New Developments of Computational Algorithms for Convex and Nonconvex Optimization

S³ Optimization Day 2022

July 7, 2022

Yinyu Ye
Stanford University
(Currently Visiting NUS)
A group for open source optimization solver development:

faculty, scientists and students from Shufe and Cardinal Operations
Today’s Talk

• New developments of ADMM-based interior point (ABIP) Method

• HDSDP: Homogeneous Dual-Scaling SDP solver
  (W. Gao, D. Ge and Y. Y)

• A Dimension Reduced Second-Order Method
  (C. Zhang, D. Ge and Y. Y)
ABIP (Lin, Ma, Zhang and Y, 2021)

- An ADMM based interior point method solver for LP problems

- The primal-dual pair of LP:

  \[
  \begin{align*}
  \text{(P)} & \quad \min \quad & c^T x \\
  \text{s.t.} & \quad & Ax = b \\
  & \quad & x \geq 0 \\
  \text{(D)} & \quad \max \quad & b^T y \\
  \text{s.t.} & \quad & A^T y + s = c \\
  & \quad & s \geq 0
  \end{align*}
  \]

- For IPM, initial feasible interior solutions are hard to find

- Consider homogeneous and self-dual (HSD) LP here!

  \[
  \begin{align*}
  & \min \quad \beta(n + 1) \theta + 1 (r = 0) + 1 (\xi = -n - 1) \\
  \text{s.t.} & \quad \begin{bmatrix} Q & 0 \\ -A^T & 0 \end{bmatrix} \begin{bmatrix} u \\ x \end{bmatrix} = \begin{bmatrix} v \\ \tau \end{bmatrix} \\
  & \quad y \text{ free}, x \geq 0, \tau \geq 0, \theta \text{ free}, s \geq 0, \kappa \geq 0
  \end{align*}
  \]

where

\[
Q = \begin{bmatrix}
0 & A & -b & \bar{b} \\
-A^T & 0 & c & \bar{c} \\
b^T & -c^T & 0 & \bar{z} \\
-\bar{b}^T & \bar{c}^T & -\bar{z} & 0
\end{bmatrix}, \quad
u = \begin{bmatrix} y \\ x \\ \tau \end{bmatrix}, \quad
v = \begin{bmatrix} r \\ s \\ \kappa \end{bmatrix}, \quad
\bar{b} = b - Ae, \quad \bar{c} = c - e, \quad \bar{z} = c^T e + 1
ABIP – Subproblem

• Introduce log-barrier penalty for HSD LP

\[
\begin{align*}
\min \quad & B(u, v, \mu) \\
\text{s.t.} \quad & Qu = v
\end{align*}
\]

where \( B(u, v, \mu) \) barrier function

• Traditional IPM, one uses Newton’s method to solve the KKT system of the above problem, the cost is too expensive when problem is large!

• Now we apply ADMM to solve it inexactly

\[
\begin{align*}
\min \quad & 1(Q\tilde{u} = \tilde{v}) + B(u, v, \mu^k) \\
\text{s.t.} \quad & (\tilde{u}, \tilde{v}) = (u, v)
\end{align*}
\]

The augmented Lagrangian function

\[
L_\beta(\tilde{u}, \tilde{v}, u, v, \mu^k, p, q) := 1(Q\tilde{u} = \tilde{v}) + B(u, v, \mu^k) - \langle \beta(p, q), (\tilde{u}, \tilde{v}) - (u, v) \rangle + \frac{\beta}{2} \| (\tilde{u}, \tilde{v}) - (u, v) \|^2
\]
ABIP – Rescale

Consider the following LP:

\[
\begin{align*}
\min & \quad c^T x \\
\text{s.t.} & \quad Ax = b \\
& \quad x \geq 0
\end{align*}
\]

Rescale the constraint matrix \( A \) to \( \tilde{A} = D_1^{-1}AD_2^{-1} \) with positive diagonal matrices \( D_1 \) and \( D_2 \) to decrease the condition number of \( A \).

- Ruiz Rescaling: take
  \[
  (D_1)_{jj} = \sqrt{\|A_{j,.}\|_{\infty}} \quad \text{and} \quad (D_2)_{ii} = \sqrt{\|A_{.,i}\|_{\infty}}
  \]
- Pock-Chambolle Rescaling: take
  \[
  (D_1)_{jj} = \sqrt{\|A_{j,.}\|_{2-\alpha}} \quad \text{and} \quad (D_2)_{ii} = \sqrt{\|A_{.,i}\|_{\alpha}}
  \]

Do Pock-Chambolle Rescaling first (take \( \alpha = 1 \)), then apply Ruiz Rescaling 10 times.
ABIP – Restart (motivated from recent PDLP, Lu at al. and others)

- Idea: Let the uniform average of the past $F$ iterates be the new iterate
- Apply the fixed frequency restart, only in the inner problem of ABIP
- Parameterized by a restart threshold $TH$ and a restart frequency $F$
- Break the inner ADMM Algorithm and restart when

$$M \geq TH \quad \text{and} \quad M_k \mod F = 0$$

where $M$ denotes # total ADMM iterations so far, and $M_k$ represents # ADMM iterations of the inner problem with respect to $\mu_k$
- Significantly reduce # ADMM iterations!
ABIP – Restart

- ABIP tends to induce a spiral trajectory
ABIP – Restart

- After restart, ABIP moves more aggressively and converges faster (reduce almost 70% ADMM iterations)!
ABIP – Inner Loop Convergence Check

- We simultaneously use the last iterate and the ergodic iterate (i.e., the average of history iterates) to check the inner loop convergence.

- We also check the global convergence in the inner loop when the barrier parameter $\mu_k < \epsilon$. 
ABIP – Half-update strategy

• The original update strategy of the ABIP

\[
\begin{align*}
\text{Update } & \tilde{u}_{i+1}^k = (I + Q)^{-1}(u_i^k + v_i^k) ; \\
\text{Update } & u_{i+1}^k ; \\
\text{Update } & v_{i+1}^k = v_i^k - \tilde{u}_{i+1}^k + u_{i+1}^k ;
\end{align*}
\]

• The half update strategy is updating the dual variable \( v \) before updating the variable \( u \), where \( \alpha \) is the step size

\[
\begin{align*}
\text{Update } & \tilde{u}_{i+1}^k = (I + Q)^{-1}(u_i^k + v_i^k) ; \\
\text{Update } & v_{i+\frac{1}{2}}^k = \alpha u_i^k + v_i^k - \alpha \tilde{u}_{i+1}^k ; \\
\text{Update } & u_{i+1}^k ; \\
\text{Update } & v_{i+1}^k = v_{i+\frac{1}{2}}^k - \tilde{u}_{i+1}^k + u_{i+1}^k ;
\end{align*}
\]

• Decrease # ADMM iterations on some specific dataset
ABIP – Adaptive barrier parameter $\mu$ reduction strategy

- Barrier parameter reduction is critical for ABIP
- Balance the progress of ADMM and IPM
- Currently **three** strategies are applied in different phases of ABIP

**LOQP strategy**

$$\xi = \frac{\min\{x, s, k\tau\}}{(x, s + k\tau)/(n+1)} \rightarrow \text{Measure of centrality}$$

$$\mu^{k+1} = \mu^k \cdot \max \left\{ 0.1 \min \left\{ 0.05 \cdot \frac{1-\xi}{\xi}, 2 \right\}^3, \alpha \right\}, \alpha \in (0, 1)$$

More decrease when close to central path

**Aggressive decrease**

$$\mu^{k+1} = \min \{ \zeta \mu^k, (\mu^k)^\eta \}, \zeta \in (0, 1), \eta > 1$$

**Stage-wise decrease**

$$\mu^{k+1} = \sigma(k)\mu^k, \sigma(k) \approx 0.8$$

- Fully adaptive and nice empirical performance in practice
ABIP – Strategy integration

• Different parameters may be significantly different

• An integration strategy based on decision tree is integrated into ABIP

\[
\begin{align*}
\min \quad & c^T x \\
\text{s.t.} \quad & Ax = b \\
\quad & x \geq 0
\end{align*}
\]

• A simple feature-to-strategy mapping is derived from a machine learning model

• For generalization limit the number of strategies (2 or 3 types)
An example of null objective problem—Primal feasibility problem

• Some LPs (e.g., PageRank) aim to find a primal feasible solution

\[
\begin{align*}
\min_x & \quad 0 \\
\text{subject to} & \quad Ax = b \\
& \quad x \geq 0,
\end{align*}
\]

• The dual problem is homogeneous and admits a trivial dual solution \((0, 0)\)

• Approximate dual solution can be scaled down to \((0, 0)\)

• Dual feasibility check is turned off for these problems
**ABIP – Netlib**

- Selected 105 Netlib instances
- $\epsilon = 10^{-6}$, use the direct method, $10^6$ max ADMM iterations

<table>
<thead>
<tr>
<th>Method</th>
<th># Solved</th>
<th># IPM</th>
<th># ADMM</th>
<th>Avg. Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABIP</td>
<td>65</td>
<td>74</td>
<td>265418</td>
<td>87.07</td>
</tr>
<tr>
<td>+ restart</td>
<td>68</td>
<td>74</td>
<td>88257</td>
<td>23.63</td>
</tr>
<tr>
<td>+ rescale</td>
<td>84</td>
<td>72</td>
<td>77925</td>
<td>20.44</td>
</tr>
<tr>
<td>+ hybrid $\mu$ (=ABIP3+)</td>
<td>86</td>
<td>22</td>
<td>73738</td>
<td>14.97</td>
</tr>
</tbody>
</table>

- Hybrid $\mu$: If $\mu > \epsilon$ use the aggressive strategy, otherwise use the LOQO strategy
- ABIP3+ decreases both # IPM iterations and # ADMM iterations significantly
ABIP – MIP2017

• 240 MIP2017 instances

• $\epsilon = 10^{-4}$, presolved by PaPILO, use the direct method, $10^6$ max ADMM iterations

<table>
<thead>
<tr>
<th>Method</th>
<th># Solved</th>
<th>SGM</th>
</tr>
</thead>
<tbody>
<tr>
<td>COPT</td>
<td>240</td>
<td>1</td>
</tr>
<tr>
<td>PDL(P(Julia))</td>
<td>202</td>
<td>17.4</td>
</tr>
<tr>
<td>ABIP</td>
<td>192</td>
<td>34.8</td>
</tr>
<tr>
<td>ABIP3+ Integration</td>
<td>212</td>
<td>16.7</td>
</tr>
</tbody>
</table>

• PDL(P(Lu et al. 2021)) is a practical first-order method (i.e., the primal-dual hybrid gradient (PDHG) method) for linear programming, and it enhances PDHG by a few implementation tricks.
ABIP – PageRank

• 122 instances, generated from sparse matrix datasets: DIMACS10, Gleich, Newman and SNAP. Second order methods in commercial solver fail in most of these instances.

• $\epsilon = 10^{-4}$, use the indirect method, 5000 max ADMM iterations.

<table>
<thead>
<tr>
<th>Method</th>
<th># Solved</th>
<th>SGM</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDL(P(Julia))</td>
<td>122</td>
<td>1</td>
</tr>
<tr>
<td>ABIP3+</td>
<td>119</td>
<td>1.31</td>
</tr>
</tbody>
</table>

• Examples:

<table>
<thead>
<tr>
<th>Instance</th>
<th># nodes</th>
<th>PDL(P(Julia))</th>
<th>ABIP3+</th>
</tr>
</thead>
<tbody>
<tr>
<td>amazon0601</td>
<td>403394</td>
<td>117.54</td>
<td>71.15</td>
</tr>
<tr>
<td>coAuthorsDBLP</td>
<td>299067</td>
<td>51.66</td>
<td>24.70</td>
</tr>
<tr>
<td>web-BerkStan</td>
<td>685230</td>
<td>447.68</td>
<td>139.75</td>
</tr>
<tr>
<td>web-Google</td>
<td>916428</td>
<td>293.01</td>
<td>148.18</td>
</tr>
</tbody>
</table>
ABIP – PageRank

• Generated by Google code

• When # nodes equals to # edges, the generated instance is a staircase matrix. For example,

```
-1.0000  0.1980  0  0  0  0  0  0  0  0
0.9900  -1.0000  0.4950  0.9900  0.4950  0.4950  0  0  0  0
0  0.1980  -1.0000  0  0  0  0.4950  0  0  0
0  0.1980  0  -1.0000  0  0  0  0  0  0
0  0.1980  0  0  -1.0000  0  0  0.9900  0  0
0  0.1980  0  0  0  -1.0000  0  0  0.9900  0
0  0  0  0.4950  0  0  0  -1.0000  0  0
0  0  0  0  0.4950  0  0  -1.0000  0  0
0  0  0  0  0  0.4950  0  0  -1.0000  0
0  0  0  0  0  0  0.4950  0  0  -1.0000
```

Staircase matrix instance (# nodes = 10)

• In this case, ABIP3+ is significantly faster than PDLP!

<table>
<thead>
<tr>
<th># nodes</th>
<th>PDL (Julia)</th>
<th>ABIP3+</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^4$</td>
<td>8.60</td>
<td>0.93</td>
</tr>
<tr>
<td>$10^5$</td>
<td>135.67</td>
<td>10.36</td>
</tr>
<tr>
<td>$10^6$</td>
<td>2248.40</td>
<td>60.32</td>
</tr>
</tbody>
</table>
ABIP – Quadratic Programming

For quadratic programs, our approach is through reformulation of QP as SOCP and then apply ABIP to solve the corresponding conic problem

\[
\begin{align*}
Q_P & : \quad \min \frac{1}{2} x^T Q x + q^T x \\
& \quad \text{s.t. } Ax = b \\
& \quad x \geq 0
\end{align*}
\]

\[
\begin{align*}
SOCP & : \quad \min c^T x \\
& \quad \text{s.t. } Ax = b \\
& \quad x \in K
\end{align*}
\]

where \( K \) is cartesian product of (rotated) second-order cone and positive orthant.

Convex QP is converted into SOCP as follows:

\[
\begin{align*}
& \min \frac{1}{2} x^T Q x + q^T x \\
& \quad \text{s.t. } Ax = b \\
& \quad x \geq 0
\end{align*} \quad \overset{Q = \Lambda^T \Lambda}{\Longrightarrow} \quad \begin{align*}
& \min \frac{1}{2} \bar{x}^T \bar{x} + q^T x \\
& \quad \text{s.t. } Ax = b \\
& \quad \bar{x} = \Lambda x \\
\end{align*} \quad \Longrightarrow \quad \begin{align*}
& \min \nu + q^T x \\
& \quad \text{s.t. } \eta = 1 \\
& \quad Ax = b \\
& \quad \bar{x} = \Lambda x \\
& \quad \eta \nu \geq \frac{1}{2} \bar{x}^T \bar{x} \\
& \quad x \geq 0
\end{align*}
\]
ABIP – Extension to SOCP and Beyond

ABIP iteration remains valid for general conic linear program

\[
\begin{align*}
\min & \quad c^T x \\
\text{s.t.} & \quad Ax = b \\
x & \in \mathcal{K}
\end{align*}
\]

- ABIP-subproblem requires to solve a proximal mapping \( x^+ = \arg\min \lambda F(x) + \frac{1}{2} \| x - c \|^2 \) with respect to the log-barrier functions \( F(x) \) in \( B(u, v, \mu^k) \)

**Positive orthant**
- \( F(x) = -\log(x) \)
- \( x = \arg\min \lambda F(x) + \frac{1}{2} \| x - c \|^2 = \frac{c + \sqrt{c^2 + 4\lambda}}{2} \)

**Second-order cone**
- \( F(x) = -\log(t^2 - \| x \|^2), x = (t; x) \)
- Can be solved by finding the root of quadratic functions

**Positive semidefinite cone**
- \( F(x) = -\log(\det x) \)
- Equivalent to solve \(-\lambda x^{-1} - c + x = 0\)
- Can be solved by eigen decomposition

- The total IPM and ADMM iteration complexities of ABIP for conic linear program are respectively:

\[
T_{\text{IPM}} = O\left( \log \left( \frac{1}{\epsilon} \right) \right), \quad T_{\text{ADMM}} = O\left( \frac{1}{\epsilon} \log \left( \frac{1}{\epsilon} \right) \right)
\]
ABIP – Customization for ML

ABIP solves linear system:

\[
\begin{pmatrix}
I_m & A \\
A^T & -I_n
\end{pmatrix}
\begin{pmatrix}
\hat{u}_1 \\
\hat{u}_2
\end{pmatrix} =
\begin{pmatrix}
\hat{\omega}_1 \\
-\hat{\omega}_2
\end{pmatrix}
\]

- If \( A \) is a general sparse matrix, we prefer augmented system, which is solved by sparse LDL decomposition

\[
\begin{pmatrix}
I_m + AA^T & -I_n \\
A^T & -I_n
\end{pmatrix}
\begin{pmatrix}
\hat{u}_1 \\
\hat{u}_2
\end{pmatrix} =
\begin{pmatrix}
\hat{\omega}_1 - A\hat{\omega}_2 \\
-\hat{\omega}_2
\end{pmatrix}
\]

- If \( A \) is dense or it has highly different row and col dimensionalities, we prefer normal equation

For many QP problems in machine learning, we provide customized linear system solver by applying Sherman-Morrison-Woodbury formula and simplifying the normal equation

**LASSO**
- Data matrix \( \tilde{A} \) of LASSO has \( n \) features, \( m \) samples
- The dimension of factorized matrix reduced from \( 2m + 2n + 3 \) to \( \min\{m, n\} \)

**SVM**
- Data matrix \( \tilde{A} \) of SVM has \( n \) features, \( m \) samples
- The dimension of factorized matrix reduced from \( 3m + 4n + 5 \) to \( n + 1 \)
ABIP – LASSO

• Randomly generate 48 instances with \( m \in [2000,10000], n \in [50,20000] \)

• \( \epsilon = 10^{-3} \), time limit = 2000s

<table>
<thead>
<tr>
<th></th>
<th>ABIP</th>
<th>OSQP</th>
<th>SCS</th>
<th>GUROBI</th>
</tr>
</thead>
<tbody>
<tr>
<td>solved</td>
<td>48</td>
<td>47</td>
<td>48</td>
<td>46</td>
</tr>
<tr>
<td>1st</td>
<td>6</td>
<td>24</td>
<td>0</td>
<td>18</td>
</tr>
<tr>
<td>2nd</td>
<td>25</td>
<td>1</td>
<td>18</td>
<td>4</td>
</tr>
<tr>
<td>3rd</td>
<td>17</td>
<td>15</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>4th</td>
<td>0</td>
<td>7</td>
<td>15</td>
<td>24</td>
</tr>
</tbody>
</table>
### ABIP – SVM

- For 6 large instances from LIBSVM, $\epsilon = 10^{-3}$, time limit = 2000s

<table>
<thead>
<tr>
<th>data</th>
<th>covtype</th>
<th>icnn1</th>
<th>news20</th>
<th>rcv1_train</th>
<th>real_sim</th>
<th>skin_orig_half</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>581012</td>
<td>49990</td>
<td>19996</td>
<td>20242</td>
<td>72309</td>
<td>122540</td>
</tr>
<tr>
<td>n</td>
<td>54</td>
<td>22</td>
<td>1355191</td>
<td>44504</td>
<td>20958</td>
<td>3</td>
</tr>
<tr>
<td>abip_time</td>
<td>158.76</td>
<td>2.58</td>
<td>1372.57</td>
<td>98.16</td>
<td>478.81</td>
<td>6.13</td>
</tr>
<tr>
<td>abip_iter</td>
<td>649</td>
<td>138</td>
<td>930</td>
<td>789</td>
<td>762</td>
<td>171</td>
</tr>
<tr>
<td>osqp_time</td>
<td>82.42</td>
<td>2.87</td>
<td>inf</td>
<td>inf</td>
<td>inf</td>
<td>0.45</td>
</tr>
<tr>
<td>osqp_iter</td>
<td>850</td>
<td>325</td>
<td>inf</td>
<td>inf</td>
<td>inf</td>
<td>25</td>
</tr>
<tr>
<td>racqp_time</td>
<td>221.25</td>
<td>6.00</td>
<td>531.67</td>
<td>66.30</td>
<td>74.23</td>
<td>4.93</td>
</tr>
<tr>
<td>racqp_iter</td>
<td>396</td>
<td>155</td>
<td>70</td>
<td>49</td>
<td>136</td>
<td>107</td>
</tr>
<tr>
<td>scs_time</td>
<td>122.38</td>
<td>4.45</td>
<td>inf</td>
<td>765.32</td>
<td>inf</td>
<td>47.80</td>
</tr>
<tr>
<td>scs_iter</td>
<td>950</td>
<td>400</td>
<td>inf</td>
<td>100</td>
<td>inf</td>
<td>2800</td>
</tr>
<tr>
<td>gurobi_time</td>
<td>1612.15</td>
<td>14.82</td>
<td>286.46</td>
<td>115.25</td>
<td>Inf</td>
<td>22.19</td>
</tr>
<tr>
<td>gurobi_iter</td>
<td>153</td>
<td>58</td>
<td>18</td>
<td>15</td>
<td>Inf</td>
<td>17</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>ABIP</th>
<th>OSQP</th>
<th>RACQP</th>
<th>SCS</th>
<th>GUROBI</th>
</tr>
</thead>
<tbody>
<tr>
<td>solved</td>
<td>6</td>
<td>3</td>
<td>6</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>1st</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2nd</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3rd</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4th</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5th</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>
ABIP – Conic Quadratic Programming

• For general QP without any structure information, SOCP reformulation may be inefficient.

• We consider the following conic quadratic programming (CQP) and its dual:

\[
(P) \quad \begin{align*}
\min & \quad \frac{1}{2} x^T Q x + c^T x \\
\text{s.t.} & \quad A x = b \\
& \quad x \in \mathcal{K}
\end{align*}
\]

\[
(D) \quad \begin{align*}
\max & \quad -\frac{1}{2} x^T Q x + b^T y \\
\text{s.t.} & \quad Q x - A^T y + c - s = 0 \\
& \quad x \in \mathcal{K}, \ s \in \mathcal{K}^*
\end{align*}
\]

The KKT condition is

\[
\begin{align*}
& A x = b \\
& Q x - A^T y + c - s = 0 \\
& x \in \mathcal{K} \\
& s \in \mathcal{K}^* \\
& s \perp x
\end{align*}
\]

which corresponds to the linear complementarity problem \( LCP(M, q, C) \) in variable \( z \) with:

\[
z = \begin{bmatrix} y \\ x \end{bmatrix}, \quad M = \begin{bmatrix} 0 & A \\ -A^T & Q \end{bmatrix}, \quad q = \begin{bmatrix} -b \\ c \end{bmatrix}, \quad C = \mathbb{R}^m \times \mathcal{K}
\]

\( (P) \) is infeasible when we find \( y \in \mathbb{R}^m \) that satisfies

\[-A^T y \in \mathcal{K}^*, \ b^T y > 0\]

\( (D) \) is infeasible when we find \( x \in \mathbb{R}^n \) that satisfies

\[Q x = 0, A x = 0, x \in \mathcal{K}, c^T x < 0\]
ABIP – Conic Quadratic Programming

Homogeneous Embedding (Andersen and Y 1999)

• LCP feasibility encoded by homogeneous operator \( F(z, \tau) = \begin{bmatrix} Mz + q \tau \\ -z^T Mz - z^T q \end{bmatrix} \)

• LCP infeasibility encoded by homogeneous operator \( I(z, \tau) = \begin{bmatrix} [Mz] \mid \kappa \leq -z^T q \end{bmatrix} \), \( \text{dom}(I) = \{(z, 0) : z^T Mz = 0\} \)

• The final embedding corresponds to solving a monotone complementarity problem \( MCP(Q, C_+) \):

\[
\text{find a } u \in \mathbb{R}^{m+n+1} \text{ for which } \exists v \in Q(u) \text{ s.t. } C_+ \ni u \perp v \in C^*_+
\]

where \( Q = F \cup I \), \( \text{dom}(Q) = \text{dom}(F) \cup \text{dom}(I) \) and \( C_+ = C \times \mathbb{R}^+ \).

• For solving such MCP, ABIP performs the update strategy similar to ABIP-LP

\[
\tilde{u}_{k+1} = (I + Q)^{-1}(u_k + v_k)
\]
\[
u_{k+1} = \text{prox}_B(\tilde{u}_{k+1} - v_k)
\]
\[
u_{k+1} = v_k + u_{k+1} - \tilde{u}_{k+1}
\]
ABIP – General Convex QP

- Maros-Meszaros convex QP dataset contains 138 instances

<table>
<thead>
<tr>
<th></th>
<th>ABIP-SOCP</th>
<th>ABIP-QCP</th>
<th>SCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>solved</td>
<td>110</td>
<td>124</td>
<td>130</td>
</tr>
</tbody>
</table>

- ABIP obtains significant improvement on convex QP

<table>
<thead>
<tr>
<th>eps</th>
<th>ABIP-QCP</th>
<th>SCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1e-4</td>
<td>114</td>
<td>125</td>
</tr>
<tr>
<td>1e-5</td>
<td>112</td>
<td>121</td>
</tr>
</tbody>
</table>
Today’s Talk

• New developments of ADMM based interior point (ABIP) Method

• HDSDP: Homogeneous dual-scaling SDP solver
  (W. Gao, D. Ge and Y. Ye)

• A Dimension Reduced Second-Order Method
  (C. Zhang, D. Ge and Y. Ye)
HDSDP: Homogeneous Dual-Scaling SDP solver

\[
\begin{align*}
\min_X & \quad \langle C, X \rangle \\
\text{subject to} & \quad AX = b \\
& \quad X \in \mathbb{S}^n_+ \\
AX - b\tau & = 0 \\
-A^*y + CT - S & = 0 \\
b^Ty - \langle C, X \rangle - \kappa & = 0 \\
X, S \succeq 0, & \quad \kappa, \tau \geq 0
\end{align*}
\]
Semi-definite programming (SDP)

SDP is a mathematical programming problem taking form of

<table>
<thead>
<tr>
<th>Primal</th>
<th>Dual</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \min_X \langle C, X \rangle )</td>
<td>( \max_y b^\top y )</td>
</tr>
<tr>
<td>subject to ( AX = b )</td>
<td>subject to ( A^\top y + S = C )</td>
</tr>
<tr>
<td>( X \in \mathbb{S}_+^n )</td>
<td>( S \in \mathbb{S}_+^n )</td>
</tr>
</tbody>
</table>

- Linear optimization over the cone of positive semi-definite matrices
- Many applications in real practice
SDP and its applications

SDP has been widely used in

- Dynamic system and numerical analysis
  Lyapunov equation and Linear matrix inequalities

- Combinatorial optimization and relaxation
  Well-known 0.878 algorithm for Max-Cut

- Graph realization and distance geometry
  Sensor network localization and Biswas-Ye algorithm

- Engineering and structure design
  Truss design optimization

SDP is popular for

powerful modeling ability + efficient numerical algorithms
Interior point method for SDPs

SDP is solvable in polynomial time using the interior point methods

• Take Newton step towards the perturbed KKT system

\[
\begin{align*}
\mathcal{A}X &= b \\
\mathcal{A}^*y + S &= C \\
XS &= 0
\end{align*}
\]

\[
\begin{align*}
\mathcal{A}X &= b \\
\mathcal{A}^*y + S &= C \\
XS &= \mu I
\end{align*}
\]

\[
\begin{align*}
\mathcal{A}\Delta X &= -R_P \\
\mathcal{A}^*\Delta y + \Delta S &= -R_D \\
H_P(X\Delta S + \Delta XS) &= -R_\mu
\end{align*}
\]

• Efficient numerical solvers have been developed

COPT, Mosek, SDPT3, SDPA, DSDP...

• Most IPM solvers adopt primal-dual path-following IPMs except DSDP

DSDP (Dual-scaling SDP) implements a dual potential reduction method
Dual-scaling and DSDP

DSDP(5.8) implements a dual-scaling interior point method that

\[
\begin{align*}
\mathcal{A}X &= b \\
\mathcal{A}^*y + S &= C \\
XS &= \mu I
\end{align*}
\]

• take Newton step towards \( X = \mu S^{-1} \). An easy inversion trick.

• only iterates in the dual space \((y, S)\)

• guides the iterations and barrier \( \mu \) reduction/update via dual potential function (a delicate algorithm)

• proves a quite efficient and elegant implementation

• requires initial dual feasible solution from big-M method

But big-M is sometimes not stable. Can we improve? Yes! We can embed it.
Simplified homogeneous self-dual model (HSD)

HSD is a skew-symmetric system that

• embeds the primal-dual information

• introduces homogenizing variables $\kappa, \tau$ to detect infeasibility

• proves numerically stable

• solved by infeasible primal-dual path-following

Linearize $XS = \mu I, \kappa \tau = \mu$ and take Newton steps

How to integrate dual-scaling and HSD?

**Hint.**

$AX = b \quad AX = b$

$A^*y + S = C \quad A^*y + S = C$

$XS = \mu I \quad X = \mu S^{-1}$

$AX - b\tau = 0$

$-A^*y + C\tau - S = 0$

$b^\top y - \langle C, X \rangle - \kappa = 0$

$X, S \succeq 0, \quad \kappa, \tau \geq 0$

$AX - b\tau = 0$

$-A^*y + C\tau - S = 0$

$b^\top y - \langle C, X \rangle - \kappa = 0$

$X = \mu S^{-1}$

$\kappa \tau = \mu$

(or $\kappa = \mu \tau^{-1}$)

Apply the inversion trick!
Homogeneous dual-scaling algorithm

From arbitrary starting dual solution \((y, S > 0, \tau > 0)\) with dual residual \(R\)

\[
\begin{align*}
\mathcal{A}X - b\tau &= 0 \\
-\mathcal{A}^*y + C\tau - S &= 0 \\
b^\top y - \langle C, X \rangle - \kappa &= 0 \\
X = \mu S^{-1} & \quad \kappa = \mu \tau^{-1}
\end{align*}
\]

\[
\begin{align*}
\mathcal{A}(X + \Delta X) - b(\tau + \Delta \tau) &= 0 \\
-\mathcal{A}^*(y + \Delta y) + C(\tau + \Delta \tau) - (S + \Delta S) &= 0 \\
\mu S^{-1} \Delta SS^{-1} + \Delta X &= \mu S^{-1} - X \\
\mu \tau^{-2} \Delta \tau + \Delta \kappa &= \mu \tau^{-1} - \kappa
\end{align*}
\]

- Primal iterations can still be fully eliminated

- \(S = -\mathcal{A}^*y + C\tau - R\) inherits sparsity pattern of data
  
  Less memory and since \(X\) is generally dense

New strategies are tailored for the method

- Infeasibility or an early feasible solution can be detected via the embedding
Computational aspects for HDSDP

To enhance performance, HDSDP is equipped with

• Pre-solving that detects special structure and dependency

• Line-searches over barrier to balance optimality & centrality

• Heuristics to update the barrier parameter $\mu$

• Corrector strategy to reuse the Schur matrix

• A complete dual-scaling algorithm from DSDP5.8

• More delicate strategies for the Schur system
Computational aspects: Schur complement

In HDSDP, main computation comes from the large Schur complement system

$$M = \begin{pmatrix}
\langle A_1, S^{-1}A_1S^{-1} \rangle & \cdots & \langle A_1, S^{-1}A_mS^{-1} \rangle \\
\vdots & \ddots & \vdots \\
\langle A_m, S^{-1}A_1S^{-1} \rangle & \cdots & \langle A_m, S^{-1}A_mS^{-1} \rangle
\end{pmatrix}$$

- A generally dense $m \times m$ system
- Setting it up dominates computation
- Most interior point solvers implement special tricks for it
  - Exploit sparsity (SDPA) or low-rank structure (DSDP)
Computational aspects: Schur complement

HDSDP optionally obtains the Eigen-decomposition of data: \[ A_i = \sum_{r=1}^{\text{rank}(A_i)} \lambda_i r a_i^T r \]

To exploit low-rank structure (DSDP)

<table>
<thead>
<tr>
<th>Technique</th>
<th>Approximate flops for Schur</th>
<th>Flops for ( S^{-1} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>( r_{\sigma(i)}(n^2 + 2n^2) + \kappa \sum_{j \geq i} f_{\sigma(j)} )</td>
<td>0</td>
</tr>
<tr>
<td>M2</td>
<td>( r_{\sigma(i)}(n^2 + \kappa \sum_{j \geq i} f_{\sigma(j)} )</td>
<td>0</td>
</tr>
<tr>
<td>M3</td>
<td>( n\kappa f_{\sigma(i)} + n^3 + \sum_{j \geq i} \kappa f_{\sigma(j)} )</td>
<td>( O(n^3) )</td>
</tr>
<tr>
<td>M4</td>
<td>( n\kappa f_{\sigma(i)} + \sum_{j \geq i} \kappa(n+1)f_{\sigma(j)} )</td>
<td>( O(n^3) )</td>
</tr>
<tr>
<td>M5</td>
<td>( \kappa(2\kappa f_{\sigma(i)} + 1) \sum_{j \geq i} f_{\sigma(j)} )</td>
<td>( O(n^3) )</td>
</tr>
</tbody>
</table>

To exploit sparsity (SDPA)

1. Compute \( M_{\sigma(i)\sigma(j)} = \sum_{r=1}^{\text{rank}(A_{\sigma(i)})} (S^{-1}a_{\sigma(i),r})^T A_{\sigma(j)}(S^{-1}a_{\sigma(i),r}) \)

2. Compute \( M_{\sigma(i)\sigma(j)} = \langle B_{\sigma(i)}S^{-1}, A_{\sigma(j)} \rangle, \forall j \geq i \).

Technique M3

\[
\begin{pmatrix}
\langle S^{-1}A_{\sigma(1)}S^{-1}, A_{\sigma(1)} \rangle \\
\vdots \\
\langle S^{-1}A_{\sigma(m)}S^{-1}, A_{\sigma(m)} \rangle
\end{pmatrix} \quad \rightarrow \quad \begin{pmatrix}
\langle S^{-1}A_{\sigma(1)}S^{-1}, A_{\sigma(1)} \rangle \\
\vdots \\
\langle S^{-1}A_{\sigma(m)}S^{-1}, A_{\sigma(m)} \rangle
\end{pmatrix}
\]

• All the five strategies are implemented in HDSDP
• Each row of the Schur complement is set up by the cheapest technique in flops
• A permutation of the Schur matrix might be generated to minimize the flops
HDSDP Solver

HDSDP is written in ANSI C and now under active development

• Compatible with state-of-the-art linear system solvers (e.g. Intel MKL and Pardiso)
• A complete solver interface and supports SDPA reading
• Automatically switches between HSD and original DSDP
Computational results

- HDSDP is tuned and tested for many benchmark datasets
- Good performance on problems with both low-rank structure and sparsity
- Solve around 70/75 Mittelmann’s benchmark problems
- Solve 90/92 SDPLIB problems

<table>
<thead>
<tr>
<th>Instance</th>
<th>DSDP5.8</th>
<th>HDSDP</th>
<th>Mosek v9</th>
<th>SDPT3</th>
<th>COPT v5</th>
</tr>
</thead>
<tbody>
<tr>
<td>G40_mb</td>
<td>18</td>
<td>7</td>
<td>174</td>
<td>25</td>
<td>18</td>
</tr>
<tr>
<td>G48_mb</td>
<td>36</td>
<td>8</td>
<td>191</td>
<td>49</td>
<td>35</td>
</tr>
<tr>
<td>G48mc</td>
<td>11</td>
<td>2</td>
<td>71</td>
<td>24</td>
<td>18</td>
</tr>
<tr>
<td>G55mc</td>
<td>200</td>
<td>179</td>
<td>679</td>
<td>191</td>
<td>301</td>
</tr>
<tr>
<td>G59mc</td>
<td>347</td>
<td>246</td>
<td>646</td>
<td>256</td>
<td>442</td>
</tr>
<tr>
<td>G60_mb</td>
<td>700</td>
<td>213</td>
<td>7979</td>
<td>592</td>
<td>714</td>
</tr>
<tr>
<td>G60mc</td>
<td>712</td>
<td>212</td>
<td>8005</td>
<td>590</td>
<td>713</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instance</th>
<th>DSDP5.8</th>
<th>HDSDP</th>
<th>Mosek v9</th>
<th>SDPT3</th>
<th>COPT v5</th>
</tr>
</thead>
<tbody>
<tr>
<td>checker1.5</td>
<td>87</td>
<td>41</td>
<td>72</td>
<td>71</td>
<td>81</td>
</tr>
<tr>
<td>foot</td>
<td>28</td>
<td>14</td>
<td>533</td>
<td>32</td>
<td>234</td>
</tr>
<tr>
<td>hand</td>
<td>4</td>
<td>2</td>
<td>76</td>
<td>8</td>
<td>40</td>
</tr>
<tr>
<td>ice_2.0</td>
<td>833</td>
<td>369</td>
<td>4584</td>
<td>484</td>
<td>1044</td>
</tr>
<tr>
<td>p_auss2</td>
<td>832</td>
<td>419</td>
<td>5948</td>
<td>640</td>
<td>721</td>
</tr>
<tr>
<td>r1_2000</td>
<td>17</td>
<td>8</td>
<td>333</td>
<td>20</td>
<td>187</td>
</tr>
<tr>
<td>torusg3-15</td>
<td>101</td>
<td>22</td>
<td>219</td>
<td>61</td>
<td>84</td>
</tr>
</tbody>
</table>

Selected Mittelmann’s benchmark problems where HDSDP is fastest (all the constraints are rank-one)

(Results run on an intel i11700K machine)
Application: optimal diagonal pre-conditioner [QYZ20]

Given matrix $M = X^TX > 0$, iterative method (e.g., CG) is often applied to solve

$$Mx = b$$

- Convergence of iterative methods depend on the condition number $\kappa(M)$
- Good performance needs pre-conditioning and we solve $P^{-1/2}MP^{-1/2}x' = b$
  
  A good pre-conditioner reduces $\kappa(P^{-1/2}MP^{-1/2})$
- Diagonal $P = D$ is called diagonal pre-conditioner
  
  Quality of diagonal pre-conditioner is hard to estimate

  Is it possible to find optimal $D^*$? SDP works!
Computational results: optimal diagonal pre-conditioner

\[ \min_{D \text{ diagonal, } D \succeq 0} \kappa(DMD) \]

subject to \( I \preceq DMD \preceq \kappa I \)

\[ \min_{D, \kappa} \kappa \]

subject to \( D \preceq M \)

\( \kappa D \succeq M \)

• Finding the optimal diagonal pre-conditioner is an SDP
• Two SDP blocks and sparse coefficient matrices
• Trivial dual interior-feasible solution \( (\delta, \text{diag}(D)) = (-1,0) \)
• 1 is an upperbound for the optimal objective value
• An ideal formulation for HDSDP
Computational results: optimal diagonal pre-conditioner

We generate random matrices $M$ and run different SDP methods

<table>
<thead>
<tr>
<th>$n$</th>
<th>Sparsity</th>
<th>HDSDP (start from $(-10^6, 0)$)</th>
<th>COPT</th>
<th>Mosek</th>
<th>SDPT3</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>0.05</td>
<td>7.1</td>
<td>6.8</td>
<td>9.1</td>
<td>18.0</td>
</tr>
<tr>
<td>1000</td>
<td>0.09</td>
<td>44.5</td>
<td>53.9</td>
<td>54.2</td>
<td>327.0</td>
</tr>
<tr>
<td>2000</td>
<td>0.002</td>
<td>34.3</td>
<td>307.1</td>
<td>374.7</td>
<td>572.3</td>
</tr>
<tr>
<td>5000</td>
<td>0.0002</td>
<td>64.3</td>
<td>&gt;1200</td>
<td>&gt;1200</td>
<td>&gt;1200</td>
</tr>
</tbody>
</table>

(Results run on Mac Mini with Apple Silicon)
Summary

HDSDP is

• a general purpose SDP solver
• using dual-scaling and simplified HSD
• developed with heuristics and intuitions from DSDP
• equipped with several new computational tricks
Today’s Talk

• New developments of ADMM based interior point (ABIP) Method
  (Q. Deng, W. Gao, B. Jiang, J. Liu, T. Liu, C. Xue, Y. Ye, C. Zhang et al.)

• HDSDP: Homogeneous dual-scaling SDP solver
  (W. Gao, D. Ge and Y. Ye)

• A Dimension Reduced Second-Order Method
  (C. Zhang, D. Ge and Y. Ye)
Dimension Reduced Second-Order Method: a motivation

- Consider the following unconstrained convex/nonconvex optimization
  \[ \min f(x), x \in \mathbb{R}^n; g^k = \nabla f(x^k), H^k = \nabla^2 f(x^k) \]

- First-order Method (FOM), where \( d \) is a direction using gradient
  \[ x^{k+1} = x^k + \alpha^k d^k \]
  including: GD, AGD, and many others.

- Second-order Method (SOM), where \( B \) is an approximation to Hessian \( H^k \)
  \[ B^k d^k = -g^k \]
  including: Newton’s method, LBFGS, Trust-region Method, etc.

- DRSOM: Dimension Reduced Second-Order Method
  motivation: using gradient and (maybe) partial Hessian?
DRSOM: a first glance

- The DRSOM constructs iterations by two directions

\[ x^{k+1} = x^k - \alpha_k^1 g_k^1 + \alpha_k^2 d_k^2 \]

where \( g_k = \nabla f(x_k) \), \( H_k = \nabla^2 f(x_k) \), and \( d_k = x_k - x_{k-1} \)

the paradigm is not new, e.g., Accelerated Gradient Method, Conjugate Gradient Method, etc.

- A new idea is to introduce a 2-D quadratic model to choose the “step-sizes”

\[
m_k^\alpha := f(x^k) + (c^k)^T \alpha + \frac{1}{2} \alpha^T Q_k^k \alpha
\]

\[
Q_k^k = \begin{bmatrix}
(g_k^T H_k^k g_k^k) & -(d_k^T H_k^k g_k^k) \\
-(d_k^T H_k^k g_k^k) & (d_k^T H_k^k d_k^k)
\end{bmatrix} \in S^2, \quad c_k^k = \begin{bmatrix}
-\|g_k^k\|^2 \\
+(g_k^T d_k^k)
\end{bmatrix} \in \mathbb{R}^2
\]

- Minimize \( m_k^\alpha(\alpha) \) for optimal stepsizes! (see the lecture notes by Ye†)

† https://web.stanford.edu/class/msande311/lecture12.pdf
DRSOM: a first glance

DRSOM can be seen as:

• “Adaptive” Accelerated Gradient Method

• A second-order method in the reduced 2-D subspace

\[
m^k_p(p) = f(x^k) + (g^k)^T(p) + \frac{1}{2} p^T(H^k)p, \quad p \in \text{span}\{g^k, d^k\}
\]

compare to, e.g., Dogleg method, 2-D Newton Trust-Region Method

\[
m^k_p(p) = f(x^k) + (g^k)^T(p) + \frac{1}{2} p^T(H^k)p, \quad p \in \text{span}\{g^k, H^{-1}g_k\}
\]

• A conjugate direction method exploring the Krylov subspace

for convex quadratic programming, DRSOM is equivalent to CG.
DRSOM: computing Hessian-vector product

In the DRSOM:

\[ Q^k = \begin{bmatrix} (g^k)^T H^k g^k & -(d^k)^T H^k g^k \\ -(d^k)^T H^k d^k & (d^k)^T H^k d^k \end{bmatrix} \]

How to cheaply obtain Q? Compute \( H^k g^k, H^k d^k \) first.

- Finite difference:
  \[
  H^{k+1} g^{k+1} \approx \frac{1}{\epsilon} \left[ g(x^{k+1} + \epsilon \cdot g^{k+1}) - g(x^{k+1}) \right] \\
  H^{k+1} d^{k+1} \approx g(x^{k+1}) - g(x^k)
  \]

- Analytic approach to fit modern automatic differentiation,
  \[
  H g = \nabla(\frac{1}{2} g^T g), H d = \nabla(d^T g),
  \]

- or use Hessian if readily available!

Then we compute Q therein.
DRSOM: subproblem strategies

Recall 2-D quadratic model:

\[ m^k_\alpha (\alpha) := f(x^k) + (c^k)^T \alpha + \frac{1}{2} \alpha^T Q^k \alpha \]

\[ Q^k = \begin{bmatrix} (g^k)^T H^k g^k & -(d^k)^T H^k g^k \\ -(d^k)^T H^k d^k & (d^k)^T H^k d^k \end{bmatrix} \in S^2, \quad c^k = \begin{bmatrix} -\|g^k\|^2 \\ + (g^k)^T d^k \end{bmatrix} \in \mathbb{R}^2 \]

If Q is indefinite, apply two strategies that ensure global convergence

• Adaptive trust-region:

\[ m^* := \min_{\alpha} m_\alpha (\alpha), \text{ s.t. } \|\alpha\| \leq \Delta_\alpha \]

• Adaptive Lagrangian penalty, “Radius-free”

\[ \psi_\alpha (\lambda) := \min f(x^k) + (c^k)^T \alpha + \frac{1}{2} \alpha^T Q^k \alpha + \lambda \|\alpha\|^2 \]

The two strategies are equivalent and each sub-problem can be solved efficiently.
DRSOM: general framework

At each iteration $k$, the DRSOM proceeds:

- Solving 2-D Quadratic model
- Computing quality of the approximation

$$\rho^k := \frac{f(x^k) - f(x^k + p^k)}{m_p^k(0) - m_p^k(p^k)} = \frac{f(x^k) - f(x^k + p^k)}{m_p^k(0) - m_p^k(\alpha^k)},$$

- If $\rho$ is too small, increase $\lambda$ (Lagrangian penalty) or decrease $\Delta$ (trust-region)
- Otherwise, decrease $\lambda$ or increase $\Delta$
Logistic Regression

• Solve the Multinomial Logistic Regression for the MNIST dataset.

• The MLR is convex, we compare DRSOM to SAGA and LBFGS

• DRSOM is comparable to FOM and SOM (not surprisingly)

<table>
<thead>
<tr>
<th>Epoch</th>
<th>Method</th>
<th>Test Error Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>SAGA</td>
<td>0.0754</td>
</tr>
<tr>
<td>10</td>
<td>LBFGS</td>
<td>0.1175</td>
</tr>
<tr>
<td>10</td>
<td>DRSOM</td>
<td>0.1108</td>
</tr>
<tr>
<td>40</td>
<td>SAGA</td>
<td>0.0754</td>
</tr>
<tr>
<td>40</td>
<td>LBFGS</td>
<td>0.0783</td>
</tr>
<tr>
<td>40</td>
<td>DRSOM</td>
<td>0.0790</td>
</tr>
</tbody>
</table>

A sample for MNIST dataset
Nonconvex L2-Lp minimization

- Consider nonconvex L2-Lp minimization, $p < 1$
  \[ f(x) = \|Ax - b\|_2^2 + \lambda \|x\|_p^p \]

- Smoothed version
  \[ f(x) = \|Ax - b\|_2^2 + \lambda \sum_{i=1}^{n} s(x_i, \epsilon)^p \]
  \[ s(x, \epsilon) = \begin{cases} 
  |x| & \text{if } |x| > \epsilon \\
  \frac{x^2}{2\epsilon} + \frac{\epsilon}{2} & \text{if } |t| \leq \epsilon 
  \end{cases} \]

- Compare DRSOM to Accelerated Gradient Descend (AGD), LBFGS, and Newton Trust-region

<table>
<thead>
<tr>
<th>n</th>
<th>m</th>
<th>DRSOM</th>
<th>AGD</th>
<th>LBFGS</th>
<th>Newton TR</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>10</td>
<td>18</td>
<td>43</td>
<td>14</td>
<td>6</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>31</td>
<td>72</td>
<td>23</td>
<td>7</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>47</td>
<td>136</td>
<td>42</td>
<td>10</td>
</tr>
<tr>
<td>200</td>
<td>10</td>
<td>21</td>
<td>27</td>
<td>15</td>
<td>5</td>
</tr>
<tr>
<td>200</td>
<td>20</td>
<td>23</td>
<td>45</td>
<td>21</td>
<td>6</td>
</tr>
<tr>
<td>200</td>
<td>100</td>
<td>40</td>
<td>131</td>
<td>39</td>
<td>9</td>
</tr>
<tr>
<td>1000</td>
<td>10</td>
<td>13</td>
<td>16</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>1000</td>
<td>20</td>
<td>16</td>
<td>23</td>
<td>13</td>
<td>5</td>
</tr>
<tr>
<td>1000</td>
<td>100</td>
<td>19</td>
<td>32</td>
<td>16</td>
<td>5</td>
</tr>
</tbody>
</table>

Iterations needed to reach $\varepsilon = 1e-6$

- DRSOM is comparable to full-dimensional SOM !
Sensor Network Location (SNL)

- Consider Sensor Network Location (SNL)

$$N_x = \{(i,j) : \|x_i - x_j\| = d_{ij} \leq r_d\}, \quad N_a = \{(i,k) : \|x_i - a_k\| = d_{ik} \leq r_d\}$$

where $r_d$ is a fixed parameter known as the radio range. The SNL problem considers the following QCQP feasibility problem,

$$\|x_i - x_j\|^2 = d_{ij}^2, \forall (i,j) \in N_x$$

$$\|x_i - a_k\|^2 = d_{ik}^2, \forall (i,k) \in N_a$$

- We can solve SNL by the nonconvex nonlinear least square (NLS) problem

$$\min_X \sum_{(i,j) \in N_x} (\|x_i - x_j\|^2 - d_{ij}^2)^2 + \sum_{(k,j) \in N_a} (\|a_k - x_j\|^2 - d_{kj}^2)^2.$$
Sensor Network Location (SNL)

• We can also apply the SDP relaxation to SNL:

\[
\begin{align*}
\min & \quad 0 \cdot Z \\
\text{s.t.} & \quad Z_{[1:2,1:2]} = I, \\
& \quad (0; e_i - e_j) (0; e_i - e_j)^T \cdot Z = d_{ij}^2 \quad \forall (i, j) \in N_x, \\
& \quad (-a_k; e_i) (-a_k; e_i)^T \cdot Z = d_{ik}^2 \quad \forall (i, k) \in N_a
\end{align*}
\]

\[Z \succeq 0.\]

where \( Z = \begin{bmatrix} I & X \\ X^T & Y \end{bmatrix} \)

• If \( \text{rank}(Z) = 2 \), SDP relaxation is exact.

• Otherwise, relaxed solution \( Z^* \) can be used to initialize the NLS.
Sensor Network Location (SNL)

- Graphical results using SDP relaxation to initialize the NLS
- \( n = 80, m = 5 \) (anchors), radio range = 0.5, degree = 25, noise factor = 0.05
- Both Gradient Descend and DRSOM can find good solutions!
Sensor Network Location (SNL)

- Graphical results without SDP relaxation, is DRSOM better?
- DRSOM can still converge to optimal solutions
Neural Networks and Deep Learning

To use DRSOM in machine learning problems

• We apply the mini-batch strategy to a vanilla DRSOM

• Use Automatic Differentiation to compute gradients

• Train ResNet18 Model with CIFAR 10

• Set Adam with initial learning rate 1e-3
Neural Networks and Deep Learning

Training results for ResNet18 with DRSOM and Adam

Pros

• DRSOM has rapid convergence (30 epochs)
• DRSOM needs almost no tuning

Cons

• DRSOM may overfit the models
• Needs 4~5x time of Adam to run same number of epoch

Huge potential to be a standard optimizer for deep learning!

Test results for ResNet18 with DRSOM and Adam
DRSOM: A Summary

Dimension Reduced Second-order Method:

• Fast convergence in convex/nonconvex problems

• Comparable performance to SOM but no matrix inversion

• Typically better solutions than FOM for solving nonconvex problems

• Big potential for Deep Learning and other nonconvex learning tasks

Takeaway: Asia can play a big role in developing Open-Source Numerical Optimization Solvers!