New Developments of Computational Algorithms for Large-Scale Convex and Nonconvex Optimization

INSTITUTE OF MATHEMATICS, HANOI

AUGUST 23, 2022

Yinyu Ye
Stanford University
(Currently Visiting VinU...)

Today’s Talk

- **New developments of ADMM-based interior point (ABIP) Method**
- **HDSDP: Homogeneous Dual-Scaling SDP solver**
- **A Dimension Reduced Second-Order Method**

Introduction to ADMM

- Consider the following convex optimization problem

$$\begin{aligned} &\min \ f(\boldsymbol{x}) \\ \text{s.t.} \quad &\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b} \\ &\boldsymbol{x} \in \boldsymbol{X} \end{aligned}$$

- Where f is a convex function, and $\boldsymbol{X}$ the Cartesian product of possibly non-convex, real, closed, nonempty sets.

- The corresponding augmented Lagrangian function is

$$L(\boldsymbol{x},\ \boldsymbol{\lambda})_{\boldsymbol{x} \in \boldsymbol{X}} = \ f(\boldsymbol{x}) - \boldsymbol{\lambda}^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) + \ \frac{\rho}{2}||\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}||_2^2$$

where $\boldsymbol{\lambda}$ is the Lagrangian multipliers or dual variables, and $\rho > 0$ is the step size.

Two-block ADMM with separable variables

- Consider the following optimization problem

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) + g(\boldsymbol{s}) \\ \text{s.t.} \quad & \boldsymbol{A}\,\boldsymbol{x} + \boldsymbol{B}\boldsymbol{s} = \boldsymbol{b} \\ & \boldsymbol{x} \in \boldsymbol{X},\ \boldsymbol{s} \in \boldsymbol{S} \end{aligned}$$

- The corresponding augmented Lagrangian function is

$$L(\boldsymbol{x},\ \boldsymbol{s},\ \boldsymbol{\lambda})_{\boldsymbol{x}\in\boldsymbol{X},\,\boldsymbol{s}\in\boldsymbol{S}} = f(\boldsymbol{x}) + g(\boldsymbol{s}) - \boldsymbol{\lambda}^T(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{s} - \boldsymbol{b}) + \frac{\rho}{2}||\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{s} - \boldsymbol{b}||_2^2$$

Two-block ADMM with separable variables

- Two-block ADMM updates as follows

$$\text{ADMM} = \begin{cases} x^{k+1} & = \arg\min_{x \in \mathcal{X}} \mathcal{L}_\beta(x, s^k; \lambda^k) \\ s^{k+1} & = \arg\min_{s \in \mathcal{X}} \mathcal{L}_\beta(x^{k+1}, s^{k+1}; \lambda^k) \\ \lambda^{k+1} & = \lambda^k - \beta(A[x; s]^{k+1} - b) \end{cases}$$

- The two-block ADMM with separable objective is guarantee to converge.

Multi-Block Cyclic ADMM algorithm

- We could also partition the variables into multiple blocks. Let $\boldsymbol{x} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_b]$

- Direct extension of multi-block (cyclic) ADMM updates as follows

$$\text{ADMM} = \begin{cases} x_1^{k+1} & = \arg\min_{x_1 \in \mathcal{X}} \mathcal{L}_\beta(x_1, \ldots, x_b^k; \lambda^k) \\ \ldots & \\ x_b^{k+1} & = \arg\min_{x_b \in \mathcal{X}} \mathcal{L}_\beta(x_1^{k+1}, \ldots, x_b; \lambda^k) \\ & \\ \lambda^{k+1} & = \lambda^k - \beta(Ax^{k+1} - b) \end{cases}$$

Direct extension of three-block ADMM does not converge

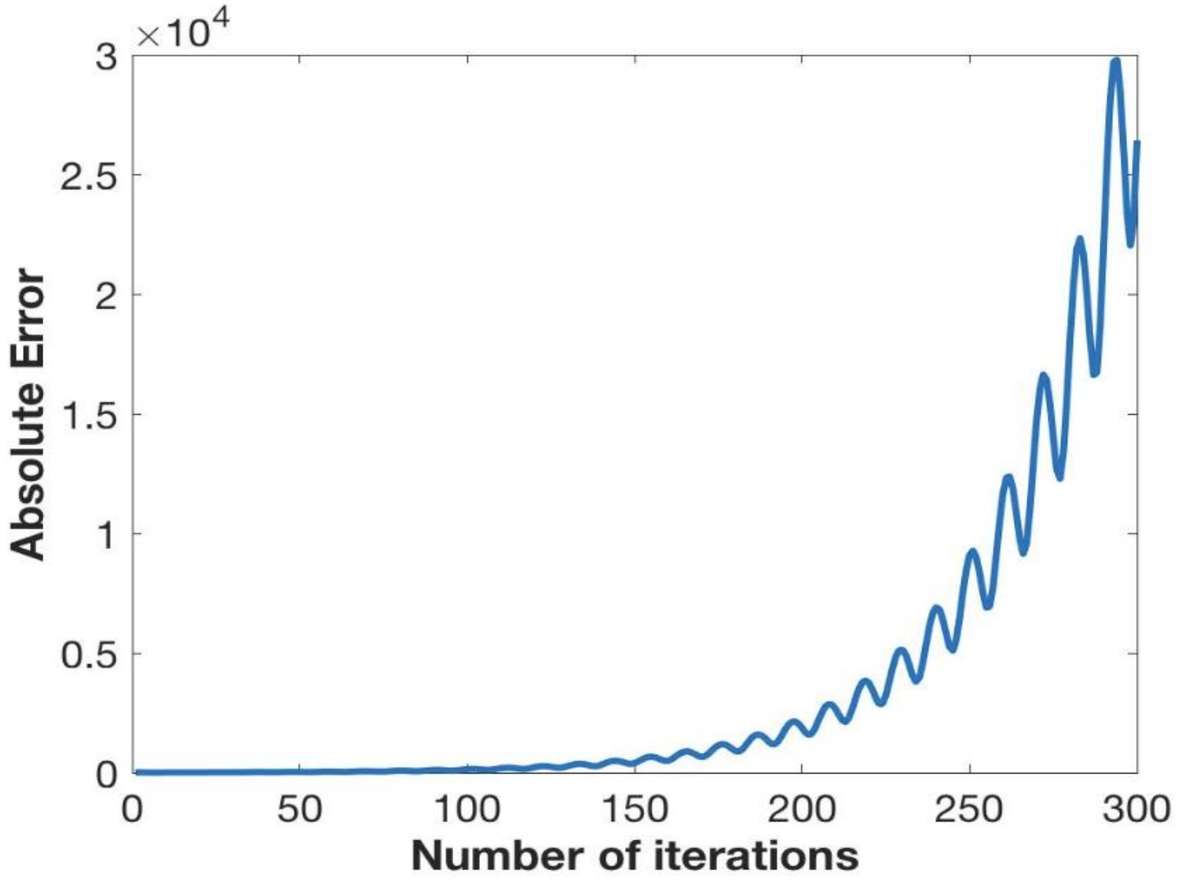


Figure 1: Non-singular system of square equations, three blocks (Chen et al. 2016)

Consensus or Variable-Splitting ADMM I

- Consensus ADMM introduces **auxiliaries** to each block of variables.

- Consider the following optimization problem with **separable** objective

$$\min \ \Sigma_{i=1}^{b} f_i(\boldsymbol{x}_i)$$
$$\text{s.t.} \quad \Sigma_{i=1}^{b} \boldsymbol{A}_i \boldsymbol{x}_i = \boldsymbol{b}$$
$$\boldsymbol{x} \in \boldsymbol{X}$$

- Primal Consensus ADMM reformulates the problem as

$$\min \ \Sigma_{i=1}^{b} f_i(\boldsymbol{x})$$
$$\text{s.t.} \quad \Sigma_{i=1}^{b} \boldsymbol{y}_i = \boldsymbol{b}$$
$$\boldsymbol{A}_i \boldsymbol{x}_i - \boldsymbol{y}_i = 0 \quad \forall i$$
$$\boldsymbol{x} \in \boldsymbol{X}$$

Consensus or Variable-Splitting ADMM II

- In each cycle of ADMM
  - Update all $\boldsymbol{x}_i$ (independently) as one block
  - Update all y_i together as one block with a closed-form solution
  - Update the multipliers the same way

- Consensus ADMM is **guaranteed to converge** with any fixed step-size.

- However, it suffers from **slow** convergence.

Double-Sweep ADMM

In each cycle of ADMM

- Sequentially update $\boldsymbol{x}_1$,…, $\boldsymbol{x}_b$ followed by updating $\boldsymbol{x}_{b-1}$,…, $\boldsymbol{x}_1$ in the **reversed order**.

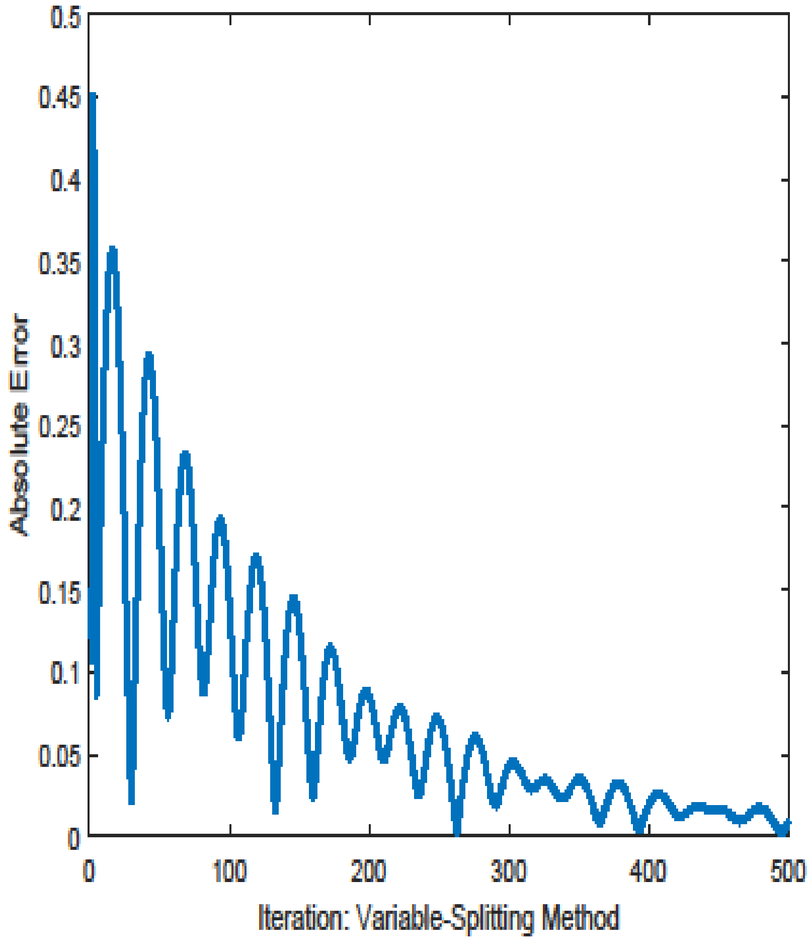
- Update the multipliers the same way.

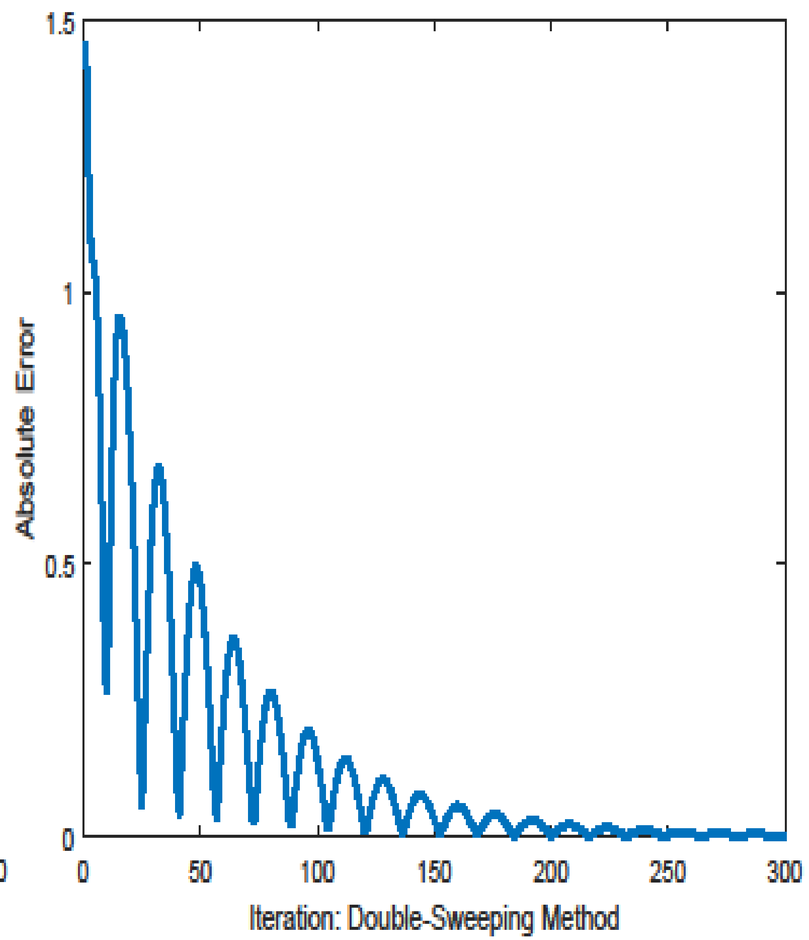
Randomly Permuted - ADMM (RP - ADMM) (Sun et al. 2016)

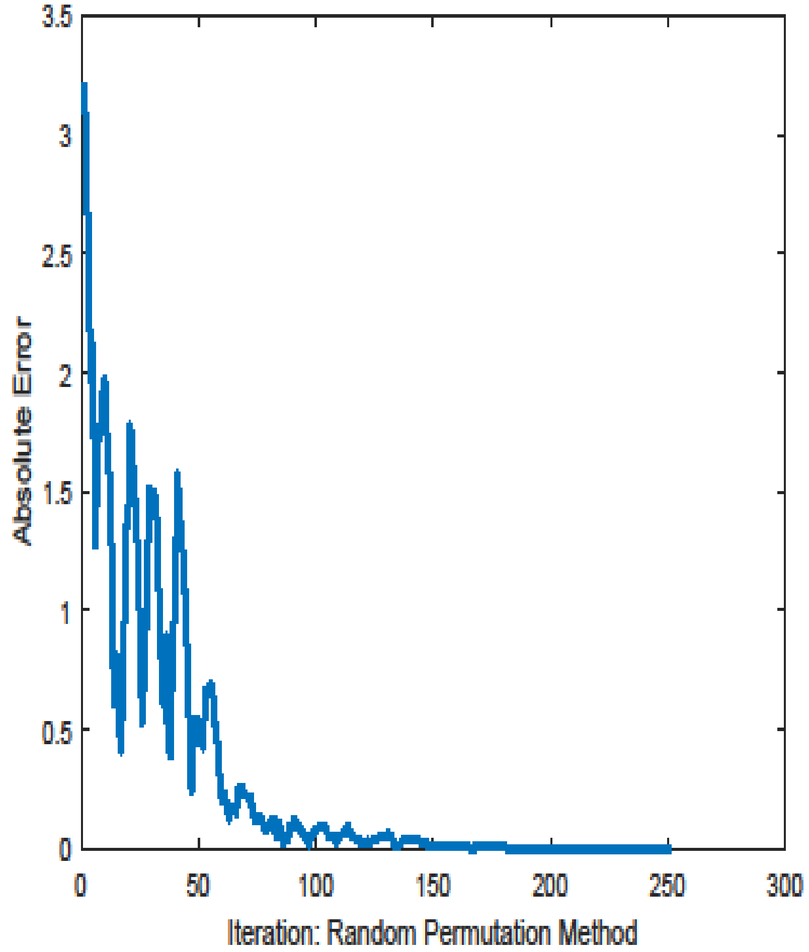
In each cycle of ADMM

- **Randomly generate a permutation** of 1,…,b; and following the permutation order to update $\boldsymbol{x}_i$ sequentially.
- Update the multipliers the same way.
- RP - ADMM is guaranteed to converge in expectation.

Performance on the diverging example







ABIP(Lin et al., 2021)

- An ADMM based interior point method solver for LP problems
- The primal-dual pair of LP:

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

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

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

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 – 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

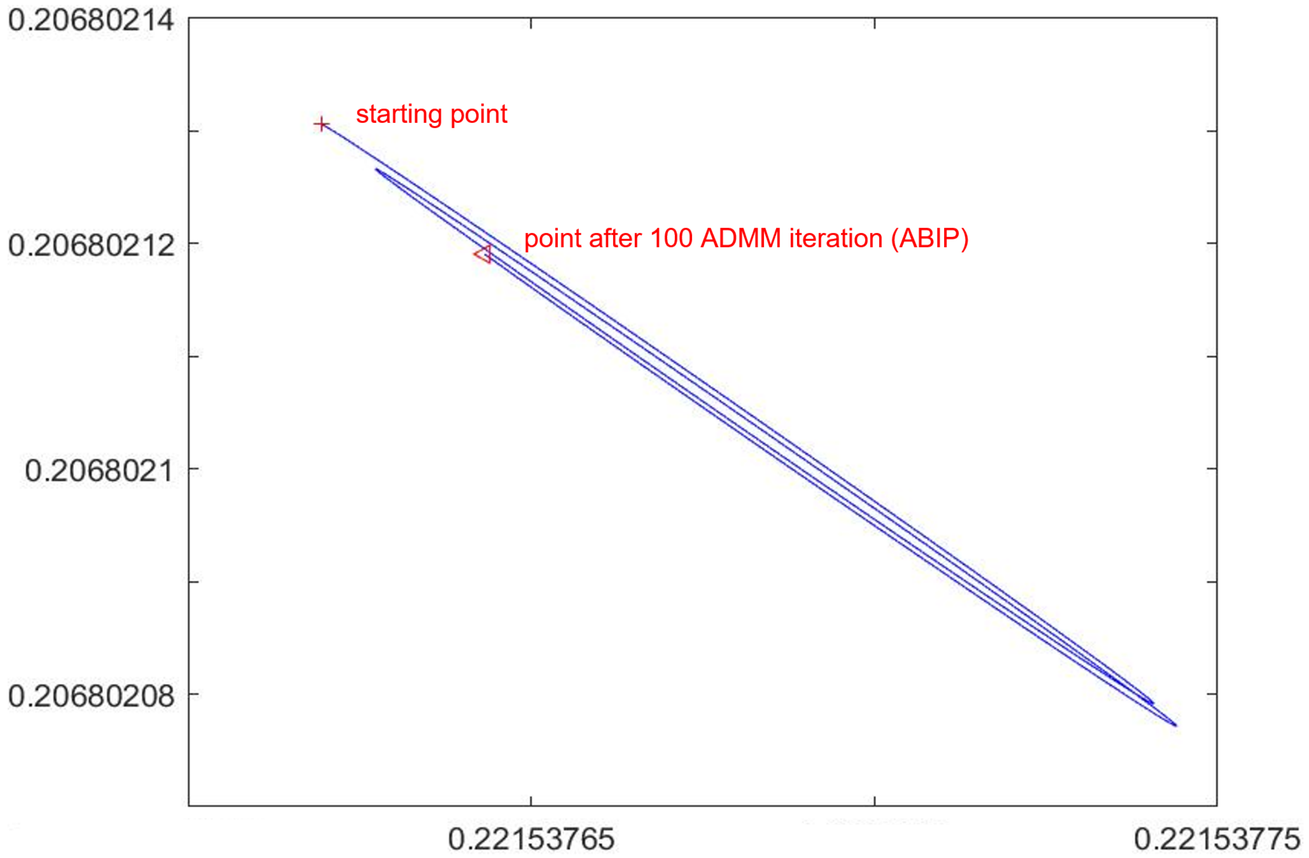
$$M \geq TH \quad \text{and} \quad M_k \bmod F = 0$$

  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

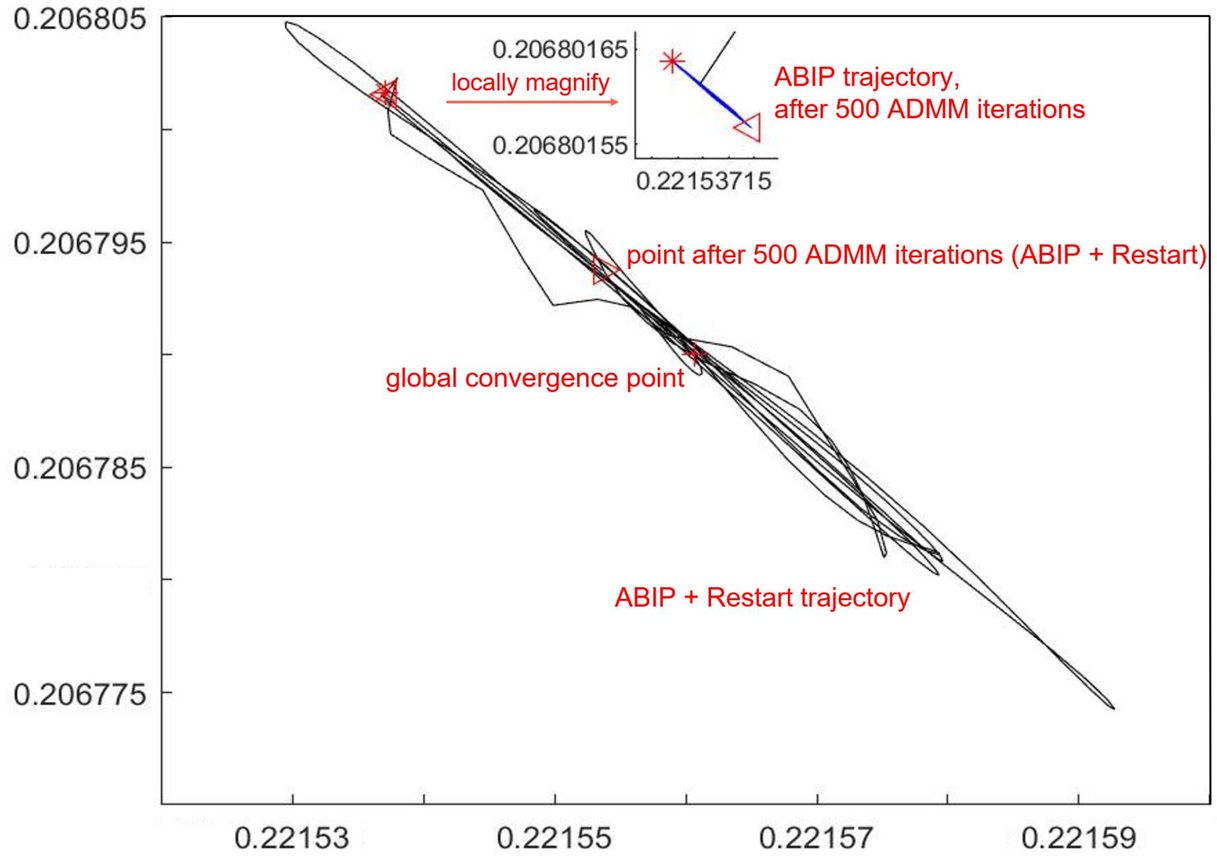
- ABIP tends to induce a spiral trajectory



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 – 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

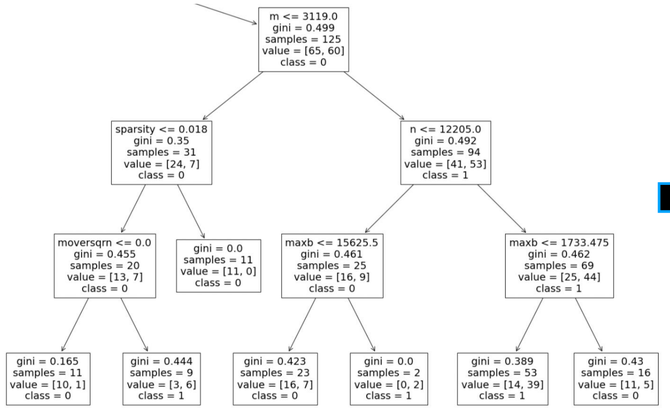
Sparsity

Constraint

Coefficient

Null Objective

…



Strategy 1 (restart)

Strategy 2 (scaling)

…

Strategy *k* (*μ reduction*)

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

- **For generalization limit the number of strategies (2 or 3 types)**

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 (=ABIP+)	**86**	**22**	**73738**	**14.97**

- Hybrid μ : If $\mu > \epsilon$ use the aggressive strategy, otherwise use another strategy
- ABIP+ 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
ABIP+	**205**	**17.0**

- 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.
- SGM stands for Shifted Geometric Mean, a standard measurement of solvers' performance

ABIP – Mittlemann LP barrier dataset

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

Method	# Solved	SGM
COPT	**48**	**1**
PDLP(Julia)	25	**32.42**
ABIP	22	57.40
ABIP+	**32**	32.59

- ABIP+ is slightly inferior to PDLP in terms of the computational time, but it solves more instances.

ABIP – PageRank

- 117 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)	**117**	**1**
ABIP+	114	1.28

- Examples:

Instance	# nodes	PDLP (Julia)	ABIP+
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

10^4

Staircase matrix instance (# nodes = 10)

- In this case, ABIP+ is significantly faster than PDLP!

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

ABIP – SVM

- For 6 large instances from LIBSVM, $\lambda = \epsilon = 10^{-3}$, time limit = 3000s. ABIP and SCS use SOCP formulation, other solvers use QP formulation

data	samples	features	ABIP	OSQP	RACQP	SCS	GUROBI
covtype	581012	54	32.11	33.82	147.62	134.98	1711.41
ijcnn1	49990	22	2.98	4.49	7.59	6.45	13.79
news20	19996	1355191	2279.93	timeout	914.90	timeout	147.70
rcv1	20242	44504	195.63	2256.39	76.73	2352.65	138.97
realsim	72309	20958	518.78	timeout	101.40	timeout	timeout
skin	122536	3	179.33	259.24	39.84	44.06	26.56

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

ABIP – SVM

- For 6 large instances from LIBSVM, $\lambda = \epsilon = 10^{-3}$, time limit = 3000s. We compare the performance of ABIP using both SOCP and QP formulation:

data	samples(m)	features(n)	ABIP-SOCP	ABIP-QP
covtype	581012	54	32.11	timeout
ijcnn1	49990	22	2.98	17.67
news20	19996	1355191	2279.93	735.81
rcv1	20242	44504	195.63	122.26
realsim	72309	20958	518.78	539.20
skin	122536	3	179.33	442.36

- ABIP-QP performs better when feature dimensionality (n) is larger than sample size (m); ABBIP-SOCP performs better for low dimensional problem, i.e. $m \gg n$;

ABIP – SVM

- For ABIP, we prefer QP formulation when m < n and SOCP when m > n; For SCS, we report the best performance between SOCP and QP formulations; all the other solvers are based on QP formulation

data	samples(m)	features(n)	ABIP	OSQP	RACQP	SCS	GUROBI
covtype	581012	54	32.11	33.82	147.62	67.97	1711.41
ijcnn1	49990	22	2.98	4.49	7.59	2.98	13.79
news20	19996	1355191	735.81	timeout	914.90	2877.66	147.70
rcv1	20242	44504	122.26	2256.39	76.73	962.90	138.97
realsim	72309	20958	518.78	timeout	101.40	timeout	timeout
skin	122536	3	179.33	259.24	39.84	44.06	26.56

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

Summary

ABIP is

- **a general purpose LP solver**
- **using ADMM to solve the subproblem**
- **developed with heuristics and intuitions from various strategies**
- **equipped with several new computational tricks**
- **Smart dual updates?**

Today’s Talk

- New developments of ADMM based interior point (ABIP) Method
- HDSDP: Homogeneous dual-scaling SDP solver
- A Dimension Reduced Second-Order Method

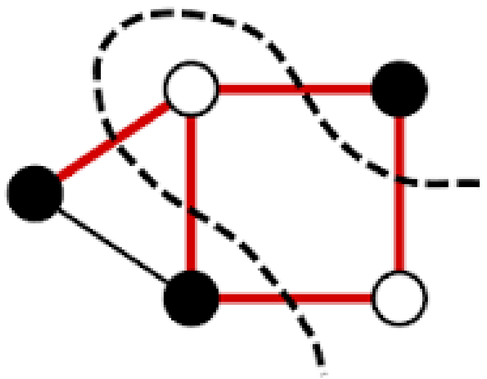
HDSDP: Homogeneous Dual-Scaling SDP solver

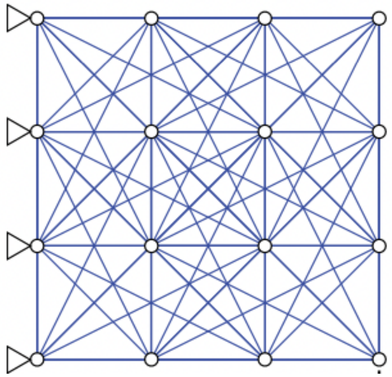
$$\begin{aligned} \min_{X} \quad & \langle C, X \rangle \\ \text{subject to} \quad & \mathcal{A}X = b \\ & X \in \mathbb{S}^n_+ \end{aligned}$$

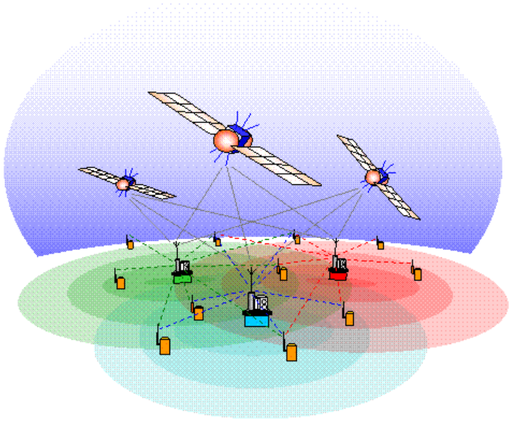
$$\begin{aligned} \mathcal{A}\mathbf{X} - \mathbf{b}\tau &= \mathbf{0} \\ -\mathcal{A}^*\mathbf{y} + \mathbf{C}\tau - \mathbf{S} &= \mathbf{0} \\ \mathbf{b}^\top\mathbf{y} - \langle \mathbf{C}, \mathbf{X} \rangle - \kappa &= 0 \\ \mathbf{X}, \mathbf{S} \succeq 0, \qquad & \kappa, \tau \geq 0 \end{aligned}$$

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

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







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

- **Dynamic system and numerical analysis**

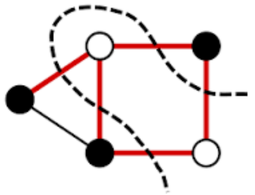
  Lyapunov equation and Linear matrix inequalities

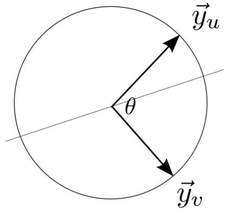
$$AP + PA + I \preceq 0$$
$$P \succeq 0$$

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

- **Combinatorial optimization and relaxation**

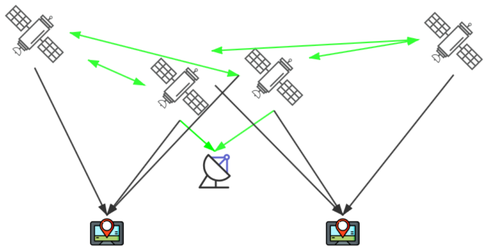
  Well-known 0.878 algorithm for Max-Cut

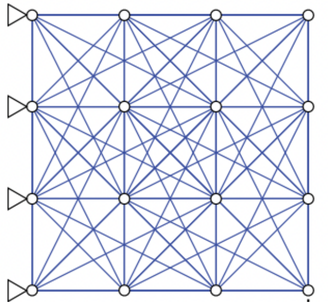




- **Graph realization and distance geometry**

  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

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

We wish to benefit from the desirable properties of HSD!

How to integrate dual-scaling and HSD?

Hint.

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

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

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

Apply the inversion trick!

Homogeneous dual-scaling algorithm

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

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

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

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

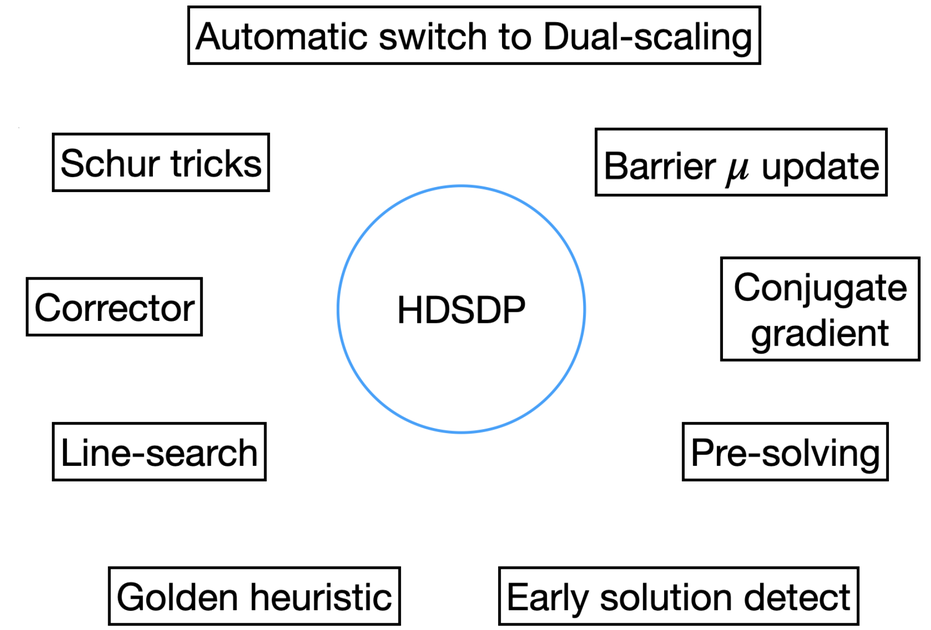
- **Primal iterations can still be fully eliminated**
- $S = -\mathcal{A}^*y + C\tau - R$ **inherits sparsity pattern of data**
  - **Less memory and since X is generally dense**
- **Infeasibility or an early feasible solution can be detected via the embedding**

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 $A_i = \sum_{r=1}^{\mathrm{rank}(A_i)} \lambda_{ir} a_{i,r} a_{i,r}^\top$

To exploit low-rank structure (DSDP)

Tec

1. **S**

2. **C**

Tec

1. **Setup** $\mathbf{S} = \mathbf{a}_{\sigma(i),r}, \ldots, \sigma(i)$.

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

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

To exploit sparsity (SDPA)

Technique M3

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

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

Technique M5

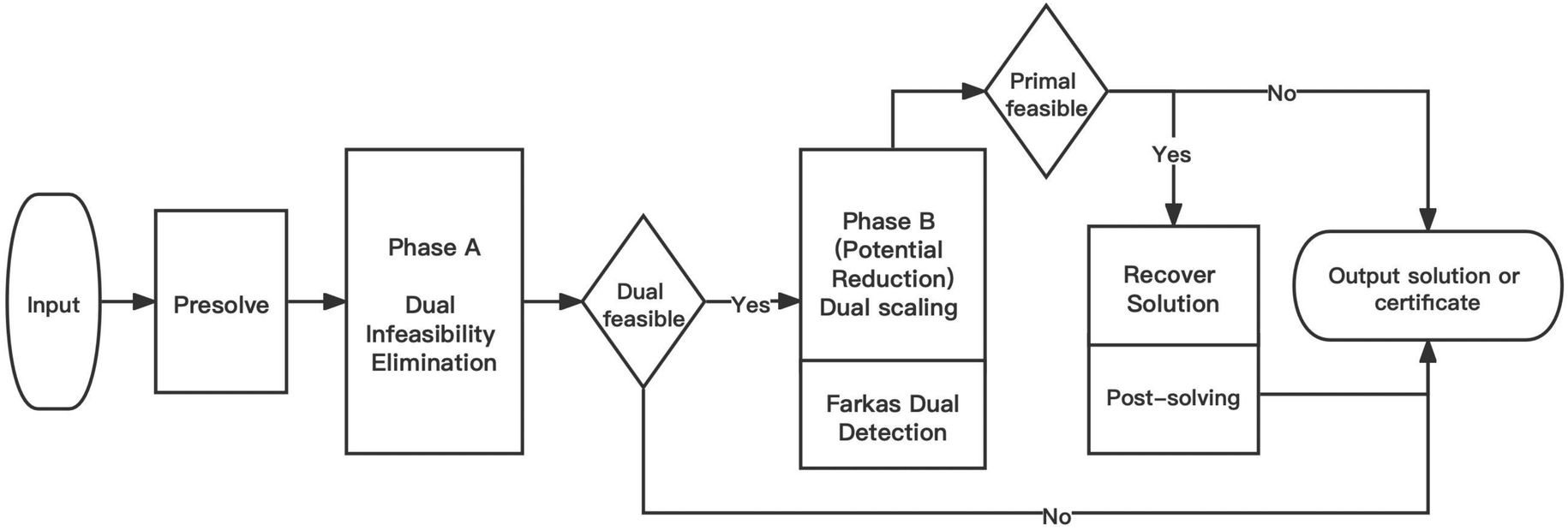
1. **Compute** $M_{\sigma(i)\sigma(j)} = \langle \mathbf{S}^{-1}\mathbf{A}_{\sigma(i)}\mathbf{S}^{-1}, \mathbf{A}_{\sigma(j)}\rangle, \forall j \geq i$ directly.

- **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 $P^{-1/2}MP^{-1/2}x' = b$**

  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

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

$$\begin{aligned} \min_{D,\kappa} \quad & \kappa \\ \text{subject to} \quad & I \preceq DMD \preceq \kappa I \end{aligned}$$

$$\begin{aligned} \min_{D,\kappa} \quad & \kappa \\ \text{subject to} \quad & D \preceq M \\ & \kappa D \succeq M \end{aligned}$$

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

$$\begin{aligned} \max_{\delta,d} \quad & \delta \\ \text{subject to} \quad & D - M \preceq 0 \\ & \delta M - D \preceq 0 \end{aligned}$$

Computational results: optimal diagonal pre-conditioner

- **Test over 491 Suite Sparse Matrices of fewer than 1000 columns**

Reduction	Number
$\geq 80\%$	121
$\geq 50\%$	190
$\geq 20\%$	261

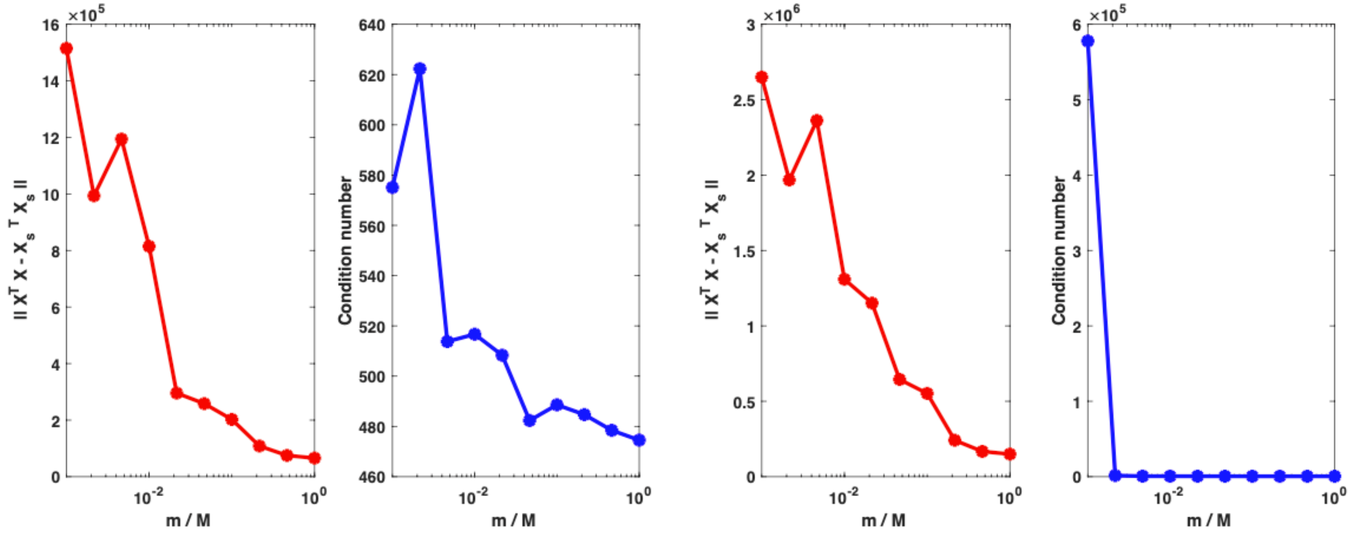
Average reduction	49.7%
Better than diagonal	36.0%
Average time	1.29

- **LIBSVM datasets**

Mat	Size	Cbef	Caft	Reduce
YearPredictionMSD	90	5233000.00	470.20	0.999910
YearPredictionMSD.t	90	5521000.00	359900.00	0.934816
abalone_scale.txt	8	2419.00	2038.00	0.157291
bodyfat_scale.txt	14	1281.00	669.10	0.477475
cadata.txt	8	8982000.00	7632.00	0.999150
cpusmall_scale.txt	12	20000.00	6325.00	0.683813
eunite2001.t	16	52450000.00	8530.00	0.999837
eunite2001.txt	16	67300000.00	3591.00	0.999947
housing_scale.txt	13	153.90	83.22	0.459371
mg_scale.txt	6	10.67	10.03	0.059988
mpg_scale.txt	7	142.50	107.20	0.247842
pyrim_scale.txt	27	49100000.00	3307.00	0.999933
space_ga_scale.txt	6	1061.00	729.60	0.312041
triazines_scale.txt	60	24580000.00	15460000.00	0.371034

Computational results: optimal diagonal pre-conditioner by sampling

- **Many machine learning tasks have $M = A^\top A$, $A \in R^{m \times n}$**
- **More samples than features ($m \gg n$) and M represents covariance**
- **A few samples already suffice to set up $\widehat{M} \approx M$**



- **Using much less computation**
- **Time of setting up $\widehat{M}$ + solving SDP < Time setting up M !**

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**
- **more iterative methods for solving subproblems?**

Today’s Talk

- New developments of ADMM-based interior point (ABIP) Method
- HDSDP: Homogeneous dual-scaling SDP solver
- A Dimension Reduced Second-Order Method

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 Hess** 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 "stepsizes"**

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

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

- **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

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

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

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

- **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? Comput $H^k g^k, H^k d^k$ 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

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

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

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

- **Trust-region:**

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

- **Adaptive, “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 subproblems 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 (Radius-Free) or decrease Δ (trust-region)**
- **Otherwise, decrease λ or increase Δ**

DRSOM: preliminary analyses

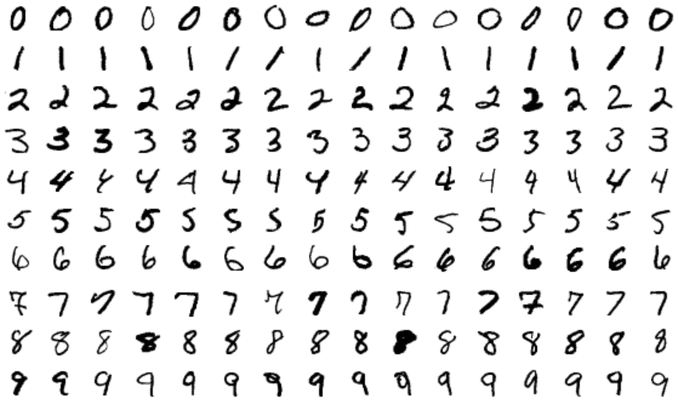
- **Theorem 1. If we apply DRSOM to convex QP, then the iterates are the same as those by Conjugate Gradient Method**

- **Theorem 2. For *f* with second-order Lipschitz condition, DRSOM terminates in $O(1/\epsilon^{3/2})$ iterations. Furthermore, the iterate x_k satisfies the first-order condition and the Hessian is positive semi-definite in the subspace spanned by the gradient and momentum.**

- **More theoretical analyses on the way!**

Logistic Regression

- Solve the Multinomial Logistic Regression for the MNIST dataset.
- The MLR is convex, we compare RSOM to SAGA and LBFGS
- RSOM 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 k	DRSOM $\|\nabla f\|$	DRSOM time	AGD k	AGD $\|\nabla f\|$	AGD time	LBFGS k	LBFGS $\|\nabla f\|$	LBFGS time	Newton TR k	Newton TR $\|\nabla f\|$	Newton TR time
100	10	28	5.8e-07	1.3e+00	58	8.5e-06	4.3e-01	21	8.9e-06	1.4e-01	10	7.1e-07	1.4e-02
100	20	47	6.0e-07	1.0e-03	150	8.2e-06	7.0e-03	35	6.2e-06	2.0e-03	9	4.9e-07	9.0e-03
100	100	98	1.8e-06	1.1e-02	632	1.0e-05	4.6e-01	106	9.8e-06	7.3e-02	47	9.9e-07	7.3e+00
200	10	24	1.3e-06	1.0e-03	37	7.8e-06	1.0e-03	18	1.4e-06	1.0e-03	13	5.9e-10	4.0e-03
200	20	47	9.3e-07	2.0e-03	115	9.4e-06	2.9e-02	33	6.2e-06	2.0e-03	17	6.7e-06	5.2e-02
200	100	107	4.3e-06	1.5e-02	814	9.9e-06	9.3e-01	85	6.2e-06	1.1e-01	36	1.1e-07	7.6e+00
1000	10	25	4.2e-06	3.0e-03	97	9.0e-06	3.6e-02	18	2.2e-06	5.0e-03	16	3.2e-07	5.4e-02
1000	20	27	5.8e-06	3.0e-03	68	7.6e-06	3.4e-02	27	4.5e-06	4.7e-02	13	7.8e-06	1.6e-01
1000	100	76	1.7e-05	2.6e-02	408	1.4e-05	2.6e+00	73	6.4e-06	6.1e-01	32	8.3e-07	1.3e+01

Iterations needed to reach ε = 10e-6

- **Compare DRSOM to Accelerated Gradient Descend (AGD), LBFGS, and Newton Trust-region**
- **DRSOM is comparable to full-dimensional SOM in iteration number**
- **DRSOM is much better in computation time !**

Sensor Network Location (SNL)

- Consider Sensor Network Location (SNL)

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

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

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

- **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:

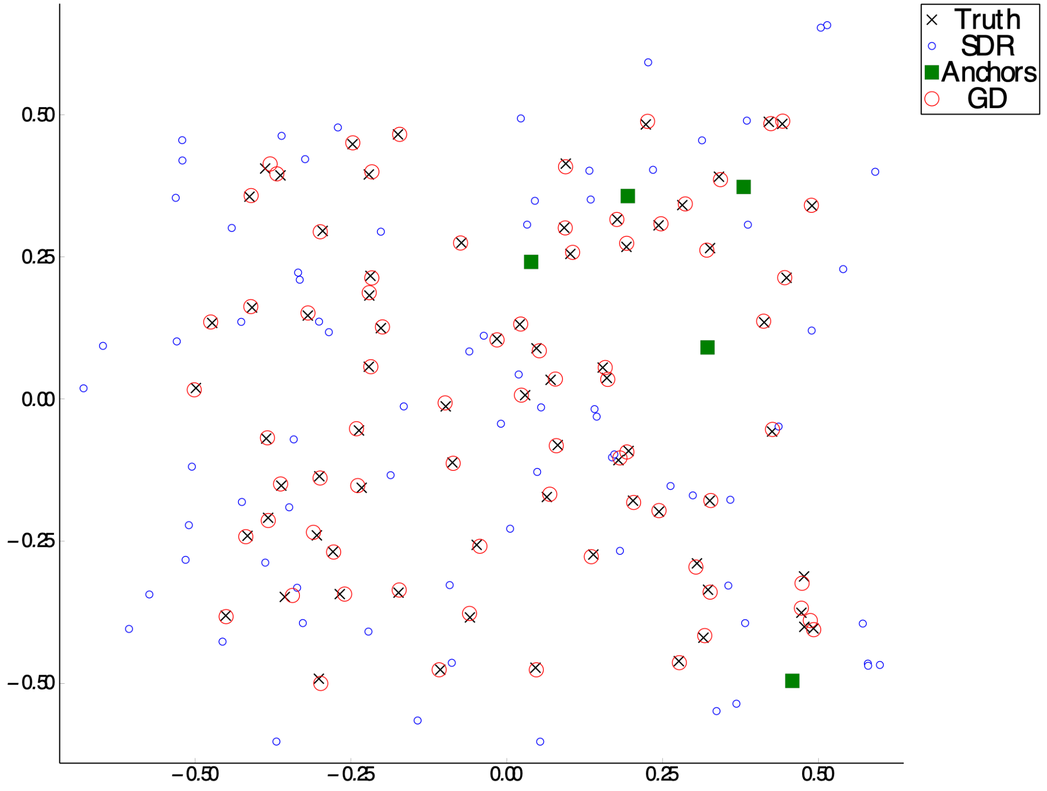
$$
\begin{aligned}
\min \quad & 0 \bullet Z \\
\text{s.t.} \quad & Z_{[1:2,1:2]} = I, \\
& (0; e_i - e_j)\,(0; e_i - e_j)^T \bullet Z = d_{ij}^2 \quad \forall (i,j) \in N_x, \\
& (-a_k; e_i)\,(-a_k; e_i)^T \bullet Z = \bar{d}_{ik}^2 \quad \forall (i,k) \in N_a \\
& Z \succeq 0.
\end{aligned}
$$

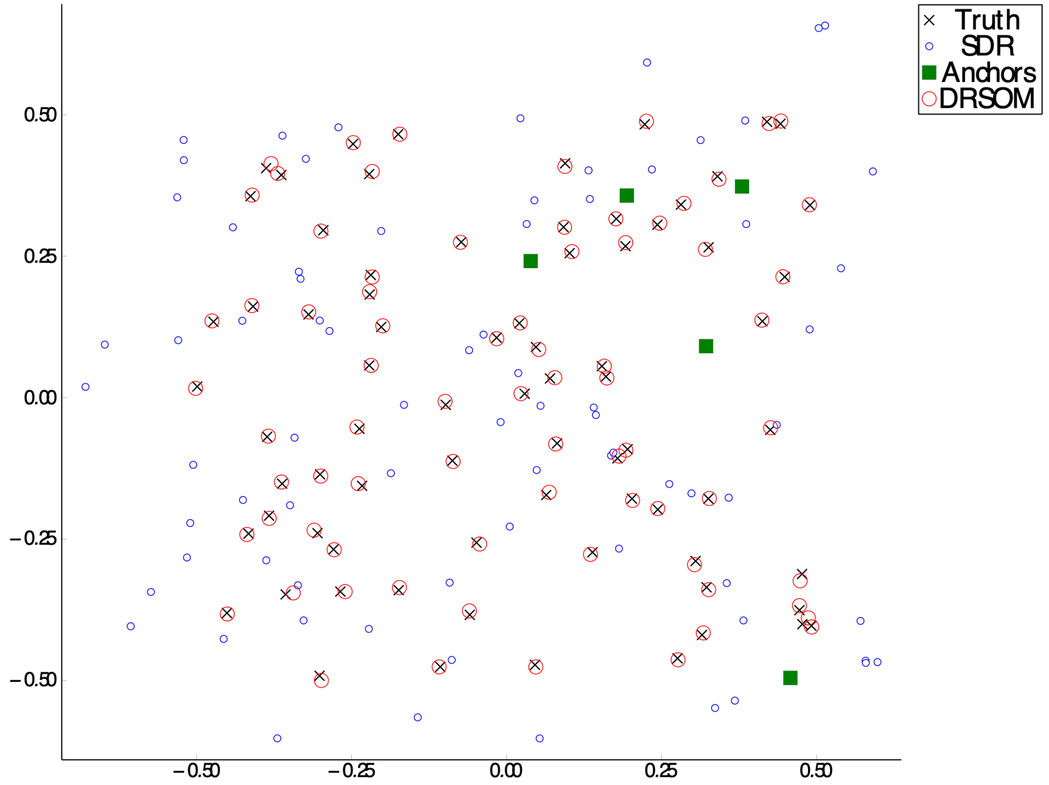
where $Z = \begin{bmatrix} I & X \\ X^T & Y \end{bmatrix}$

- **If rank(Z) = 2, SDP relaxation is exact.**
- **Otherwise, relaxed solution Z* can be used to initialize the NLS**

Sensor Network Location (SNL)

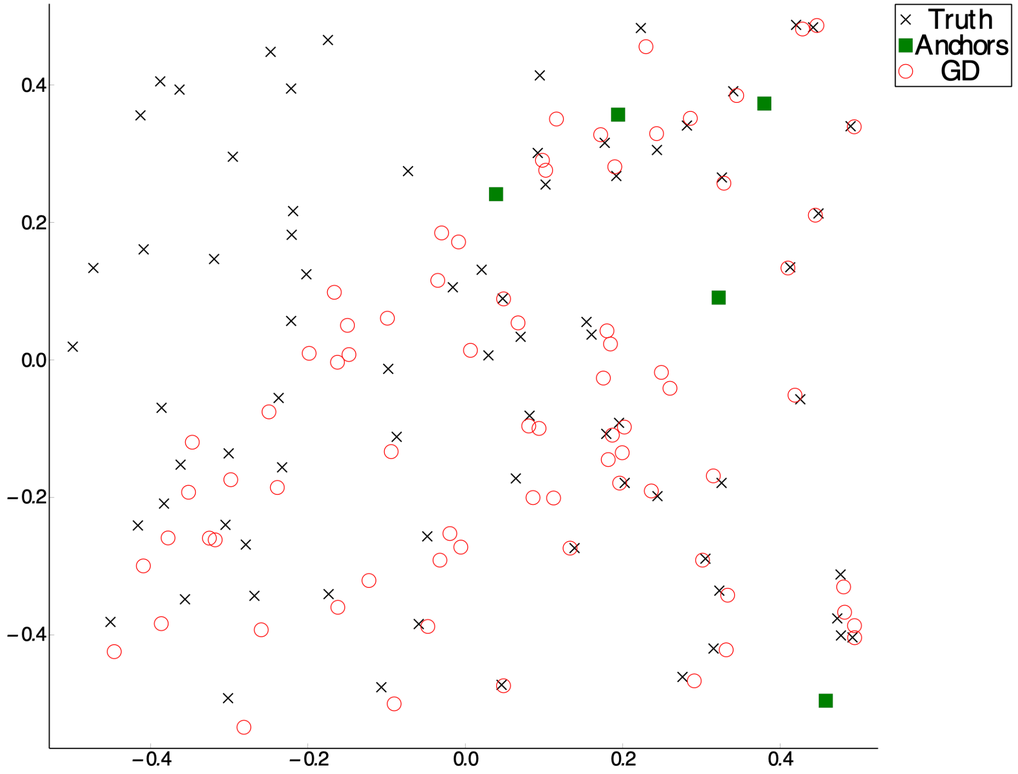
- Graphical results using SDP relaxation to initialize the NLS
- **n = 80, m = 5 (anchors), radio range = 0.5, degree = 25, noise factor = 0.05**
- **Both Gradient Descend and DRSOM can find good solutions !**

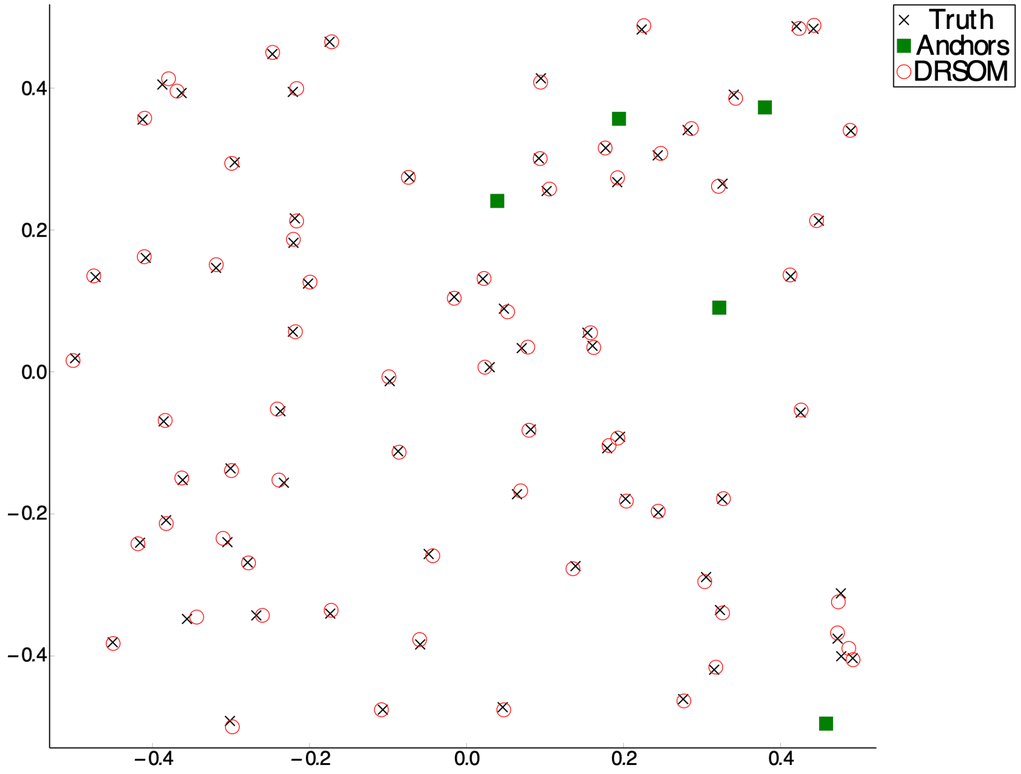




Sensor Network Location (SNL)

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

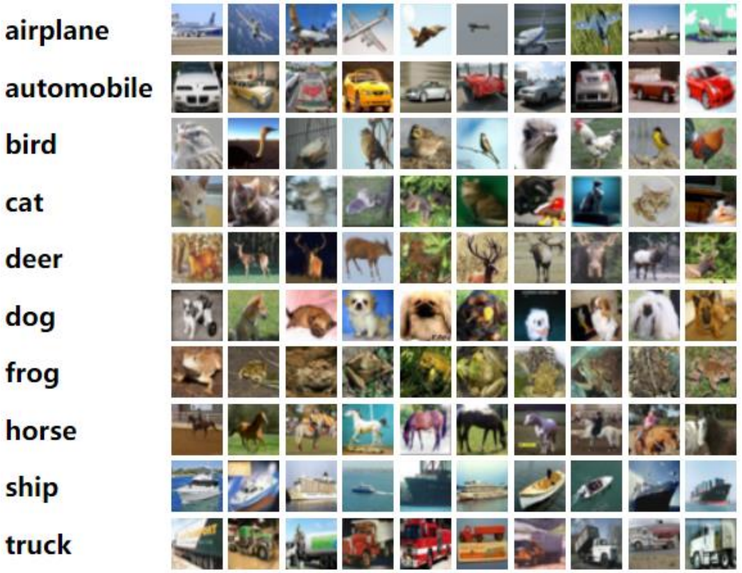




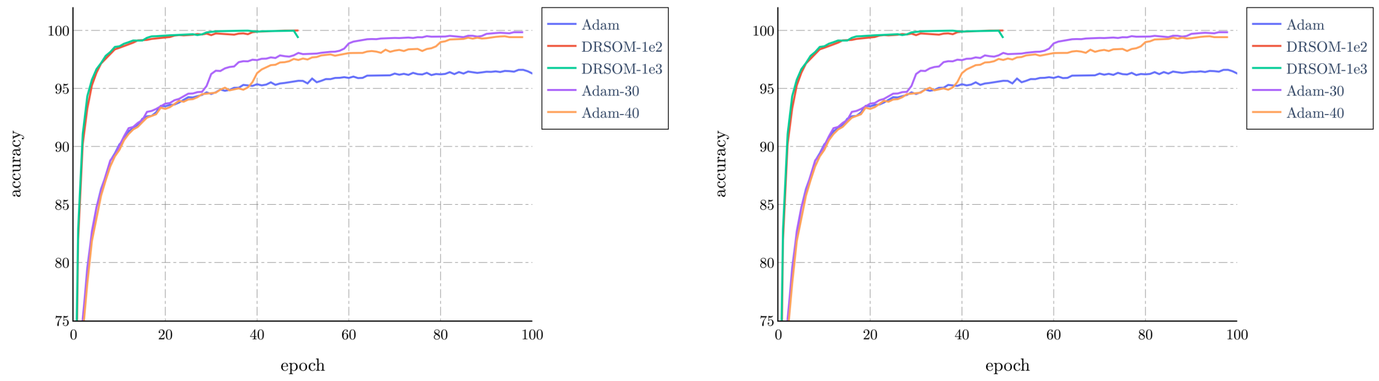
Neural Networks and Deep Learning

To use DRSOM in machine learning problems

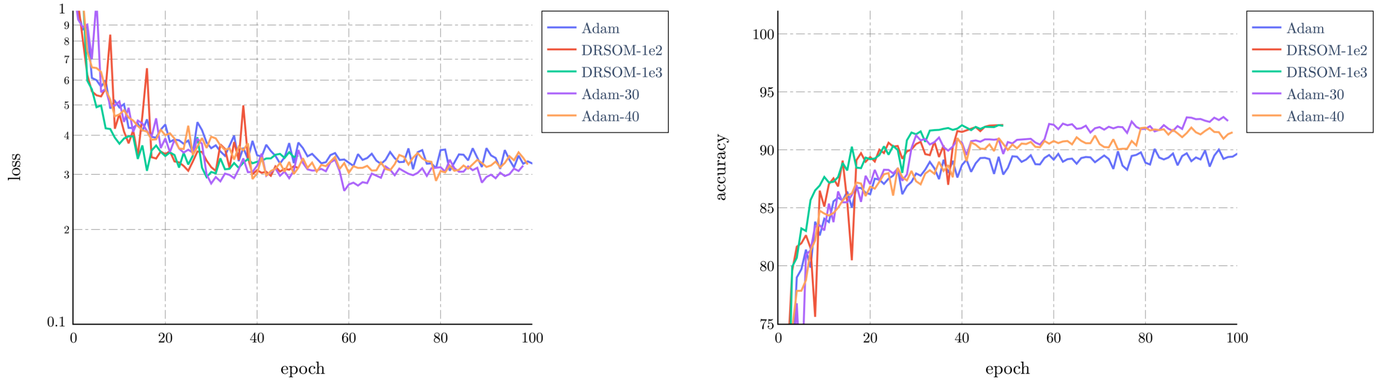
- **We apply the mini-batch strategy to a vanilla DRSOM**
- **Use Automatic Differentiation to compute gradients**
- **Train ResNet18 Model with CIFAR 10**
- **Set Adam with initial learning rate 1e-3**



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 than Adam to run same number of epoch**

Huge potential to be a standard optimizer for deep learning!

Summary

Dimension Reduced Second-order Method:

- **Fast convergence in convex/nonconvex problems**
- **Comparable performance to SOM**
- **Possibly better solutions than FOM in nonconvex problems**
- **Huge potential for Deep Learning tasks**