.

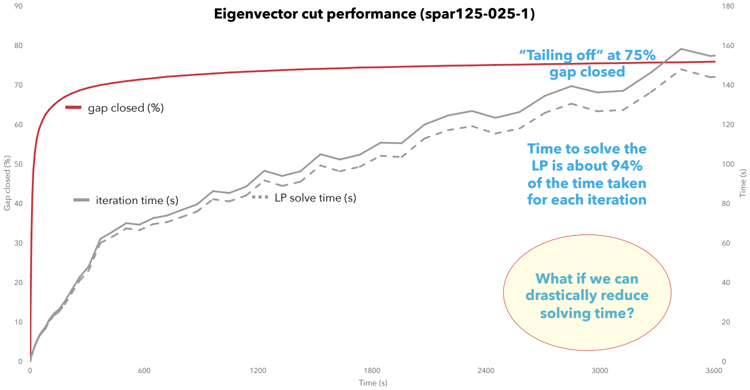
Gap closed vs. time



Gap closed vs. time



Gap closed vs. time



Why are the LPs becoming so difficult to solve?

- $n = 125$. So LP has 7875 variables.

Why are the LPs becoming so difficult to solve?

- $n = 125$. So LP has 7875 variables.
- $$\tilde{X} = \sum_{i=1}^{n} \lambda_i v^i (v^i)^T \text{ and, say } \lambda_1 < 0,$$

  Typically, the eigenvector (v^1) corresponding to the negative eigenvalue is dense.

Why are the LPs becoming so difficult to solve?

- $n = 125$. So LP has 7875 variables.
- $$\tilde{X} = \sum_{i=1}^{n} \lambda_i v^i (v^i)^T \text{ and, say } \lambda_1 < 0,$$

  Typically, the eigenvector (v^1) corresponding to the negative eigenvalue is dense.
- Therefore, the inequality

  $$\langle V, X \rangle \geq 0,$$

  is a completely dense inequality in an LP which is already “largish".

Why are the LPs becoming so difficult to solve?

- $n = 125$. So LP has 7875 variables.
- $$\tilde{X} = \sum_{i=1}^{n} \lambda_i v^i (v^i)^T \text{ and, say } \lambda_1 < 0,$$

  Typically, the eigenvector (v^1) corresponding to the negative eigenvalue is dense.
- Therefore, the inequality

  $$\langle V, X \rangle \geq 0,$$

  is a completely dense inequality in an LP which is already “largish".
- As these dense inequalities keep getting added, the LP solve times increase.

Why are the LPs becoming so difficult to solve?

- $n = 125$. So LP has 7875 variables.
- $$\tilde{X} = \sum_{i=1}^{n} \lambda_i v^i (v^i)^T \text{ and, say } \lambda_1 < 0,$$

  Typically, the eigenvector (v^1) corresponding to the negative eigenvalue is dense.
- Therefore, the inequality

  $$\langle V, X \rangle \geq 0,$$

  is a completely dense inequality in an LP which is already “largish".
- As these dense inequalities keep getting added, the LP solve times increase.
- On the other hand, LP solvers love sparsity (Many linear algebra routine can exploit sparsity)!

Conclusion

- It is no enough to just explore linear programming relaxations of SDPs.

Conclusion

- It is no enough to just explore linear programming relaxations of SDPs.
- What we really want is *sparse linear programming relaxations to SDPs.*

Conclusion

- It is no enough to just explore linear programming relaxations of SDPs.
- What we really want is *sparse linear programming relaxations to SDPs.*
- Lets decide a sparsity level $k << n$: If $\tilde{X} \notin S^n_+$, then find a vector v such that

$$v^\top \tilde{X} v < 0, \|v\|_0 \leq k,$$

then $V = vv^\top$ will be sparse.

Conclusion

- It is no enough to just explore linear programming relaxations of SDPs.
- What we really want is *sparse linear programming relaxations to SDPs.*
- Lets decide a sparsity level $k << n$: If $\tilde{X} \notin S^n_+$, then find a vector v such that

$$v^\top \tilde{X} v < 0, \|v\|_0 \leq k,$$

  then $V = vv^\top$ will be sparse.
- Questions:
  1. How to find such *"sparse eigenvectors"*?

Conclusion

- It is no enough to just explore linear programming relaxations of SDPs.
- What we really want is *sparse linear programming relaxations to SDPs.*
- Lets decide a sparsity level $k << n$: If $\tilde{X} \notin S^n_+$, then find a vector v such that

$$v^\top \tilde{X} v < 0, \|v\|_0 \leq k,$$

  then $V = vv^\top$ will be sparse.
- Questions:
  1. How to find such *"sparse eigenvectors"*?
  2. Will such sparse linear approximation of SDPs work?

Conclusion

- It is no enough to just explore linear programming relaxations of SDPs.
- What we really want is *sparse linear programming relaxations to SDPs.*
- Lets decide a sparsity level $k << n$: If $\tilde{X} \notin S^n_+$, then find a vector v such that

$$v^\top \tilde{X} v < 0, \|v\|_0 \leq k,$$

  then $V = vv^\top$ will be sparse.
- Questions:
  1. How to find such *"sparse eigenvectors"*?
  2. Will such sparse linear approximation of SDPs work?
- Some previous papers dealing with the same topic: [A. Qualizza, P. Belotti, and F. Margot (2012)], [R. Baltean-Lugojan, P. Bonami, R. Misener, and A. Tramontani (2018)]

2

Our key technique - Sparse PCA : a well studied problem in statistics/ML literature

Principal component analysis (PCA) and sparse PCA

- Let $\bar{X} \in S_+^n$ (covariance matrix), the PCA problem:

$$\left.\begin{array}{rcl} w^* \in & \operatorname{argmax}_w & w^\top \bar{X} w \\ & \text{s.t.} & \|w\|_2 \leq 1 \end{array}\right\} \text{PCA}$$

Principal component analysis (PCA) and sparse PCA

- Let $\bar{X} \in S^n_+$ (covariance matrix), the PCA problem:

$$\left.\begin{array}{rll} w^* \in & \text{argmax}_w & w^\top \bar{X} w \\ & \text{s.t.} & \|w\|_2 \leq 1 \end{array}\right\} \text{PCA}$$

  - The optimal objective function of the above problem is the largest eigen value of $\bar{X}$.
  - Given $\bar{X}$ is covariance matrix – the optimal solution of the above problem can be interpreted as the the direction that captures the maximum variance.
  - If w^* is dense, this this is no ideal from interpretability.

Principal component analysis (PCA) and sparse PCA

- Let $\bar{X} \in S^n_+$ (covariance matrix), the PCA problem:

$$\left.\begin{array}{rll} w^* \in & \text{argmax}_w & w^\top \bar{X} w \\ & \text{s.t.} & \|w\|_2 \leq 1 \end{array}\right\} \text{PCA}$$

  - The optimal objective function of the above problem is the largest eigen value of $\bar{X}$.
  - Given $\bar{X}$ is covariance matrix – the optimal solution of the above problem can be interpreted as the the direction that captures the maximum variance.
  - If w^* is dense, this this is no ideal from interpretability.
-

$$\left.\begin{array}{rll} \tilde{w} := & \text{argmax}_w & w^\top \bar{X} w \\ & \text{s.t.} & \|w\|_2 \leq 1 \\ & & \|w\|_0 \leq k \end{array}\right\} \text{Sparse PCA}$$

How does SPCA help?

Goal (given $\tilde{X}$):

$$\text{Find } v : \text{ such that } v^\top \tilde{X} v < 0, \|v\|_2 \leq 1, \|v\|_0 \leq k$$

SPCA (assuming $\bar{X} \in S^n_+$) :

$$\text{Solve } : \max v^\top \bar{X} v : \|v\|_2 \leq 1, \|v\|_0 \leq k$$

How does SPCA help?

Goal (given $\tilde{X}$):

$$\text{Find } v : \text{ such that } v^\top \tilde{X} v < 0, \|v\|_2 \leq 1, \|v\|_0 \leq k$$

SPCA (assuming $\bar{X} \in S^n_+$) :

$$\text{Solve } : \max v^\top \bar{X} v : \|v\|_2 \leq 1, \|v\|_0 \leq k$$

$\tilde{X} \rightarrow$

How does SPCA help?

Goal (given $\tilde{X}$):

$$\text{Find } v : \text{ such that } v^\top \tilde{X} v < 0, \|v\|_2 \leq 1, \|v\|_0 \leq k$$

SPCA (assuming $\bar{X} \in S^n_+$) :

$$\text{Solve } : \max v^\top \bar{X} v : \|v\|_2 \leq 1, \|v\|_0 \leq k$$

$$\tilde{X} \rightarrow -\tilde{X}(\text{ Not necessarily PSD })$$

How does SPCA help?

Goal (given $\tilde{X}$):

$$\text{Find } v: \text{ such that } v^\top \tilde{X} v < 0, \|v\|_2 \leq 1, \|v\|_0 \leq k$$

SPCA (assuming $\bar{X} \in S^n_+$) :

$$\text{Solve} : \max v^\top \bar{X} v : \|v\|_2 \leq 1, \|v\|_0 \leq k$$

$$\tilde{X} \to -\tilde{X}(\text{ Not necessarily PSD }) \to \underbrace{-\tilde{X} + \lambda^{max}(\tilde{X})I}_{\bar{X}}$$

How does SPCA help?

Goal (given $\tilde{X}$):

$$\text{Find } v : \ \text{such that } v^\top \tilde{X} v < 0, \|v\|_2 \leq 1, \|v\|_0 \leq k$$

SPCA (assuming $\bar{X} \in S^n_+$) :

$$\text{Solve} \ : \ \max v^\top \bar{X} v : \|v\|_2 \leq 1, \|v\|_0 \leq k$$

$$\tilde{X} \rightarrow -\tilde{X}(\text{ Not necessarily PSD }) \rightarrow \underbrace{-\tilde{X} + \lambda^{max}(\tilde{X}) I}_{\bar{X}}$$

Proposition (Sparse separation via SPCA)

- *If* $\max \left\{ v^\top \left(-\tilde{X} + \lambda^{max}(\tilde{X}) I \right) v : \|v\|_2 \leq 1, \|v\|_0 \leq k \right\} \leq \lambda^{max}(\tilde{X})$, *then there is no sparse inequality.*
- *If* $\max \left\{ v^\top \left(-\tilde{X} + \lambda^{max}(\tilde{X}) I \right) v : \|v\|_2 \leq 1, \|v\|_0 \leq k \right\} > \lambda^{max}(\tilde{X})$, *then if v^* is the optimal solution then $\langle v^*(v^*)^\top, X \rangle \geq 0$ is a sparse separating hyperplane.*

Sparse PCA is “mathematically hopeless" to solve — what about “in practice"?

- Approximation algorithm with multiplicative guarantee: [S. On Chan, D. Papailliopoulos, A. Rubinstein, 2016], [M. Magdon-Ismail, (2017)] NP-hardness of approximation to within $(1-\varepsilon)$, for some small constant $\varepsilon > 0$.

Sparse PCA is “mathematically hopeless" to solve — what about “in practice"?

- Approximation algorithm with multiplicative guarantee: [S. On Chan, D. Papailliopoulos, A. Rubinstein, 2016], [M. Magdon-Ismail, (2017)] NP-hardness of approximation to within $(1-\varepsilon)$, for some small constant $\varepsilon > 0$. The best approximation guarantee of $n^{\frac{1}{3}}$.

Sparse PCA is “mathematically hopeless" to solve — what about “in practice"?

- Approximation algorithm with multiplicative guarantee: [S. On Chan, D. Papailliopoulos, A. Rubinstein, 2016], [M. Magdon-Ismail, (2017)] NP-hardness of approximation to within $(1-\varepsilon)$, for some small constant $\varepsilon > 0$. The best approximation guarantee of $n^{\frac{1}{3}}$.
- Fixed parameter tractable: [D. Papailiopoulos, A. G. Dimakis, S. Korokythakis (2014)] [A. Del Pia (2019)] Poly-time algorithm under fixed rank of matrix.

Sparse PCA is “mathematically hopeless" to solve — what about “in practice"?

- Approximation algorithm with multiplicative guarantee: [S. On Chan, D. Papailliopoulos, A. Rubinstein, 2016], [M. Magdon-Ismail, (2017)] NP-hardness of approximation to within $(1-\varepsilon)$, for some small constant $\varepsilon > 0$. The best approximation guarantee of $n^{\frac{1}{3}}$.
- Fixed parameter tractable: [D. Papailiopoulos, A. G. Dimakis, S. Korokythakis (2014)] [A. Del Pia (2019)] Poly-time algorithm under fixed rank of matrix.
- Exact/near exact mathematical programming methods: [SSD., Rahul Mazumder, Wang (2018)], [Yongchun Li, Weijun Xie (2020)]. Can scale up to matrices of size 1000×1000 in a few hours.

Sparse PCA is “mathematically hopeless" to solve — what about “in practice"?

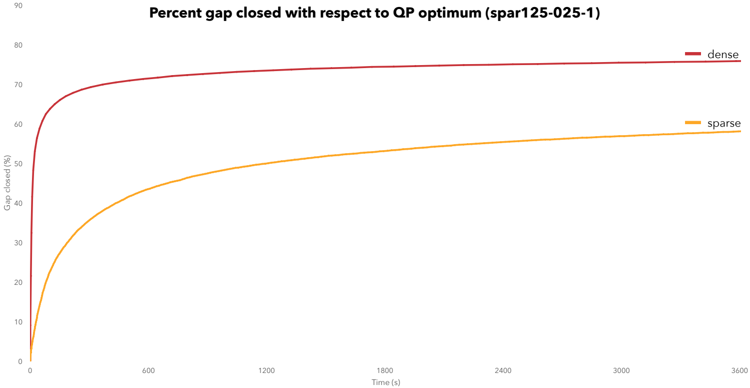
- Approximation algorithm with multiplicative guarantee: [S. On Chan, D. Papailliopoulos, A. Rubinstein, 2016], [M. Magdon-Ismail, (2017)] NP-hardness of approximation to within $(1-\varepsilon)$, for some small constant $\varepsilon > 0$. The best approximation guarantee of $n^{\frac{1}{3}}$.
- Fixed parameter tractable: [D. Papailiopoulos, A. G. Dimakis, S. Korokythakis (2014)] [A. Del Pia (2019)] Poly-time algorithm under fixed rank of matrix.
- Exact/near exact mathematical programming methods: [SSD., Rahul Mazumder, Wang (2018)], [Yongchun Li, Weijun Xie (2020)]. Can scale up to matrices of size 1000×1000 in a few hours.

ML community has come up with some fantastic heuristics:

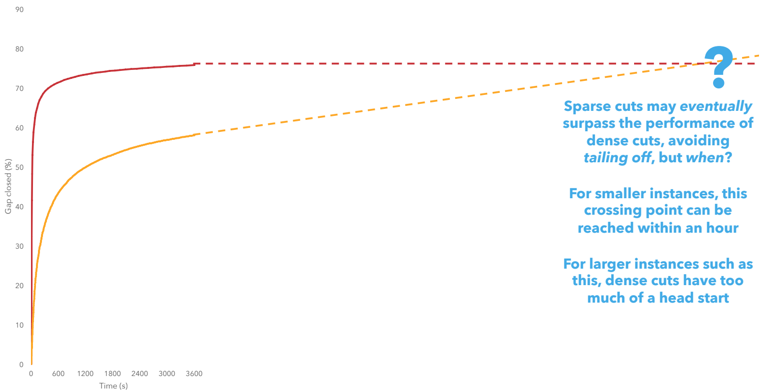
- [X.-T. Yuan and T. Zhang (2013)] “Truncated power method"
- [M. Journee, Y. Nesterov, P. Richtarik, and R. Sepulchre, (2010)]

3 Experiments

Back to experiemental results with $k = 0.25n$, One cut per iteration



Back to experiemental results with $k = 0.25n$, One cut per iteration



Need to add multiple cuts in each iteration

After trying out various ideas, we settled on the following scheme:

- We add multiple cuts in each iteration generated as follows:

Need to add multiple cuts in each iteration

After trying out various ideas, we settled on the following scheme:

- We add multiple cuts in each iteration generated as follows:
  - $\tilde{X}$, we find v^0 such that $(v^0)^\top \tilde{X} v^0 = \lambda^0 < 0$.

Need to add multiple cuts in each iteration

After trying out various ideas, we settled on the following scheme:

- We add multiple cuts in each iteration generated as follows:
  - $\tilde{X}$, we find v^0 such that $(v^0)^\top \tilde{X} v^0 = \lambda^0 < 0$.
  - Update

$$\tilde{X}^1 := \tilde{X} - \lambda^0 (v^0)^\top v^0,$$

Need to add multiple cuts in each iteration

After trying out various ideas, we settled on the following scheme:

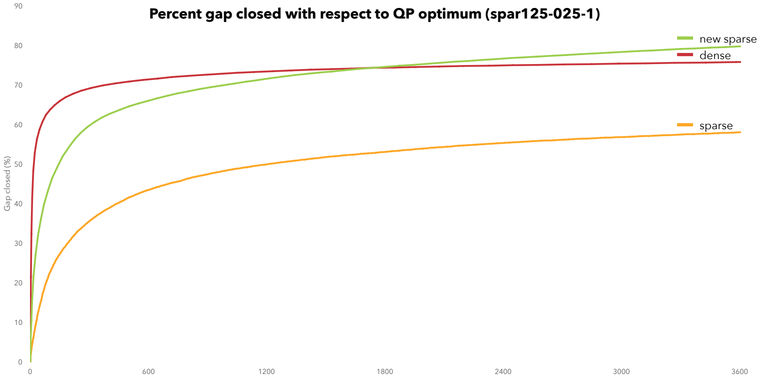
- We add multiple cuts in each iteration generated as follows:
  - $\tilde{X}$, we find v^0 such that $(v^0)^\top \tilde{X} v^0 = \lambda^0 < 0$.
  - Update
    $$\tilde{X}^1 := \tilde{X} - \lambda^0 (v^0)^\top v^0,$$
    and we find v^1 such that $(v^1)^\top \tilde{X}^1 v^1 = \lambda^1 < 0$.
  - Repeat...

Need to add multiple cuts in each iteration

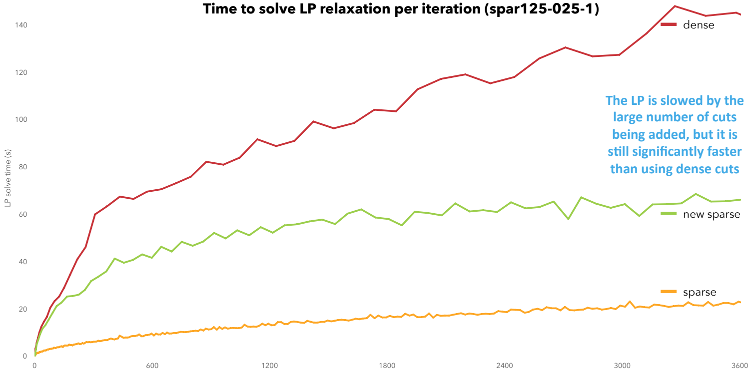
After trying out various ideas, we settled on the following scheme:

- We add multiple cuts in each iteration generated as follows:
  - $\tilde{X}$, we find v^0 such that $(v^0)^\top \tilde{X} v^0 = \lambda^0 < 0$.
  - Update
    $$\tilde{X}^1 := \tilde{X} - \lambda^0 (v^0)^\top v^0,$$
    and we find v^1 such that $(v^1)^\top \tilde{X}^1 v^1 = \lambda^1 < 0$.
  - Repeat...
  - $v^0, v^1, \ldots$ are used to generate cuts.

More experiments (multiple cuts per iteration):



More experiments (multiple cuts per iteration):



Strength of sparse cuts

Theorem (G. Blekherman, SSD., M. Molinaro, K. Shu, S. Sun (2020))

$$\left.\begin{array}{ll} \max & \mathrm{dist}_F(X, S^n_+) \\ \text{s.t.} & \|X\|_F \leq 1 \\ & \text{every } k \times k \text{ principal submatrix of } X \text{ is PSD.} \end{array}\right\} \leq \max\left\{\frac{1-k}{n}, \frac{(n-k)^{3/2}}{\sqrt{(n-k)^2+(n-1)k^2}}\right\}$$

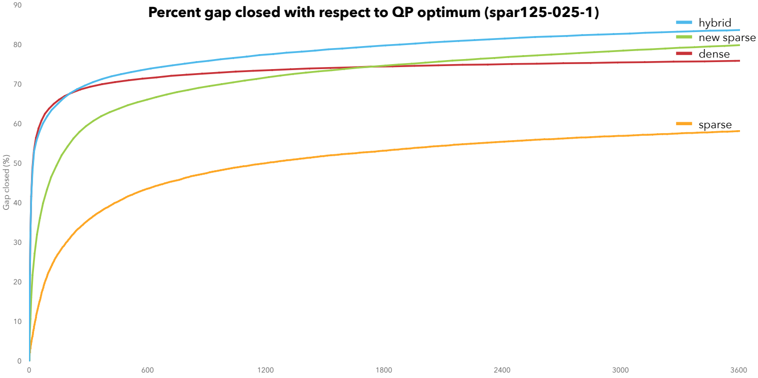
Strength of sparse cuts

Theorem (G. Blekherman, SSD., M. Molinaro, K. Shu, S. Sun (2020))

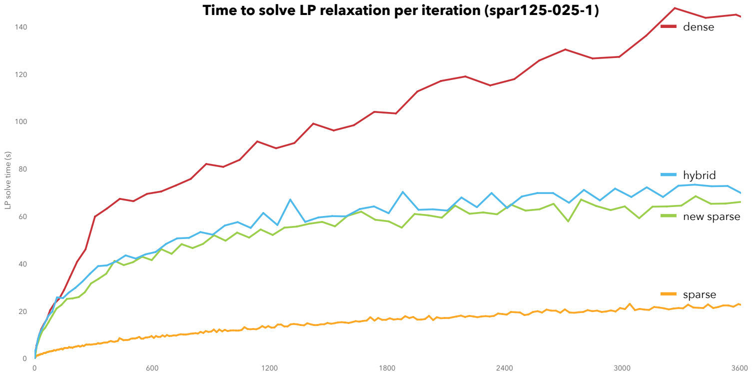
$$\left.\begin{array}{ll} \max & \mathrm{dist}_F(X, S^n_+) \\ \text{s.t.} & \|X\|_F \leq 1 \\ & \text{every } k \times k \\ & \text{ principal submatrix} \\ & \text{ of } X \text{ is PSD.} \end{array}\right\} \leq \max\left\{\frac{1-k}{n}, \frac{(n-k)^{3/2}}{\sqrt{(n-k)^2 + (n-1)k^2}}\right\}$$

- Lets add some dense cuts in the first iteration.

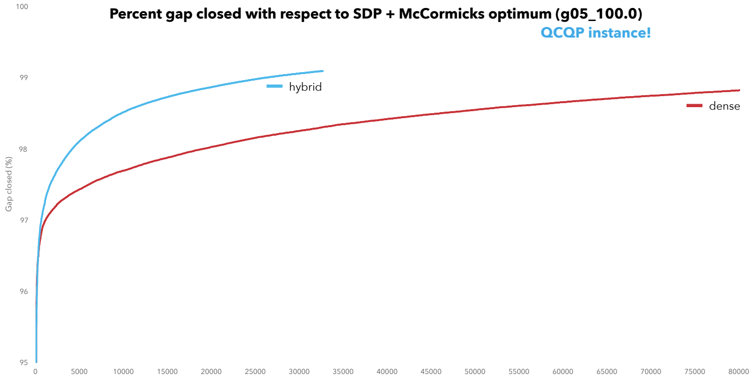
More experiments (multiple cuts per iteration, dense cuts in first iteration):



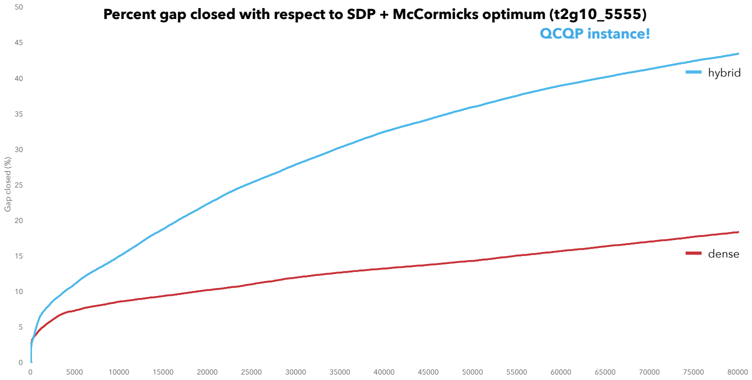
More experiments (multiple cuts per iteration, dense cuts in first iteration):



More experiments (multiple cuts per iteration, dense cuts in first iteration):



More experiments (multiple cuts per iteration, dense cuts in first iteration):



Results

Results on 111 BoxQP instances for SPARSE, DENSE, and HYBRID. Results are averages over instances grouped by size, under a time limit of 1 hour.

		Gap closed (%)			Last LP time (s)		
Instance group	#	SPARSE	DENSE	HYBRID	SPARSE	DENSE	HYBRID
$n \in [20, 30]$	18	98.50	100.00	100.00	0.10	2.93	2.93
$n \in [40, 50]$	33	98.83	99.90	99.89	0.64	10.73	7.34
$n \in [60, 80]$	21	98.45	96.24	98.17	6.49	28.27	11.69
$n \in [90, 125]$	27	94.62	90.68	95.48	48.09	106.54	49.08
$n \in [200, 250]$	12	75.16	84.70	83.92	520.24	764.30	506.98

Commercial SDP solver (Mosek) for $n \in [200, 250]$ needs approximately 35 GB memory, needs > 1 hour to solve.

Results

Results on 135 BIQ instances for SPARSE, DENSE, and HYBRID. Results are averages over instances grouped by size, under a time limit of 1 hour.

		Gap closed (%)			Last LP time (s)		
Instance group	#	SPARSE	DENSE	HYBRID	SPARSE	DENSE	HYBRID
$n \in [20, 90]$	18	98.70	99.47	99.81	1.33	15.14	7.26
$n = 100$	31	94.92	82.53	95.00	31.33	91.35	34.82
$n \in [120, 150]$	41	90.18	89.35	92.61	125.87	262.56	132.43
$n \in [200, 250]$	45	54.72	65.72	64.06	479.61	830.75	519.96

Results

Results on 151 MaxCut instances for SPARSE, DENSE, and HYBRID. Results are averages over instances grouped by size, under a time limit of 1 hour.

		Gap closed (%)			Last LP time (s)		
Instance group	#	SPARSE	DENSE	HYBRID	SPARSE	DENSE	HYBRID
$n = 60$	10	97.45	98.73	98.86	3.20	13.69	10.98
$n = 80$	30	93.61	93.43	96.65	18.59	47.48	24.07
$n = 100$	99	79.36	77.44	82.66	60.09	107.76	86.74
$n \in [150, 225]$	12	6.00	5.13	5.85	717.56	775.20	704.32

Thank You.

- Santanu S. Dey, Aleksandr M. Kazachkov, Andrea Lodi, Gonzalo Muñoz, "Cutting Plane Generation Through Sparse Principal Component Analysis" http://www.optimization-online.org/DBHTML/2021/02/8259.html
- Grigoriy Blekherman, Santanu S. Dey, Marco Molinaro, Shengding Sun, "Sparse PSD approximation of the PSD cone," *Mathematical Programming*, 2020.
- Grigoriy Blekherman, Santanu S. Dey, Kevin Shu, Shengding Sun, "Hyperbolic relaxation of k-Locally positive semidefinite matrices" arXiv:2012.04031.