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

  (Q. Deng, W. Gao, B. Jiang, J. Liu, T. Liu, C. Xue, C. Zhang, et al.)

- 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:

$$(P)\quad \begin{array}{ll} \min & \mathbf{c}^\top \mathbf{x} \\ \text{s.t.} & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq 0 \end{array} \qquad (D)\quad \begin{array}{ll} \max & \mathbf{b}^\top \mathbf{y} \\ \text{s.t.} & A^\top \mathbf{y} + \mathbf{s} = \mathbf{c} \\ & \mathbf{s} \geq 0 \end{array}$$

- For IPM, initial feasible interior solutions are hard to find
- Consider homogeneous and self-dual (HSD) LP here!

$$\begin{array}{ll} \min & \beta(n+1)\theta + \mathbf{1}(\mathbf{r}=0) + \mathbf{1}(\xi = -n-1) \\ \text{s.t.} & Q\mathbf{u} = \mathbf{v}, \\ & \mathbf{y} \text{ free}, \mathbf{x} \geq 0, \tau \geq 0, \theta \text{ free}, \mathbf{s} \geq 0, \kappa \geq 0 \end{array}$$

where

$$Q = \begin{bmatrix} 0 & A & -\mathbf{b} & \bar{\mathbf{b}} \\ -A^\top & 0 & \mathbf{c} & -\bar{\mathbf{c}} \\ \mathbf{b}^\top & -\mathbf{c}^\top & 0 & \bar{\mathbf{z}} \\ -\bar{\mathbf{b}}^\top & \bar{\mathbf{c}}^\top & -\bar{\mathbf{z}} & 0 \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} \mathbf{y} \\ \mathbf{x} \\ \tau \\ \theta \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} \mathbf{r} \\ \mathbf{s} \\ \kappa \\ \xi \end{bmatrix}, \quad \bar{\mathbf{b}} = \mathbf{b} - A\mathbf{e}, \quad \bar{\mathbf{c}} = \mathbf{c} - \mathbf{e}, \quad \bar{\mathbf{z}} = \mathbf{c}^\top \mathbf{e} + 1$$

ABIP – Subproblem

- Introduce log-barrier penalty for HSD LP

$$\begin{aligned} \min \quad & B(\mathbf{u}, \mathbf{v}, \mu) \\ \text{s.t.} \quad & Q\mathbf{u} = \mathbf{v} \end{aligned}$$

where $B(\mathbf{u}, \mathbf{v}, \mu)$ 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

$$\begin{aligned} \min \quad & \mathbf{1}(Q\tilde{\mathbf{u}} = \tilde{\mathbf{v}}) + B\big(\mathbf{u}, \mathbf{v}, \mu^k\big) \\ \text{s.t.} \quad & (\tilde{\mathbf{u}}, \tilde{\mathbf{v}}) = (\mathbf{u}, \mathbf{v}) \end{aligned}$$

The augmented Lagrangian function

$$\mathcal{L}_\beta\big(\tilde{\mathbf{u}}, \tilde{\mathbf{v}}, \mathbf{u}, \mathbf{v}, \mu^k, \mathbf{p}, \mathbf{q}\big) := \mathbf{1}(Q\tilde{\mathbf{u}} = \tilde{\mathbf{v}}) + B\big(\mathbf{u}, \mathbf{v}, \mu^k\big) - \langle \beta(\mathbf{p}, \mathbf{q}), (\tilde{\mathbf{u}}, \tilde{\mathbf{v}}) - (\mathbf{u}, \mathbf{v}) \rangle + \frac{\beta}{2} \|(\tilde{\mathbf{u}}, \tilde{\mathbf{v}}) - (\mathbf{u}, \mathbf{v})\|^2$$

ABIP – Rescale

Consider the following LP:

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

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

- Ruiz Rescaling: take

$$(D_1)_{jj} = \sqrt{\|A_{j,\cdot}\|_\infty} \text{ and } (D_2)_{ii} = \sqrt{\|A_{\cdot,i}\|_\infty}$$

- Pock-Chambolle Rescaling: take

$$(D_1)_{jj} = \sqrt{\|A_{j,\cdot}\|_{2-\alpha}} \text{ and } (D_2)_{ii} = \sqrt{\|A_{\cdot,i}\|_\alpha}$$

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

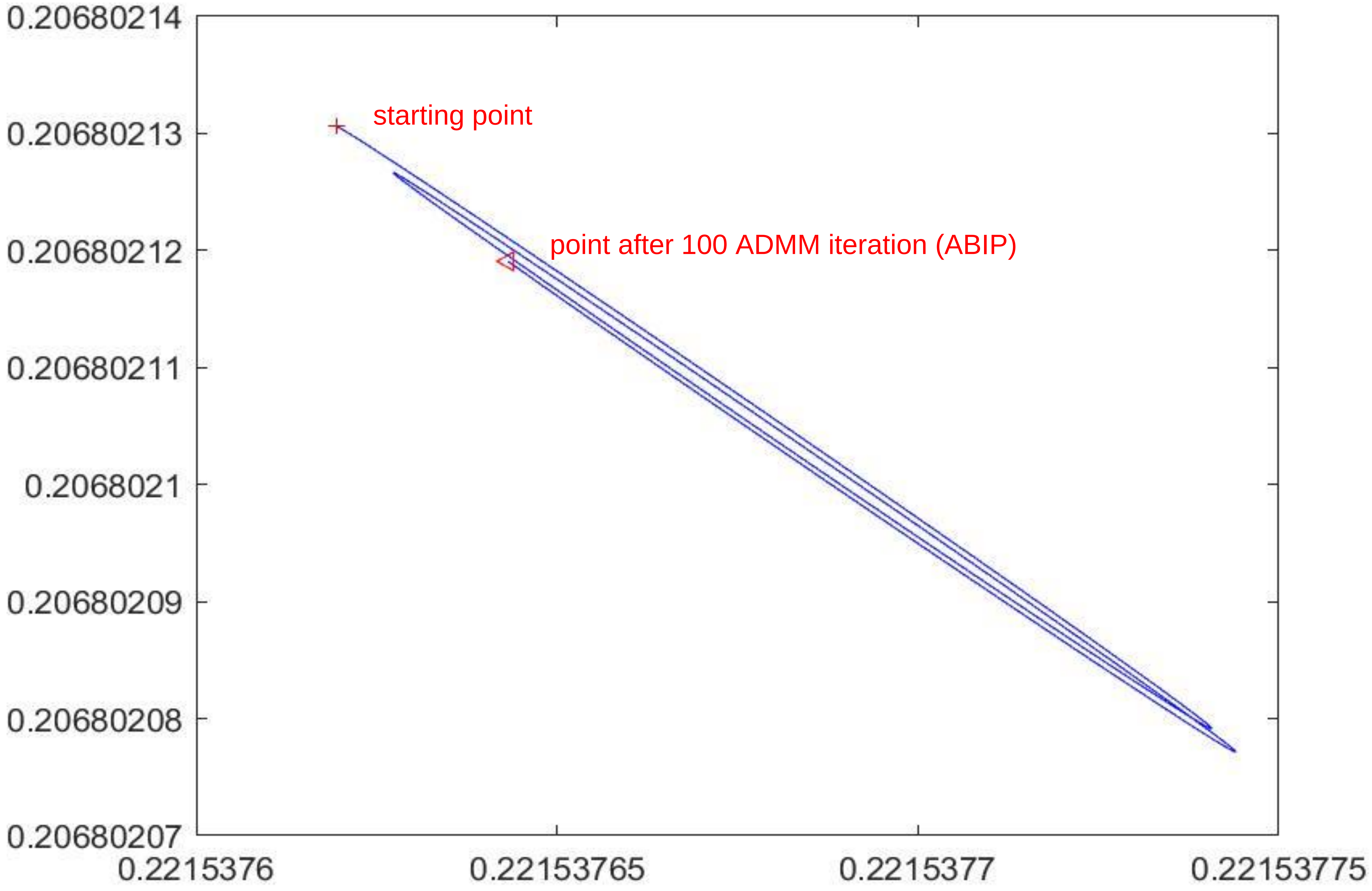
$$M \geq TH \quad \text{and} \quad M_k \bmod F = 0$$

where M denotes # total ADMM iterations so far, and M_k represents # ADMM iterations of the inner problem with respect to μ_k

- Significantly reduce # ADMM iterations!

ABIP – Restart

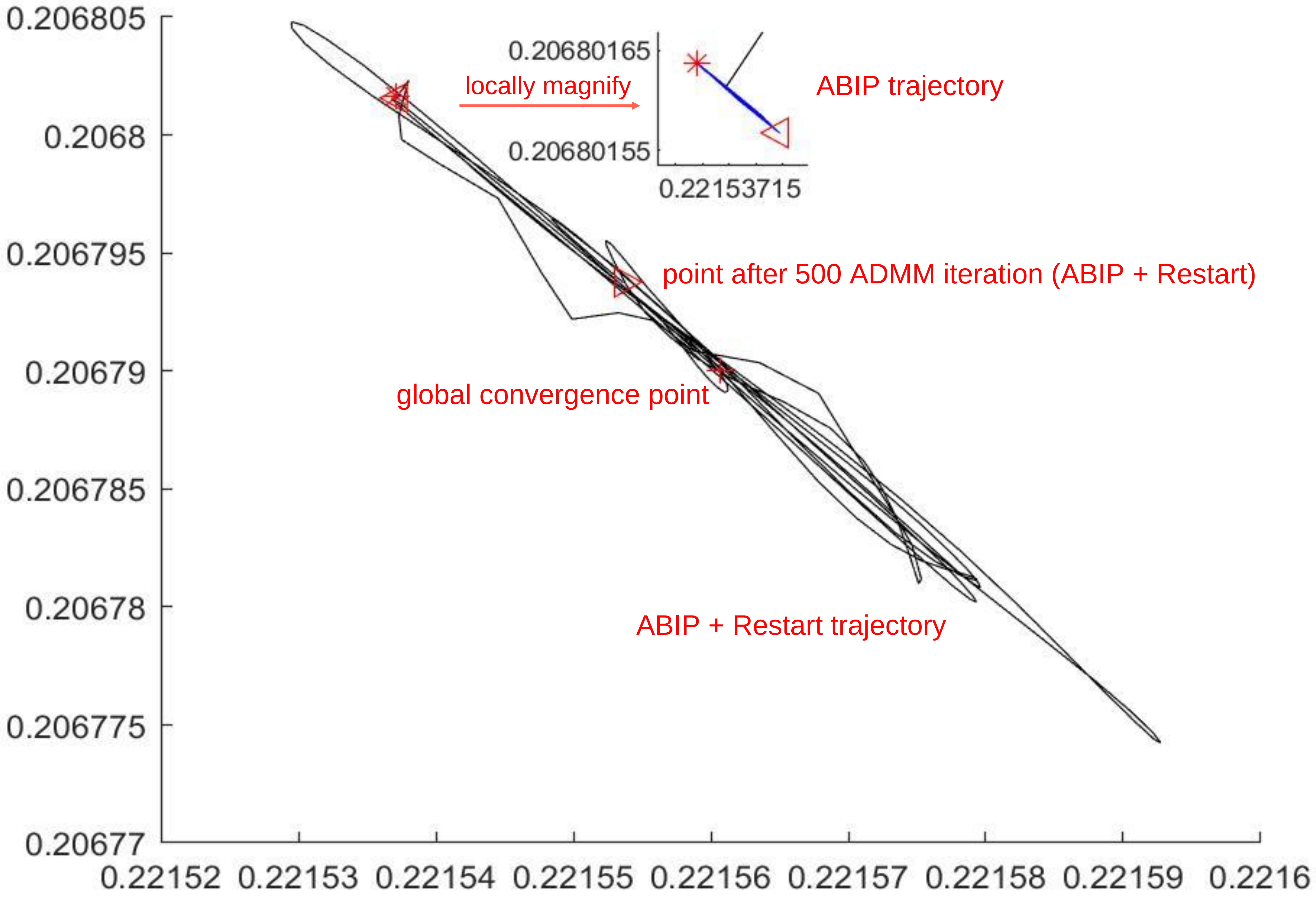
- ABIP tends to induce a spiral trajectory



Instance SC50B (only plot the first two dimension,)

ABIP – Restart

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



Instance SC50B (only plot the first two dimension, after restart)

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

$$
\begin{aligned}
&\text{Update } \tilde{\mathbf{u}}_{i+1}^k = (I+Q)^{-1}(\mathbf{u}_i^k + \mathbf{v}_i^k) \ ; \\
&\text{Update } \mathbf{u}_{i+1}^k \ ; \\
&\text{Update } \mathbf{v}_{i+1}^k = \mathbf{v}_i^k - \tilde{\mathbf{u}}_{i+1}^k + \mathbf{u}_{i+1}^k \ ;
\end{aligned}
$$

- The half update strategy is updating the dual variable $\boldsymbol{v}$ before updating the variable $\boldsymbol{u}$, where α is the step size

$$
\begin{aligned}
&\text{Update } \tilde{\mathbf{u}}_{i+1}^k = (I+Q)^{-1}(\mathbf{u}_i^k + \mathbf{v}_i^k) \ ; \\
&\text{Update } \mathbf{v}_{i+\frac{1}{2}}^k = \alpha\mathbf{u}_i^k + \mathbf{v}_i^k - \alpha\tilde{\mathbf{u}}_{i+1}^k \ ; \\
&\text{Update } \mathbf{u}_{i+1}^k \ ; \\
&\text{Update } \mathbf{v}_{i+1}^k = \mathbf{v}_{i+\frac{1}{2}}^k - \tilde{\mathbf{u}}_{i+1}^k + \mathbf{u}_{i+1}^k \ ;
\end{aligned}
$$

- Decrease # ADMM iterations on some specific dataset

ABIP – Adaptive barrier parameter μ 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

$$\xi = \frac{\min\{x_i s_i, \kappa\tau\}}{(\langle x, s\rangle + \kappa\tau)/(n+1)} \to \text{Measure of centrality}$$

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

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

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

Dimension

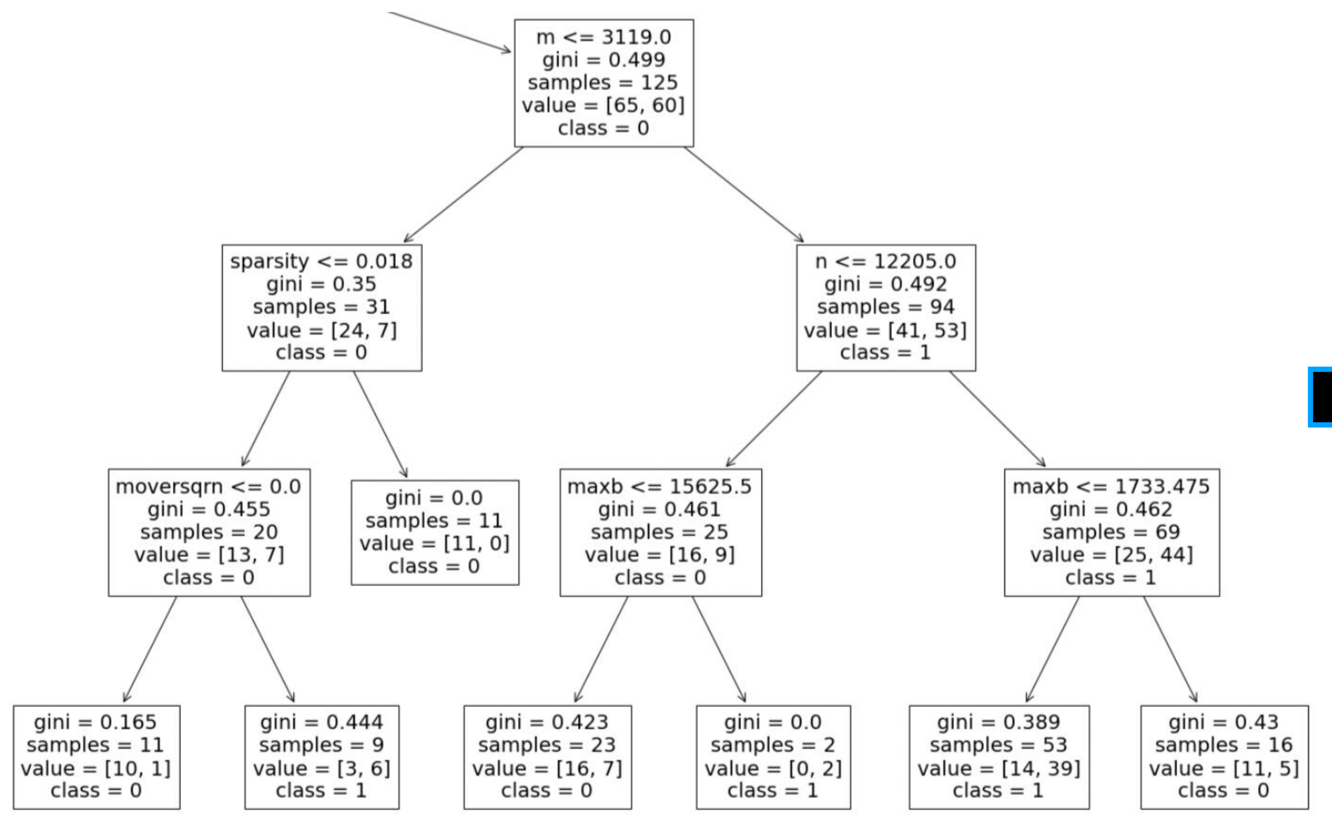
Sparsity

Constraint

Coefficient

Null Objective

...



Strategy 1

Strategy 2

...

Strategy *k*

- **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

$$\begin{aligned} \min_{x} \quad & 0 \\ \text{subject to} \quad & Ax = b \\ & x \geq 0, \end{aligned}$$

- **The dual problem is homogeneous and admits a trivial dual solution $(\mathbf{0}, \mathbf{0})$**
- **Approximate dual solution can be scaled down to $(\mathbf{0}, \mathbf{0})$**
- **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

Method	# Solved	# IPM	# ADMM	Avg.Time (s)
ABIP	65	74	265418	87.07
+ restart	68	74	88257	23.63
+ rescale	84	72	77925	20.44
+ hybrid μ (=ABIP3+)	**86**	**22**	**73738**	**14.97**

- Hybrid μ : 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

Method	# Solved	SGM
COPT	**240**	**1**
PDLP(Julia)	202	17.4
ABIP	192	34.8
ABIP3+ Integration	**212**	**16.7**

- PDLP (Lu et al. 2021) is a practical first-order method (i.e., the primal-dual hybrid gradient (PDHG) method) for linear programming, and it enhences 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.

Method	# Solved	SGM
PDLP(Julia)	**122**	**1**
ABIP3+	119	1.31

- Examples:

Instance	# nodes	PDLP (Julia)	ABIP3+
amazon0601	403394	117.54	**71.15**
coAuthorsDBLP	299067	51.66	**24.70**
web-BerkStan	685230	447.68	**139.75**
web-Google	916428	293.01	**148.18**

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	0.4950	0	0	0	-1.0000	0	0	0.9900
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!

# nodes	PDLP (Julia)	ABIP3+
10^4	8.60	**0.93**
10^5	135.67	**10.36**
10^6	2248.40	**60.32**

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

QP

$$\begin{aligned} \min\ & \frac{1}{2}x^T Q x + q^T x \\ \text{s.t.}\ & Ax = b \\ & x \geq 0 \end{aligned}$$

SOCP

$$\begin{aligned} \min\ & c^T x \\ \text{s.t.}\ & Ax = b \\ & x \in \mathcal{K} \end{aligned}$$

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

Convex QP is converted into SOCP as follows:

$$\begin{aligned} \min\ & \frac{1}{2}x^T Q x + q^T x \\ \text{s.t.}\ & Ax = b \\ & x \geq 0 \end{aligned} \quad \overset{Q=\Lambda^T\Lambda}{\Longrightarrow} \quad \begin{aligned} \min\ & \frac{1}{2}\bar{x}^T \bar{x} + q^T x \\ \text{s.t.}\ & Ax = b \\ & \bar{x} = \Lambda x \\ & x \geq 0 \end{aligned} \quad \Longrightarrow \quad \begin{aligned} \min\ & \nu + q^T x \\ \text{s.t.}\ & \eta = 1 \\ & Ax = b \\ & \bar{x} = \Lambda x \\ & \eta\nu \geq \frac{1}{2}\bar{x}^T\bar{x} \\ & x \geq 0 \end{aligned}$$

ABIP – Extension to SOCP and Beyond

ABIP iteration remains valid for general conic linear program

$$\begin{aligned} \min\ & c^T x \\ \text{s.t. } & Ax = b \\ & x \in \mathcal{K} \end{aligned}$$

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

Positive orthant

● $F(x) = -\log(x)$
● $x = \operatorname{argmin} \lambda F(x) + \frac{1}{2}|x - c|^2$
$= \frac{c+\sqrt{c^2+4\lambda}}{2}$

Second-order cone

● $F(\boldsymbol{x}) = -\log(t^2 - \|x\|^2)$, $\boldsymbol{x} = (t; x)$
● Can be solved by finding the root of quadratic functions

Positive semidefinite cone

● $F(\boldsymbol{x}) = -\log(\det x)$
● 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:

$$T_{IPM} = O\left(\log\left(\frac{1}{\varepsilon}\right)\right), \quad T_{ADMM} = O\left(\frac{1}{\varepsilon}\log\left(\frac{1}{\varepsilon}\right)\right)$$

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

elimination

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

- 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

	ABIP	OSQP	SCS	GUROBI
solved	48	47	48	46
1st	6	24	0	18
2nd	25	1	18	4
3rd	17	15	15	0
4th	0	7	15	24

ABIP – SVM

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

data	covtype	ijcnn1	news20	rcv1_train	real_sim	skin_orig_half
m	581012	49990	19996	20242	72309	122540
n	54	22	1355191	44504	20958	3
abip_time	158.76	2.58	1372.57	98.16	478.81	6.13
abip_iter	649	138	930	789	762	171
osqp_time	82.42	2.87	inf	inf	inf	0.45
osqp_iter	850	325	inf	inf	inf	25
racqp_time	221.25	6.00	531.67	66.30	74.23	4.93
racqp_iter	396	155	70	49	136	107
scs_time	122.38	4.45	inf	765.32	inf	47.80
scs_iter	950	400	inf	100	inf	2800
gurobi_time	1612.15	14.82	286.46	115.25	Inf	22.19
gurobi_iter	153	58	18	15	Inf	17

	ABIP	OSQP	RACQP	SCS	GUROBI
solved	6	3	6	4	5
1st	1	2	2	0	1
2nd	2	1	2	1	0
3rd	3	0	0	1	1
4th	0	0	2	1	1
5th	0	0	0	1	2

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:

$$(P)\quad \begin{aligned} \min\ & \frac{1}{2}x^TQx + c^Tx \\ \text{s.t. } & Ax = b \\ & x \in \mathcal{K} \end{aligned} \qquad\qquad (D)\quad \begin{aligned} \max\ & -\frac{1}{2}x^TQx + b^Ty \\ \text{s.t. } & Qx - A^Ty + c - s = 0 \\ & x \in \mathcal{K},\, s \in \mathcal{K}^* \end{aligned}$$

The KKT condition is

$$\begin{cases} Ax = b \\ Qx - A^Ty + c - s = 0 \\ x \in \mathcal{K} \\ s \in \mathcal{K}^* \\ s \perp x \end{cases} \quad \Longrightarrow \quad \mathbb{R}^m \times \mathcal{K} \ni \begin{bmatrix} y \\ x \end{bmatrix} \perp \begin{bmatrix} 0 & A \\ -A^T & Q \end{bmatrix} \begin{bmatrix} y \\ x \end{bmatrix} + \begin{bmatrix} -b \\ c \end{bmatrix} \in \{0\}^m \times \mathcal{K}^*$$

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

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

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

$$-A^Ty \in \mathcal{K}^*, b^Ty > 0$$

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

$$Qx = 0, Ax = 0, x \in \mathcal{K}, c^Tx < 0$$

ABIP – Conic Quadratic Programming

Homogeneous Embedding (Andersen and Y 1999)

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

- LCP infeasibility encoded by homogeneous operator $\mathcal{J}(z,\tau) = \left\{ \begin{bmatrix} Mz \\ \kappa \end{bmatrix} \middle| \kappa \leq -z^T q \right\}$, $\mathrm{dom}(\mathcal{J}) = \{(z,0): z^T Mz = 0\}$

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

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

where $Q = \mathcal{F} \cup \mathcal{J}$ $\mathrm{dom}(Q) = \mathrm{dom}(\mathcal{F}) \cup \mathrm{dom}(\mathcal{J})$ and $C_+ = C \times \mathbb{R}_+$.

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

$$
\begin{aligned}
\tilde{u}_{k+1} &= (I + \mathcal{Q})^{-1}(u_k + v_k) \\
u_{k+1} &= \mathrm{prox}_B(\tilde{u}_{k+1} - v_k) \\
v_{k+1} &= v_k + u_{k+1} - \tilde{u}_{k+1}
\end{aligned}
$$

ABIP – General Convex QP

- Maros-Meszaros convex QP dataset contains 138 instances

Table 1: $\epsilon = 1e - 3$

	ABIP-SOCP	ABIP-QCP	SCS
solved	110	124	130

- ABIP obtains significant improvement on convex QP

eps	ABIP-QCP	SCS
1e-4	114	125
1e-5	112	121

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)

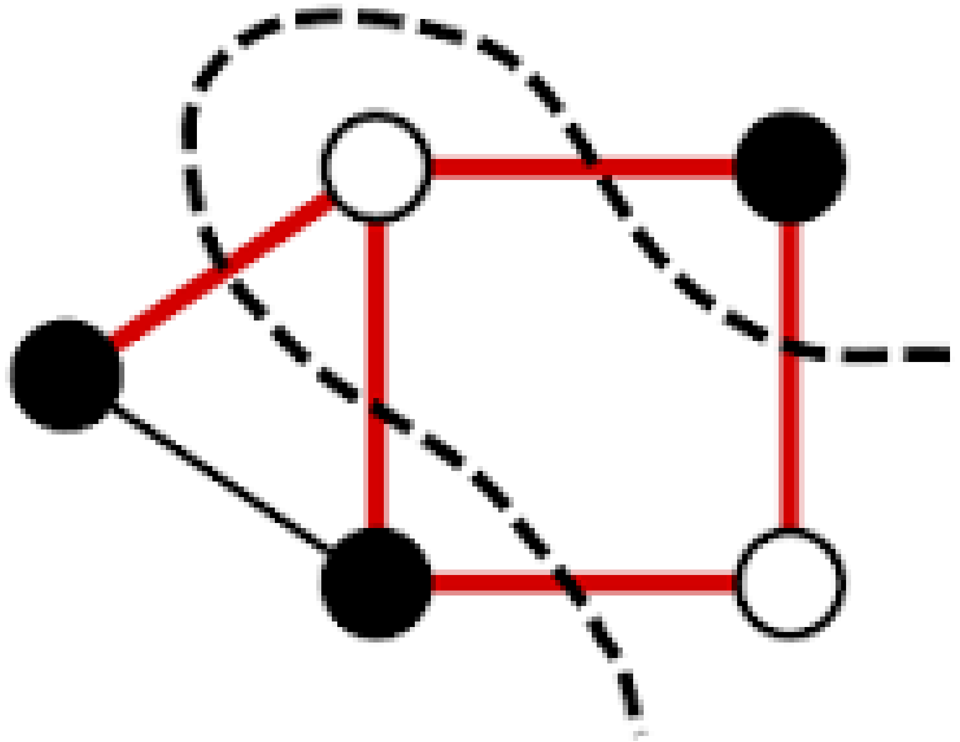
HDSDP: Homogeneous Dual-Scaling SDP solver

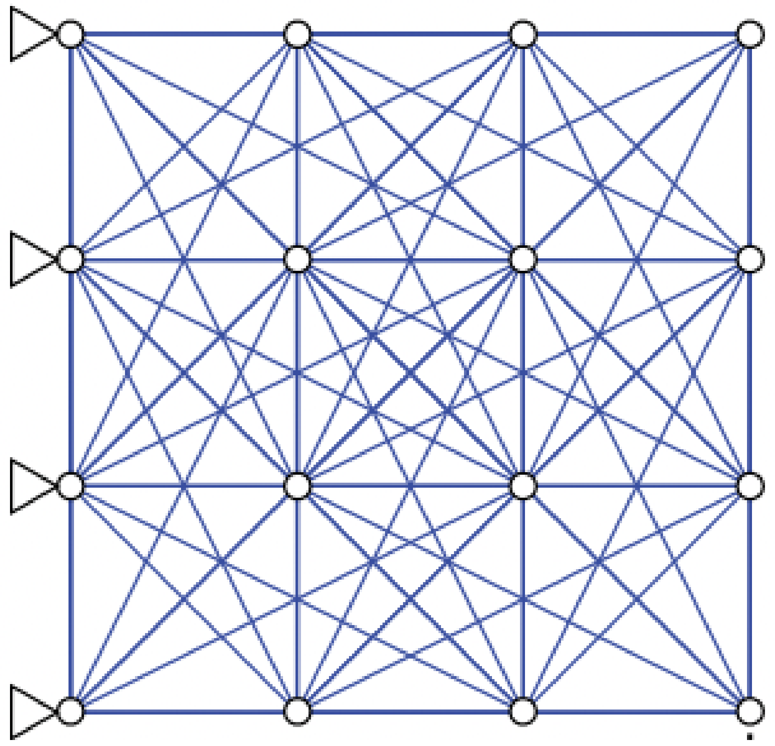
$$\begin{array}{rl} \min\limits_{X} & \langle C, X \rangle \\ \text{subject to} & \mathcal{A}X = b \\ & X \in \mathbb{S}^n_+ \end{array}$$

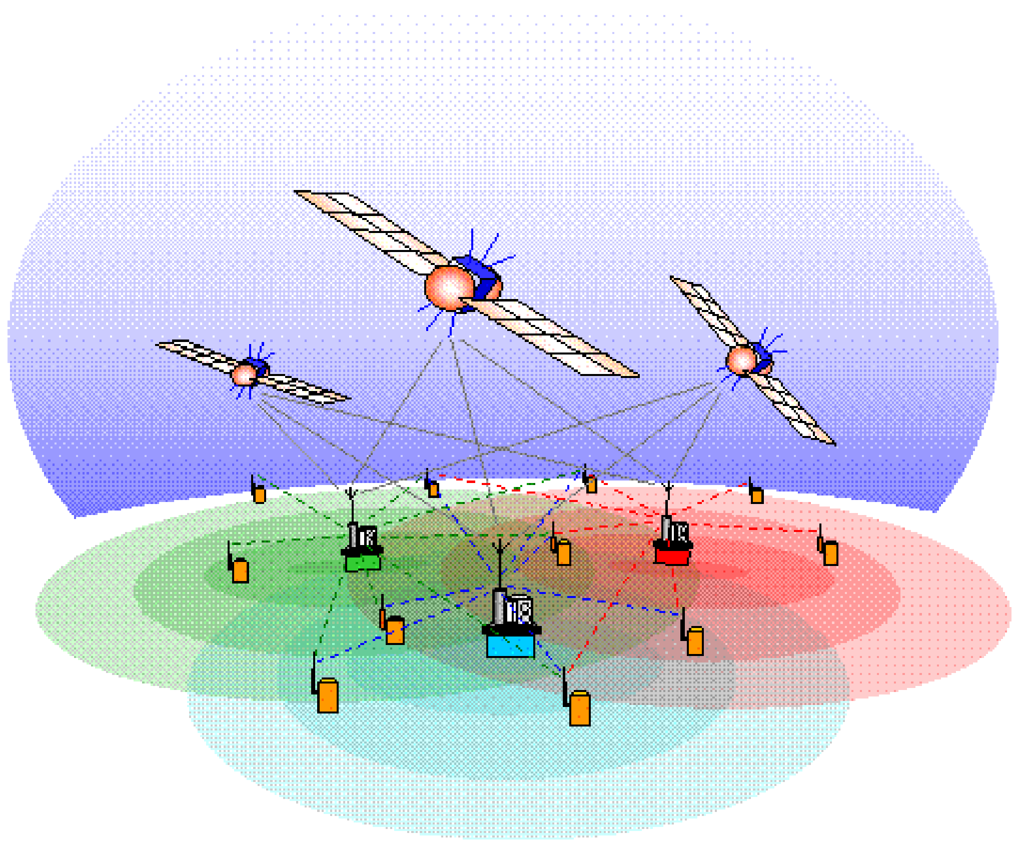
$$\begin{array}{rcl} \mathcal{A}\mathbf{X} - \mathbf{b}\tau & = & \mathbf{0} \\ -\mathcal{A}^*\mathbf{y} + \mathbf{C}\tau - \mathbf{S} & = & \mathbf{0} \\ \mathbf{b}^\top\mathbf{y} - \langle \mathbf{C}, \mathbf{X} \rangle - \kappa & = & 0 \\ \mathbf{X}, \mathbf{S} \succeq 0, & & \kappa, \tau \geq 0 \end{array}$$

$$\begin{array}{rcl} AP + PA + I & \preceq & 0 \\ P & \succeq & 0 \end{array}$$

$$\sum_{i=1}^{m} F_i y_i \preceq F_0$$







Semi-definite programming (SDP)

SDP is a mathematical programming problem taking form of

Primal

$$\begin{aligned} \min_{X} \quad & \langle C, X \rangle \\ \text{subject to} \quad & \mathcal{A}X = b \\ & X \in \mathbb{S}^n_+ \end{aligned}$$

Dual

$$\begin{aligned} \max_{y} \quad & b^\top y \\ \text{subject to} \quad & \mathcal{A}^* y + S = C \\ & S \in \mathbb{S}^n_+ \end{aligned}$$

- **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

$$\begin{aligned} AP + PA + I &\preceq 0 \\ P &\succeq 0 \end{aligned}$$

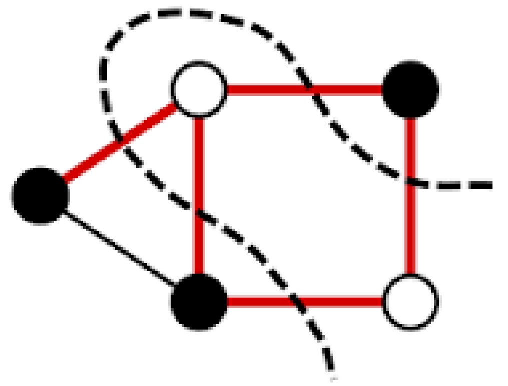
$$\sum_{i=1}^{m} F_i y_i \preceq F_0$$

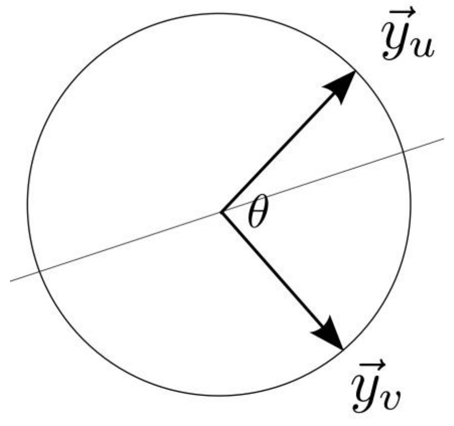
- **Dynamic system and numerical analysis**

  Lyapunov equation and Linear matrix inequalities

- **Combinatorial optimization and relaxation**

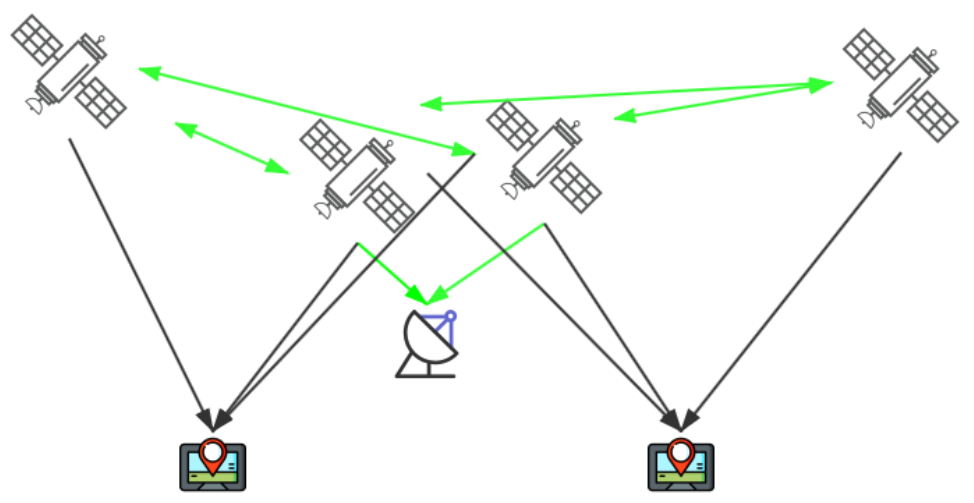
  Well-known 0.878 algorithm for Max-Cut





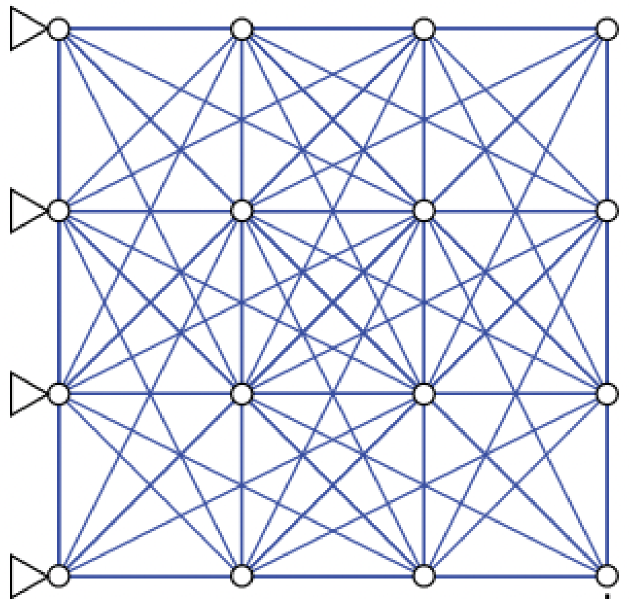
- **Graph realization and distance geometry**

  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**

$$\begin{aligned} \mathcal{A}X &= b \\ \mathcal{A}^*y + S &= C \\ XS &= 0 \end{aligned} \qquad \begin{aligned} \mathcal{A}X &= b \\ \mathcal{A}^*y + S &= C \\ XS &= \mu I \end{aligned} \qquad \begin{aligned} \mathcal{A}\Delta X &= -R_P \\ \mathcal{A}^*\Delta y + \Delta S &= -R_D \\ H_P(X\Delta S + \Delta X S) &= -R_\mu \end{aligned}$$

- **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

$$\begin{aligned} \mathcal{A}X &= b \\ \mathcal{A}^*y + S &= C \\ XS &= \mu I \end{aligned} \quad \Rightarrow \quad \begin{aligned} \mathcal{A}X &= b \\ \mathcal{A}^*y + S &= C \\ X &= \mu S^{-1} \end{aligned} \qquad \begin{aligned} \mathcal{A}(X + \Delta X) &= b \\ \mathcal{A}^*(y + \Delta y) + (S + \Delta S) &= C \\ \mu S^{-1} \Delta S S^{-1} + \Delta X &= \mu S^{-1} - X \end{aligned}$$

- **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 μ 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 κ, τ to detect infeasibility**
- **proves numerically stable**
- **solved by infeasible primal-dual path-following**

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

$$\begin{aligned} \mathcal{A}X - b\tau &= 0 \\ -\mathcal{A}^*y + C\tau - S &= 0 \\ b^\top y - \langle C, X\rangle - \kappa &= 0 \\ X, S \succeq 0, \qquad & \kappa, \tau \geq 0 \end{aligned}$$

$$\begin{aligned} \mathcal{A}X - b\tau &= 0 \\ -\mathcal{A}^*y + C\tau - S &= 0 \\ b^\top y - \langle C, X\rangle - \kappa &= 0 \\ X = \mu S^{-1} \qquad & \kappa\tau = \mu \end{aligned}$$

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

How to integrate dual-scaling and HSD ?

Hint.

$$\begin{aligned} \mathcal{A}X &= b \\ \mathcal{A}^*y + S &= C \\ XS &= \mu I \end{aligned} \qquad \begin{aligned} \mathcal{A}X &= b \\ \mathcal{A}^*y + S &= C \\ X &= \mu S^{-1} \end{aligned}$$

Apply the inversion trick!

Homogeneous dual-scaling algorithm

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

$$
\begin{aligned}
\mathcal{A}X - b\tau &= 0\\
-\mathcal{A}^* y + C\tau - S &= 0\\
b^\top y - \langle C, X\rangle - \kappa &= 0
\end{aligned}
$$

$$X = \mu S^{-1} \qquad \kappa = \mu\tau^{-1}$$

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

$$
\begin{pmatrix} \mu M & -b - \mu\mathcal{A}S^{-1}CS^{-1} \\ -b + \mu\mathcal{A}S^{-1}CS^{-1} & -\mu(\langle C, S^{-1}CS^{-1}\rangle + \tau^{-2}) \end{pmatrix}\begin{pmatrix}\Delta y\\ \Delta\tau\end{pmatrix} = \begin{pmatrix} b\tau \\ b^\top y - \mu\tau^{-1}\end{pmatrix} - \mu\begin{pmatrix}\mathcal{A}S^{-1} \\ \langle C, S^{-1}\rangle\end{pmatrix} + \mu\begin{pmatrix}\mathcal{A}S^{-1}RS^{-1} \\ \langle C, S^{-1}RS^{-1}\rangle\end{pmatrix}
$$

- **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

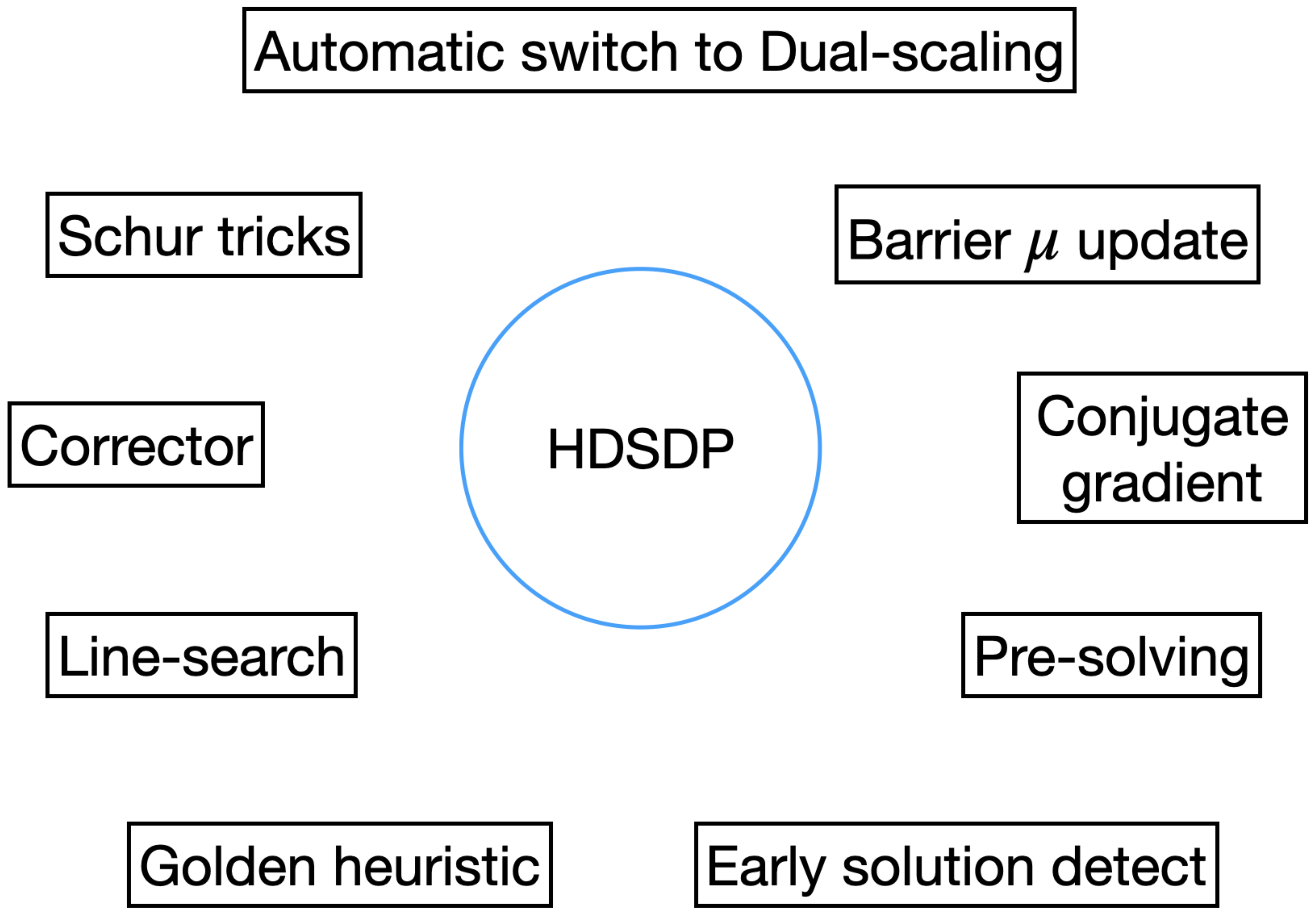
- **Infeasibility or an early feasible solution can be detected via the embedding**

New strategies are tailored for the method

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 μ**
- **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 dat $A_i = \sum_{r=1}^{\text{rank}(A_i)} \lambda_{ir} a_{i,r} a_{i,r}^\top$

To exploit low-rank structure (DSDP)

Technique	Approximate flops for Schur	Flops for S^{-1}
M1	$r_{\sigma(i)}(n^2+2n^2)+\kappa\sum_{j\ge i} f_{\sigma(j)}$	0
M2	$r_{\sigma(i)}(n^2+\kappa\sum_{j\ge i} f_{\sigma(j)})$	0
M3	$n\kappa f_{\sigma(i)}+n^3+\sum_{j\ge i}\kappa f_{\sigma(j)}$	$\mathcal{O}(n^3)$
M4	$n\kappa f_{\sigma(i)}+\sum_{j\ge i}\kappa(n+1) f_{\sigma(j)}$	$\mathcal{O}(n^3)$
M5	$\kappa(2\kappa f_{\sigma(i)}+1)\sum_{j\ge i} f_{\sigma(j)}$	$\mathcal{O}(n^3)$

2. **Compute** $M_{\sigma(i)\sigma(j)} = \sum_{r=1}^{\text{rank}(\mathbf{A}_{\sigma(i)})} (\mathbf{S}^{-1}\mathbf{a}_{\sigma(i),r})^\top \mathbf{A}_{\sigma(j)} (\mathbf{S}^{-1}\mathbf{a}_{\sigma(i),r})$

To exploit sparsity (SDPA)

Technique M3

$$\begin{pmatrix} \langle S^{-1}A_{\sigma(1)}S^{-1}, A_{\sigma(1)}\rangle & \rightarrow & \langle S^{-1}A_{\sigma(1)}S^{-1}, A_{\sigma(m)}\rangle \\ & \ddots & \vdots \\ & \rightarrow & \langle S^{-1}A_{\sigma(m)}S^{-1}, A_{\sigma(m)}\rangle \end{pmatrix}$$

2. **Compute** $M_{\sigma(i)\sigma(j)} = \langle \mathbf{B}_{\sigma(i)}\mathbf{S}^{-1}, \mathbf{A}_{\sigma(j)}\rangle, \forall j \ge i.$

Technique M5

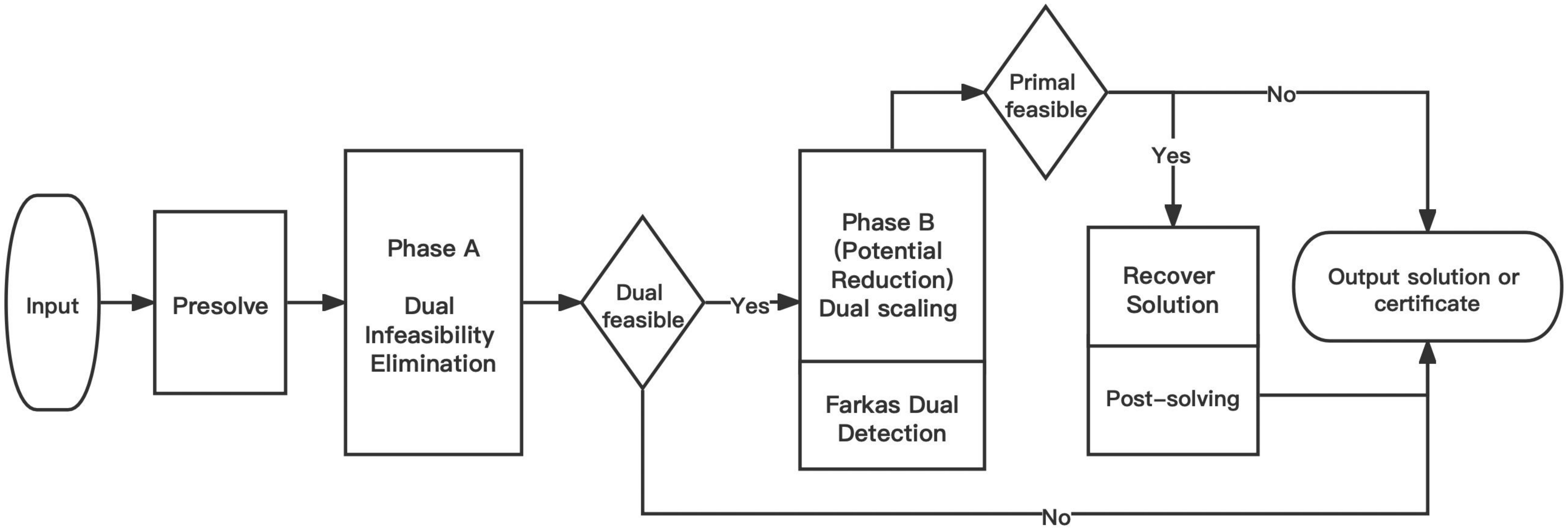
1. **Compute** $M_{\sigma(i)\sigma(j)} = \langle \mathbf{S}^{-1}\mathbf{A}_{\sigma(i)}\mathbf{S}^{-1}, \mathbf{A}_{\sigma(j)}\rangle, \forall j \ge i$ directly.

- **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**

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

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

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 $M = X^\top X \succ 0$, 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 $P^{-1/2} M P^{-1/2} x' = b$**

  A good pre-conditioner reduces $\kappa(P^{-1/2} M P^{-1/2})$
- **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

$$\min_{D \text{ diagonal}, D \succeq 0} \kappa(DMD) \quad \Rightarrow \quad \begin{array}{ll} \min\limits_{D,\kappa} & \kappa \\ \text{subject to} & I \preceq DMD \preceq \kappa I \end{array} \quad \Rightarrow \quad \begin{array}{ll} \min\limits_{D,\kappa} & \kappa \\ \text{subject to} & D \preceq M \\ & \kappa D \succeq M \end{array}$$

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

$$\begin{array}{ll} \max\limits_{\delta,d} & \delta \\ \text{subject to} & D - M \preceq 0 \\ & \delta M - D \preceq 0 \end{array}$$

Computational results: optimal diagonal pre-conditioner

We generate random matrices M and run different SDP methods

n	Sparsity	HDSDP (start from $(-10^6, 0)$)	COPT	Mosek	SDPT3
500	0.05	7.1	6.8	9.1	18.0
1000	0.09	44.5	53.9	54.2	327.0
2000	0.002	34.3	307.1	374.7	572.3
5000	0.0002	64.3	>1200	>1200	>1200

(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**

```
------------------------------------------------------------------------------------------
| DSDP is initialized with Ry = -1.824e+05 * I
| DSDP Phase A starts. Eliminating dual infeasibility
------------------------------------------------------------------------------------------

Phase A Log: 'P': Primal solution found. '*': Phase A ends. 'F': Error happens. 'M': Max iteration
------------------------------------------------------------------------------------------
| Iter |         pObj |         dObj |      dInf |       k/t |        mu |  step |    Pnorm |   E |
------------------------------------------------------------------------------------------
|    1 |  1.00000e+05 |  0.00000e+00 |  2.92e+06 | 0.00e+00 | 8.69e+00 |  0.00 |  1.0e+20 |     |
|    2 |  1.00000e+05 | -3.64854e+05 |  0.00e+00 | 0.00e+00 | 1.39e+00 |  1.00 |  1.3e+03 |   * |
------------------------------------------------------------------------------------------
Phase A Log Ends.

------------------------------------------------------------------------------------------
| DSDP Phase A ends with status: DSDP_PRIMAL_DUAL_FEASIBLE
| Elapsed Time: 1.895 seconds
------------------------------------------------------------------------------------------
| DSDP Phase A certificates primal-dual feasibility
| Primal relaxation penalty is set to  3.649e+06
| Perturbing dual iterations by  0.000e+00
| DSDP Phase B starts. Restarting dual-scaling
| Heuristic start: mu:  1.390e+00 pObj:  1.000e+05  dObj: -3.649e+05
------------------------------------------------------------------------------------------

Phase B Log: 'P': Primal solution found. '*': Optimal. 'F': Error happens. 'M': Max iteration
------------------------------------------------------------------------------------------
| Iter |              pObj |              dObj |      pInf |        mu |  step |    Pnorm |   E |
------------------------------------------------------------------------------------------
|    1 |  1.0000000000e+05 | -3.6485418601e+05 |  2.000e+00 | 1.39e+00 |  1.00 |  1.3e+03 |     |
|    2 |  1.0000000000e+05 | -1.8364017699e+04 |  2.000e+00 | 7.13e+01 |  0.01 |  1.6e+03 |     |
|    3 |  2.3013744393e+06 | -1.0413425712e+03 |  1.965e-05 | 4.04e+00 |  0.23 |  6.4e+01 |   P |
|    4 |  1.3036627128e+05 | -1.5343573903e+02 |  1.108e-06 | 5.88e-01 |  0.21 |  6.4e+01 |   P |
|    5 |  1.8955361279e+04 | -7.8842930079e+01 |  1.611e-07 | 2.92e-01 |  0.12 |  6.4e+01 |   P |
|    6 |  9.4118890551e+03 | -4.5382073201e+01 |  8.000e-08 | 1.50e-01 |  0.12 |  6.3e+01 |   P |
|    7 |  4.8187976557e+03 | -3.0772119147e+01 |  4.100e-08 | 6.82e-02 |  0.11 |  5.9e+01 |   P |
|    8 |  2.1880948334e+03 | -2.6133686830e+01 |  1.870e-08 | 1.27e-02 |  0.13 |  4.5e+01 |   P |
|    9 |  3.8580146936e+02 | -2.5626752469e+01 |  3.472e-09 | 7.14e-04 |  0.21 |  2.8e+01 |   P |
|   10 | -2.4021733499e+00 | -2.5601339383e+01 |  1.958e-10 | 3.60e-05 |  0.23 |  2.5e+01 |   P |
|   11 | -2.4431151604e+01 | -2.5600139036e+01 |  9.864e-12 | 7.20e-06 |  0.22 |  2.5e+01 |   P |
|   12 | -2.5366228786e+01 | -2.5600099863e+01 |  1.973e-12 | 1.44e-06 |  0.17 |  9.9e+00 |   P |
|   13 | -2.5553271187e+01 | -2.5600005654e+01 |  3.944e-13 | 2.88e-07 |  0.11 |  5.1e+01 |   P |
|   14 | -2.5590644923e+01 | -2.5600004411e+01 |  7.895e-14 | 5.76e-08 |  0.08 |  1.4e+01 |   P |
|   15 | -2.5598129984e+01 | -2.5600000256e+01 |  1.578e-14 | 1.15e-08 |  0.10 |  5.7e+01 |   P |
|   16 | -2.5599625541e+01 | -2.5600000196e+01 |  3.160e-15 | 2.30e-09 |  0.08 |  1.6e+01 |   P |
|   17 | -2.5599925142e+01 | -2.5600000015e+01 |  6.317e-16 | 4.62e-10 |  0.09 |  6.4e+01 |   P |
|   18 | -2.5599985006e+01 | -2.5600000010e+01 |  1.265e-16 | 9.23e-11 |  0.08 |  2.2e+01 |   * |
------------------------------------------------------------------------------------------
Phase B Log Ends.

------------------------------------------------------------------------------------------
| DSDP Phase B ends with status: DSDP_OPTIMAL
| Elapsed Time: 143.418 seconds
------------------------------------------------------------------------------------------
| DSDP Ends.
------------------------------------------------------------------------------------------
```

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

$$x^{k+1} = x^k - \alpha_1^k g^k + \alpha_2^k d^k$$

where $g^k = \nabla f(x^k), H^k = \nabla^2 f(x^k), \text{and } d^k = x^k - x^{k-1}$

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”

$$m_\alpha^k(\alpha) := f(x^k) + (c^k)^T\alpha + \frac{1}{2}\alpha^T Q^k \alpha$$

$$Q^k = \begin{bmatrix} (g^k)^T H^k g^k & -(d^k)^T H^k g^k \\ -(d^k)^T H^k g^k & (d^k)^T H^k d^k \end{bmatrix} \in \mathcal{S}^2, c^k = \begin{bmatrix} -\left\|g^k\right\|^2 \\ +(g^k)^T d^k \end{bmatrix} \in \mathbb{R}^2$$

- Minimize $m_\alpha^k(\alpha)$ 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

$$m_p^k(p) = f(x^k) + (g^k)^T(p) + \frac{1}{2}p^\top(H^k)p, \quad p \in \mathsf{span}\{g^k, d^k\}$$

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

$$m_p^k(p) = f(x^k) + (g^k)^T(p) + \frac{1}{2}p^\top(H^k)p, \quad p \in \mathsf{span}\{g^k, H^{-1}g_k\}$$

- 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:

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

How to cheaply obtain Q? Compute $\boldsymbol{H^k g^k, H^k d^k}$ 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,

$$Hg = \nabla(\frac{1}{2} g^T g), Hd = \nabla(d^T g),$$

- or use Hessian if readily available !

Then we compute Q therein.

DRSOM: subproblem strategies

Recall 2-D quadratic model:

$$m_\alpha^k(\alpha) := f(x^k) + (c^k)^T\alpha + \frac{1}{2}\alpha^T Q^k \alpha$$

$$Q^k = \begin{bmatrix} (g^k)^T H^k g^k & -(d^k)^T H^k g^k \\ -(d^k)^T H^k g^k & (d^k)^T H^k d^k \end{bmatrix} \in \mathcal{S}^2, c^k = \begin{bmatrix} -\left\|g^k\right\|^2 \\ +(g^k)^T d^k \end{bmatrix} \in \mathbb{R}^2$$

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

- Adaptive trust-region:

$$m^* := \min_\alpha \; m_\alpha(\alpha), \text{ s.t. } \|\alpha\| \le \Delta_\alpha$$

- 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:

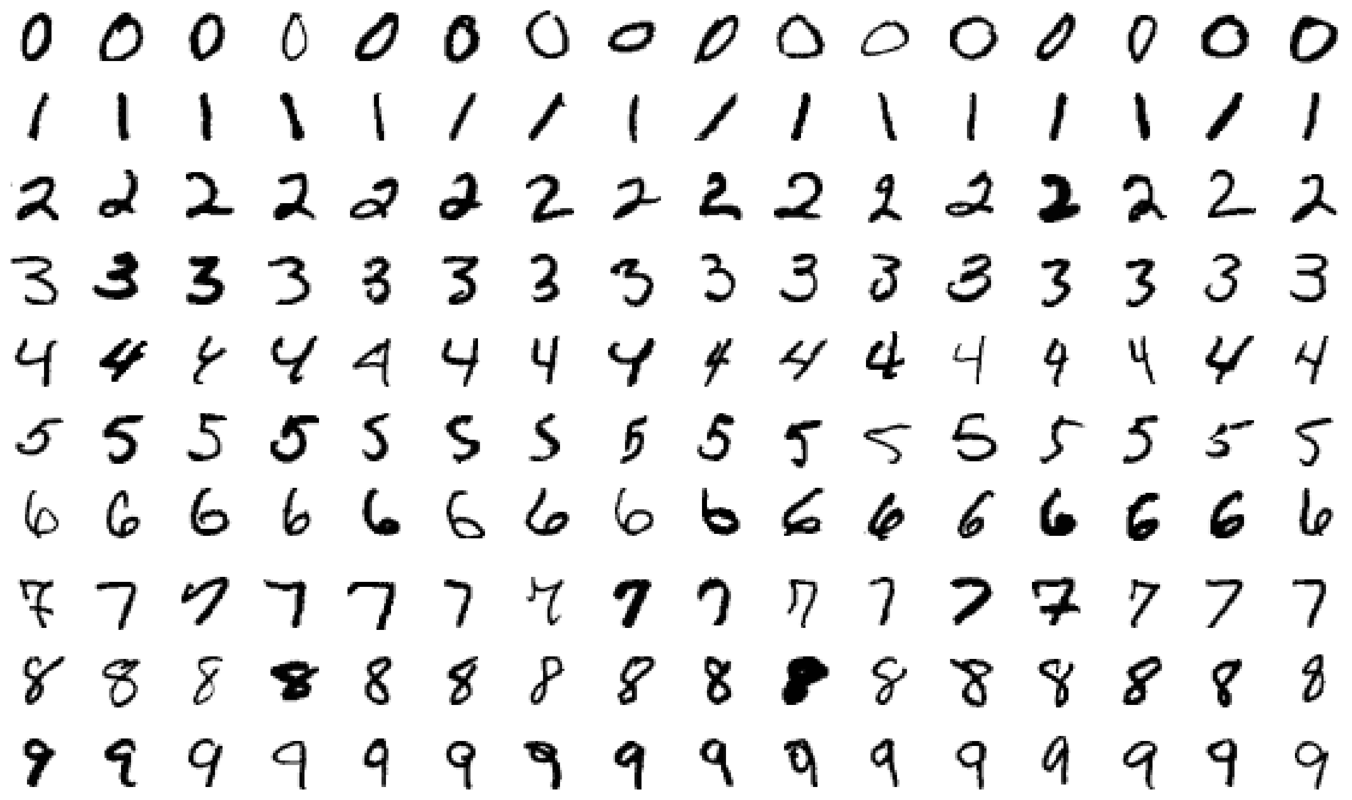
- Solving 2-D Quadratic model
- Computing quality of the approximation

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

- If ρ is too small, increase λ (Lagrangian penalty) or decrease Δ (trust-region)
- Otherwise, decrease λ or increase Δ

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)



A sample for MNIST dataset

Epoch	Method	Test Error Rate
10	SAGA	0.0754
10	LBFGS	0.1175
10	DRSOM	0.1108
40	SAGA	0.0754
40	LBFGS	0.0783
40	DRSOM	0.0790

Nonconvex L2-Lp minimization

- Consider nonconvex L2-Lp minimization, p < 1

$$f(x) = \|Ax - b\|_2^2 + \lambda \|x\|_p^p$$

- Smoothed version

$$f(x) = \|Ax - b\|_2^2 + \lambda \sum_{i=1}^{n} s(x_i, \epsilon)^p$$

$$s(x, \epsilon) = \begin{cases} |x| & \text{if } |x| > \epsilon \\ \frac{x^2}{2\epsilon} + \frac{\epsilon}{2} & \text{if } |t| \le \epsilon \end{cases}$$

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

Iterations needed to reach ε = 1e-6

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

- DRSOM is comparable to full-dimensional SOM !

Sensor Network Location (SNL)

- Consider Sensor Network Location (SNL)

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

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

$$\|x_i - x_j\|^2 = d_{ij}^2, \forall (i,j) \in N_x$$
$$\|x_i - a_k\|^2 = \bar{d}_{ik}^2, \forall (i,k) \in N_a$$

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

$$\min_X \sum_{(i<j,j) \in N_x} (\|x_i - x_j\|^2 - d_{ij}^2)^2 + \sum_{(k,j) \in N_a} (\|a_k - x_j\|^2 - \bar{d}_{kj}^2)^2.$$

Sensor Network Location (SNL)

- We can also apply the SDP relaxation to SNL:

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

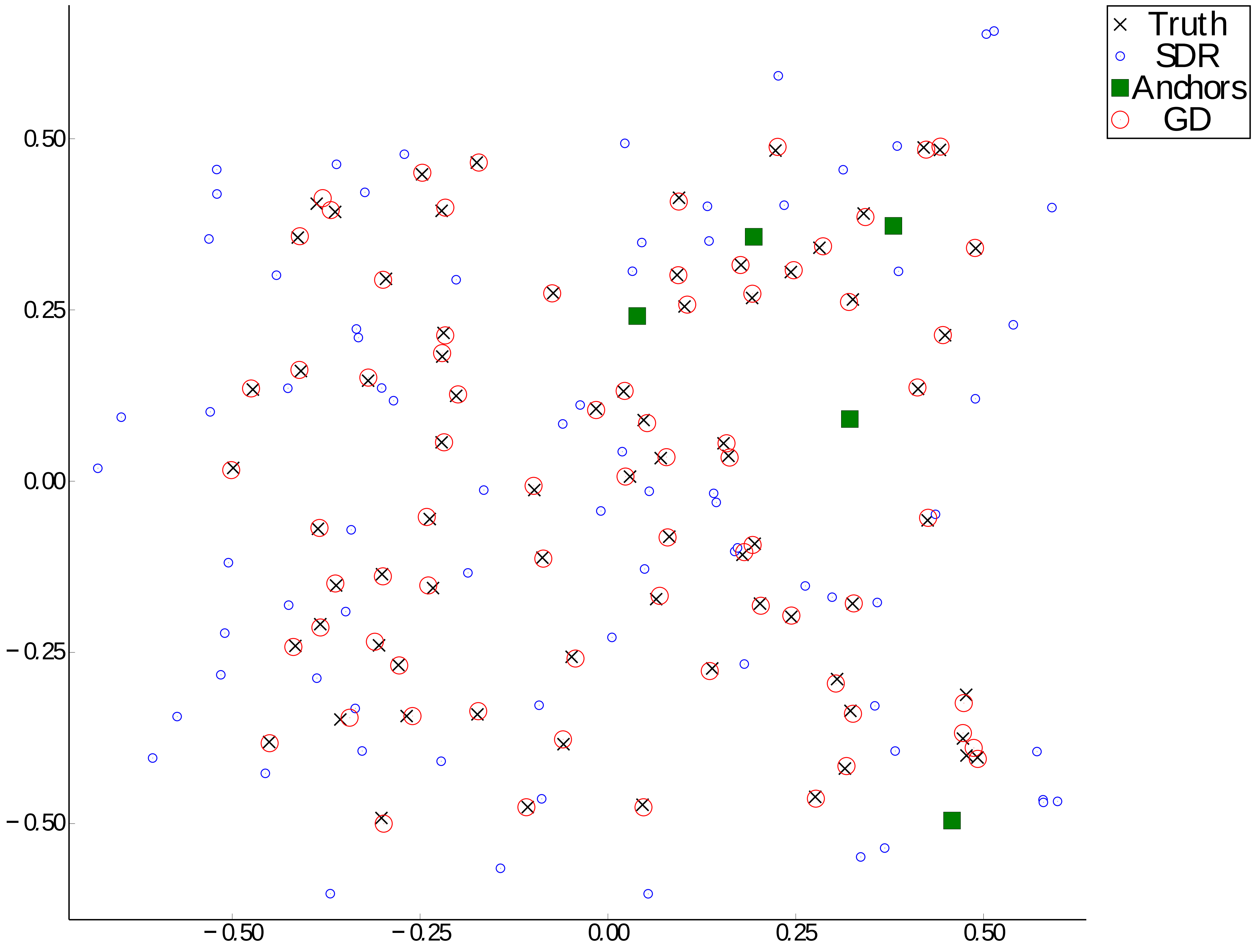
where $Z = \begin{bmatrix} I & X \\ X^T & Y \end{bmatrix}$

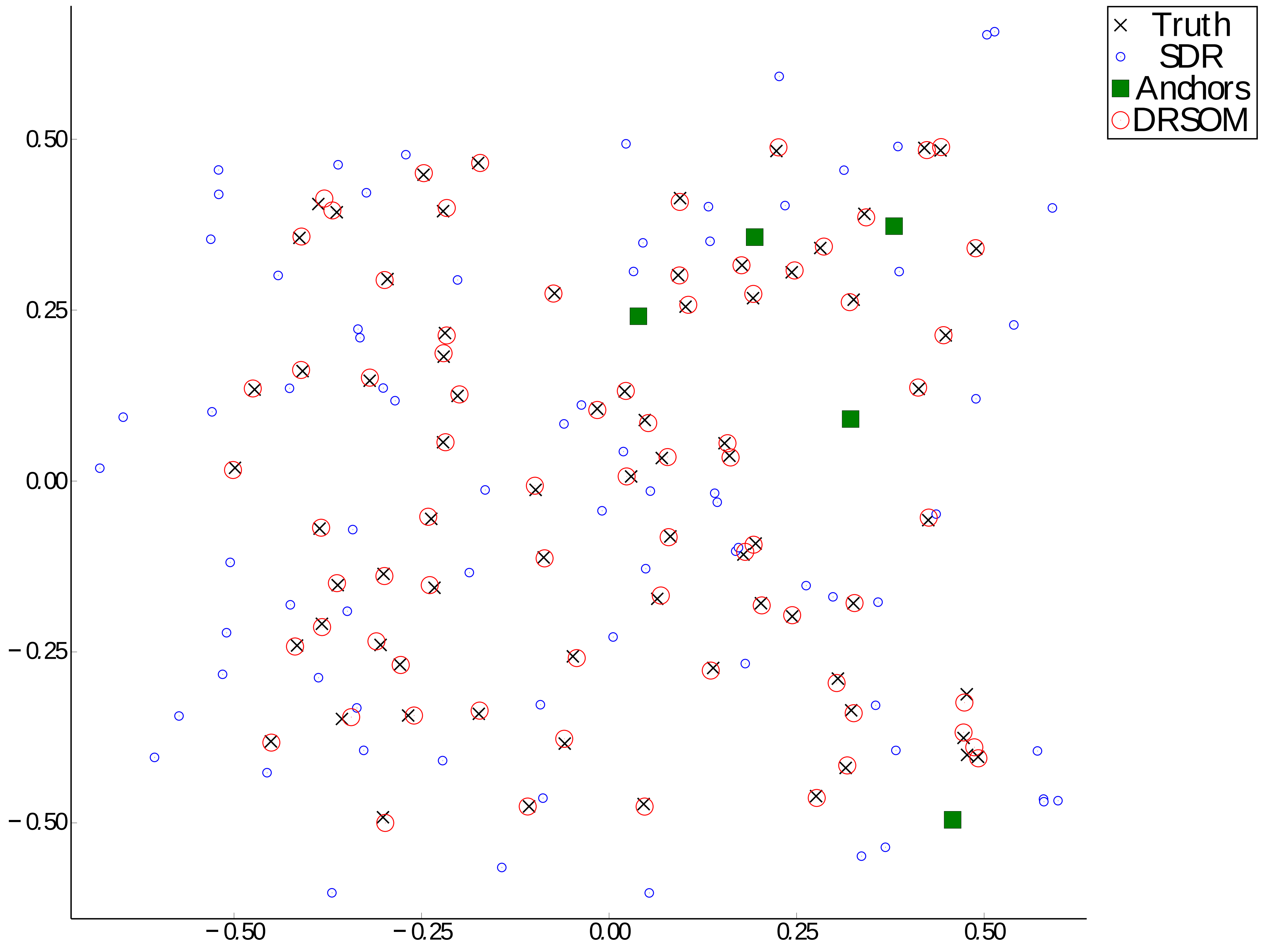
- If rank(Z) = 2, SDP relaxation is exact.

- Otherwise, relaxed solution Z* can be used to initialize the NLS

Sensor Network Location (SNL)

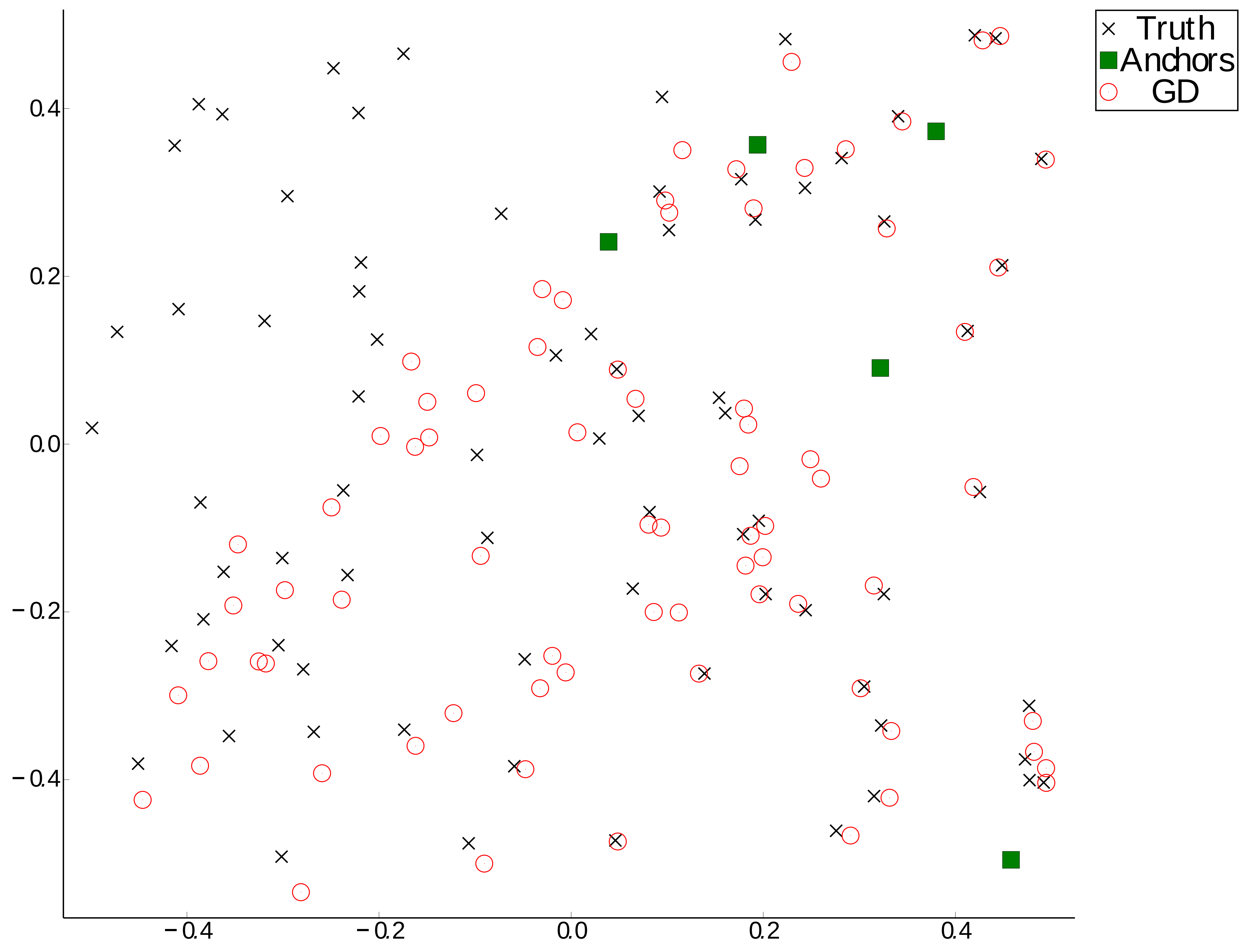
- Graphical results using SDP relaxation to initialize the NLS
- n = 80, m = 5 (anchors), radio range = 0.5, degree = 25, noise factor = 0.05
- Both Gradient Descend and DRSOM can find good solutions !

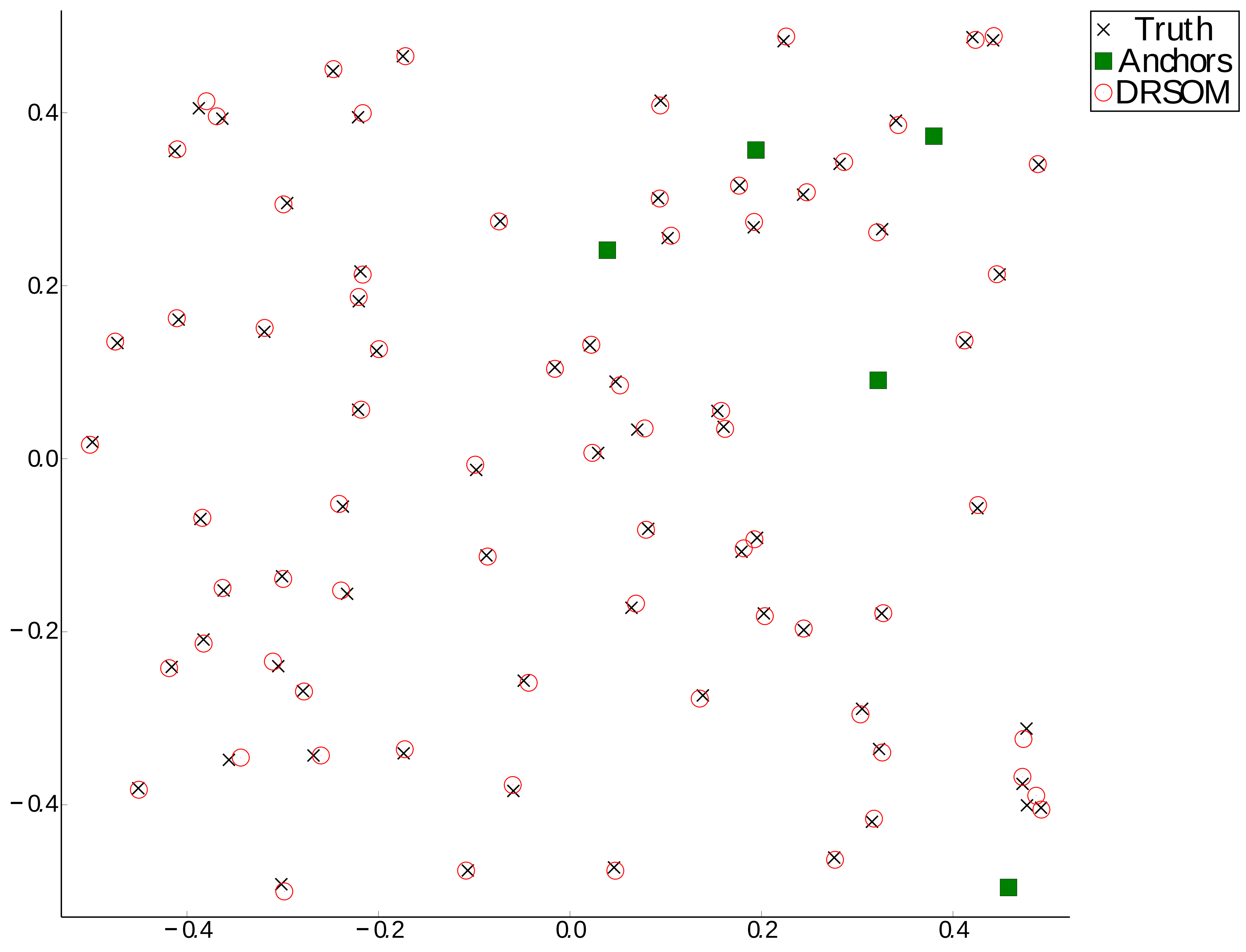




Sensor Network Location (SNL)

- Graphical results without SDP relaxation, is DRSOM better?
- DRSOM can still converge to optimal solutions

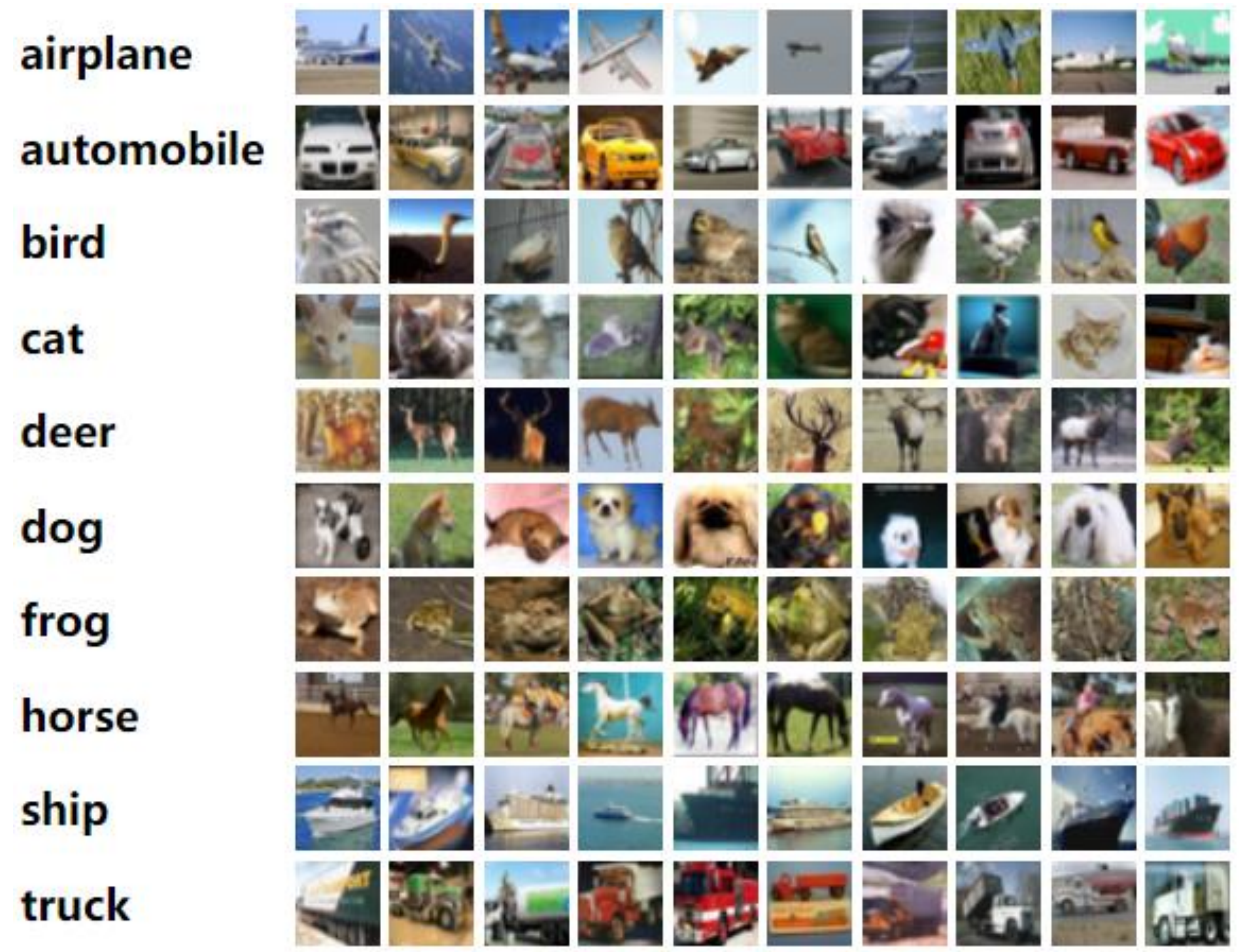




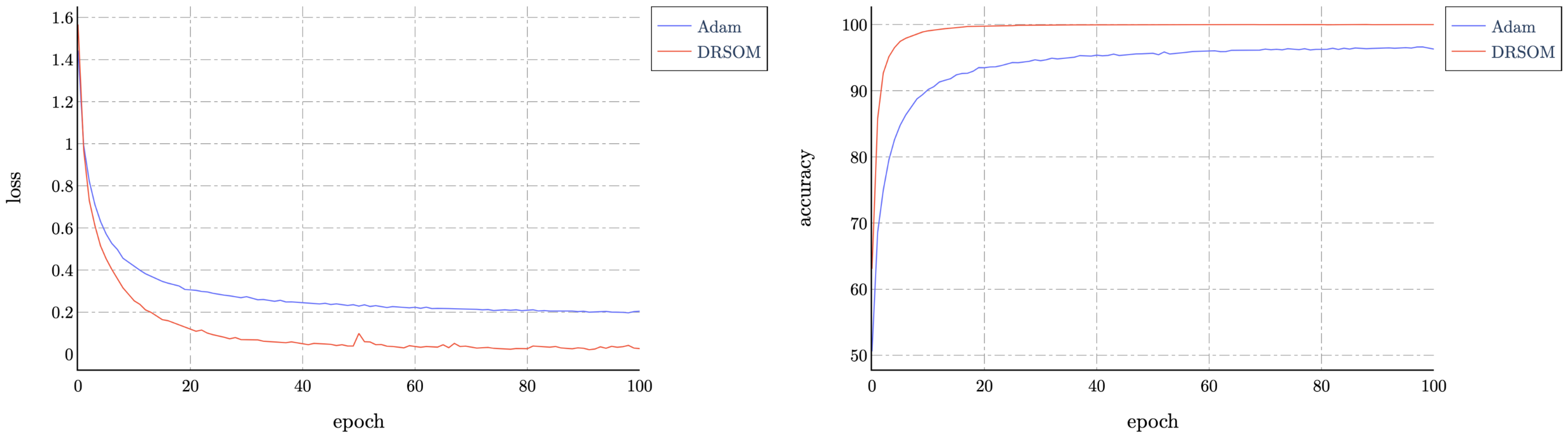
Neural Networks and Deep Learning

To use DRSOM in machine learning problems

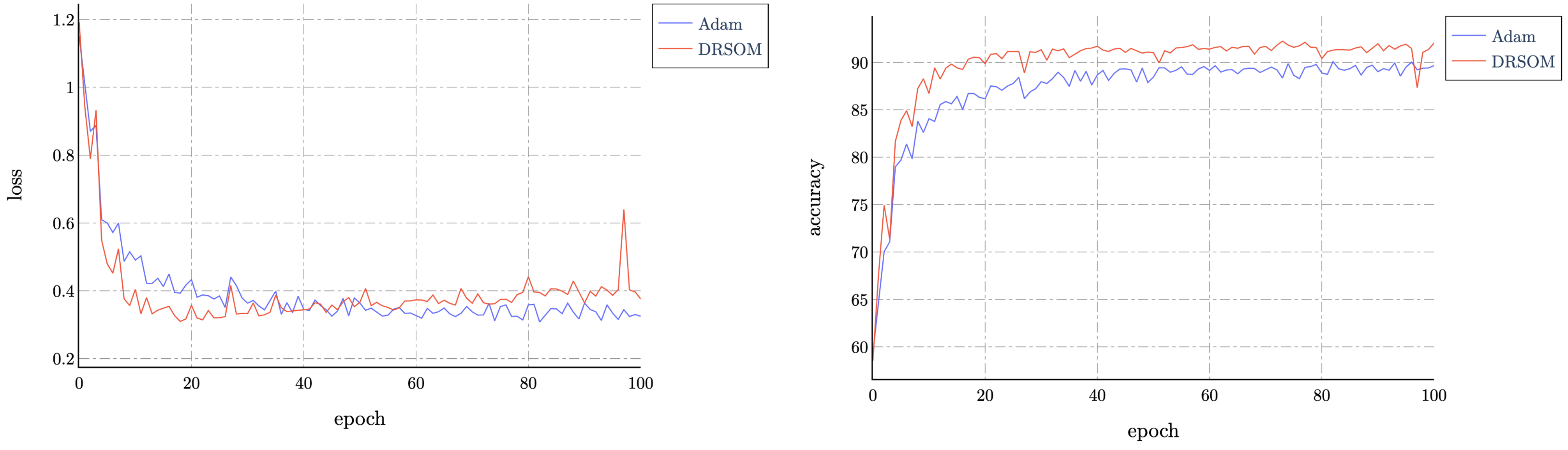
- We apply the mini-batch strategy to a vanilla DRSOM
- Use Automatic Differentiation to compute gradients
- Train ResNet18 Model with CIFAR 10
- Set Adam with initial learning rate 1e-3



Neural Networks and Deep Learning



Training results for ResNet18 with DRSOM and Adam



Test 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!

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!