DRSOM: A Dimension-Reduced Second-Order Method for Machine and Deep Learning

DISTINGUISHED LECTURES SEMINAR @ HKUST

SEPTEMBER 29, 2022

Yinyu Ye (joint work with many…)
Stanford University and CUHKSZ
(Currently Visiting CUHK and HK PolyU)

Today’s Talk

(1) **Motivation and Literature Review**

(2) The Algorithm and Preliminary Convergence Analyses

(3) Computational Experiments

Part (1)

Motivation and Literature Review

Early Complexity Analyses for Nonconvex Optimization

$\min f(x), x \in X \text{ } in \text{ } \mathbb{R}^n,$

- where f is nonconvex and twice-differentiable,

  $g_k = \nabla f(x_k), H_k = \nabla^2 f(x_k)$

- Goal: find x_k such that:

  $\| \nabla f(x_k) \| \leq \epsilon$ (primary, first-order condition)

  $\lambda_{min}(H_k) \geq -\sqrt{\epsilon}$ (in active subspace, secondary, second-order condition)

- For the ball-constrained nonconvex QP: $\min \text{ } \mathrm{cTx} + 0.5\mathrm{xTQx} \text{ } s.t. \| \mathrm{x} \|_2 \leq 1$

  O($\log\log(\epsilon^{-1})$); see Y (1989,93), Vavasis&Zippel (1990)

- For nonconvex QP with polyhedral constraints: O(ϵ^{-1}); see Y (1998), Vavasis (2001)

Standard methods for general nonconvex optimization I

First-order Method (FOM): Gradient-Type Methods

- Assume f has L-Lipschitz cont. gradient
- Global convergence by, e.g., linear-search (LS)
- No guarantee for the second-order condition
- Worst-case complexity, $O(\epsilon^{-2})$; see the textbook by Nesterov (2004)

Each iteration requires $O(n^2)$ operations

Standard methods for general nonconvex optimization II

Second-order Method (SOM): Hessian-Type Methods

- Assume f has M-Lipschitz cont. Hessian
- Global convergence by, e.g., linear-search (LS), Trust-region (TR), or Cubic Regularization
- Convergence to second-order points
- No better than $O(\epsilon^{-2})$, for traditional methods (steepest descent and Newton); according to Cartis et al. (2010) .

Each iteration requires $O(n^3)$ operations

Analyses of SOM for general nonconvex optimization since 2000

Variants of SOM

- Trust-region with the fixed-radius strategy, $O(\epsilon^{-3/2})$, see the lecture notes by Y since 2005
- Cubic regularization, $O(\epsilon^{-3/2})$, see Nesterov and Polyak (2006), Cartis, Gould, and Toint (2011)
- A new trust-region framework, $O(\epsilon^{-3/2})$, Curtis, Robinson, and Samadi (2017)

With "slight" modification, complexity of SOM reduces from $O(\epsilon^{-2})$ to $O(\epsilon^{-3/2})$

Other complexity analyses for some structural nonconvex optimization

- Ge, Jiang, and Y (2011), $O(\epsilon^{-1}\log(1/\epsilon))$, for L_p minimization arisen from Comp. Sensing.
- Bian, Chen, and Y (2015), $O(\epsilon^{-3/2})$, for certain non-Lipschitz and nonconvex optimization.
- Chen et al. (2014) shows strongly NP-hardness for $L_2 - L_p$ minimization; but later Ge, He, and He (2017) proposes a method with complexity of $O(\log(\epsilon^{-1}))$ to find a local minimum
- Haeser, Liu, and Y (2019) uses the first-order and second-order interior point trust-region method achieving first-order ϵ-KKT points with complexity of $O(\epsilon^{-2})$ and $O(\epsilon^{-3/2})$, respectively.

Recent efforts for general nonconvex optimization

FOM Improvements:

- FOM with Hessian negative curvature (NC) detections, $O(\epsilon^{-7/4}\log(1/\epsilon))$
  - Carmon et al. (2018), with Hessian-vector product (HVP) and Lanczos
  - cost $O(\epsilon^{-1/4})$ for each negative curvature request
  - Also, Carmon et al. (2017), does not require HVP (only first-order condition)
- Agarwal et al. (2016), also $O(\epsilon^{-7/4})$, using accelerated methods for fast approximate matrix inversion

They are hybrid and/or randomized methods and seem difficult to be implemented

Our approach: Reduce dimension in SOM

Part (2)

The Algorithm and Preliminary Convergence Analyses

Motivation from multi-directional FOM

- Two-directional FOM, with d_k being the momentum direction $(x_k - x_{k-1})$

$$x_{k+1} = x_k - \alpha_k^1 \nabla f(x_k) + \alpha_k^2 d_k \; = x_k + d_{k+1}$$

  where step-sizes are constructed; including CG, PT, AGD, Polyak, and many others.

- In SOM, a method typically minimizes a full dimensional quadratic Taylor expansion to obtain direction vector d_{k+1}. For example, one TR step solves for d_{k+1} from

$$\min_d \quad (g_k)^T d + 0.5 d^T H_k d \quad s.t. ||\mathrm{d}||_2 \leq \Delta_k$$

  where Δ_k is the trust-region radius.

- DRSOM: Dimension Reduced Second-Order Method

  Motivation: using few directions in SOM

DRSOM I

- The DRSOM in general uses m-independent directions

$$d(\alpha) := D_k\alpha \, , D_k \in R^{nm}, \ \alpha \in R^m$$

- Plug the expression into the full-dimension TR quadratic minimization problem, we minimize a m-dimension trust-region subproblem to decide "m stepsizes":

$$\min \ m_k^{\alpha}(\alpha) \coloneqq (c_k)^T\alpha + \frac{1}{2}\alpha^T Q_k \alpha$$

$$||\alpha||_{G_k} \le \Delta_k$$

$$G_k = D_k^T D_k, Q_k = D_k^T H_k D_k, c_k = (g_k)^T D_k$$

How to choose D_k? How great would *m* be? Rank of H_k?
(Randomized) rank reduction of a symmetric matrix to log(n) (So et al. 08)?

DRSOM II

- In following, as an example, DRSOM adopts two FOM directions

$$d = -\alpha^1 \nabla f(x_k) + \alpha^2 d_k := d(\alpha)$$

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

- Then we minimize a 2-D trust-region problem to decide “two step-sizes”:

$$\min\ m_k^{\alpha}(\alpha) := f(x_k) + (c_k)^T \alpha + \frac{1}{2}\alpha^T Q_k \alpha$$

$$||\alpha||_{G_k} \le \Delta_k$$

$$G_k = \begin{bmatrix} g_k^T g_k & -g_k^T d_k \\ -g_k^T d_k & d_k^T d_k \end{bmatrix}, Q_k = \begin{bmatrix} g_k^T H_k g_k & -g_k^T H_k d_k \\ -g_k^T H_k d_k & d_k^T H_k d_k \end{bmatrix}, c_k = \begin{bmatrix} -||g_k||^2 \\ g_k^T d_k \end{bmatrix}$$

DRSOM III

DRSOM can be seen as:

- “Adaptive” **Accelerated Gradient Method** (Polyak’s momentum 60)
- A second-order method minimizing quadratic model in the reduced 2-D

  $$m_k(d) = f(x_k) + \nabla f(x_k)^T d + \frac{1}{2} d^T \nabla^2 f(x_k) d, d \in \text{span}\{-g_k, d_k\}$$

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

  $d \in \text{span}\{g_k, [H(x_k)]^{-1} g_k\}$ (e.g., Powell 70)
- A conjugate direction method for convex optimization exploring the **Krylov Subspace** (e.g., Yuan&Stoer 95)
- For convex quadratic programming with no radius limit, terminates in n steps

Computing Hessian-Vector Product in DRSOM is the Key

In the DRSOM with two directions:

$$Q_k = \begin{bmatrix} g_k^T H_k g_k & -g_k^T H_k d_k \\ -g_k^T H_k d_k & d_k^T H_k d_k \end{bmatrix}, c_k = \begin{bmatrix} -||g_k||^2 \\ g_k^T d_k \end{bmatrix}$$

How to cheaply obtain Q? Compute $H_k g_k, H_k d_k$ first.

- Finite difference:

$$H_k \cdot v \approx \frac{1}{\epsilon}[g(x_k + \epsilon \cdot v) - g_k],$$

- Analytic approach to fit modern automatic differentiation,

$$H_k g_k = \nabla(\frac{1}{2} g_k^T g_k), H_k d_k = \nabla(d_k^T g_k),$$

- or use Hessian if readily available !

Subproblem adaptive strategies in DRSOM I

Recall 2-D quadratic model:

$$\min m_k^{\alpha}(\alpha) := f(x_k) + (c_k)^T\alpha + \frac{1}{2}\alpha^T Q_k \alpha$$

$$||\alpha||_{G_k} \leq \Delta_k, G_k = \begin{bmatrix} g_k^T g_k & -g_k^T d_k \\ -g_k^T d_k & d_k^T d_k \end{bmatrix}, Q_k = \begin{bmatrix} g_k^T H_k g_k & -g_k^T H_k d_k \\ -g_k^T H_k d_k & d_k^T H_k d_k \end{bmatrix}, c_k = \begin{bmatrix} -||g_k||^2 \\ g_k^T d_k \end{bmatrix}$$

Apply two strategies that ensure global and convergence

- Trust-region: **Adaptive radius**

$$\min_{\alpha} m_k^{\alpha}(\alpha), \| \alpha \|_{G_k} \leq \Delta_k, G_k = \begin{bmatrix} g_k^T g_k & -g_k^T d_k \\ -g_k^T d_k & d_k^T d_k \end{bmatrix}$$

- Radius-free: Apply **Lagrangian multiplier λ_k**

$$\min_{\alpha} m_k^{\alpha}(\alpha) + \lambda_k \|\alpha\|_{G_k}^2$$

- The subproblems can be solved efficiently.

Subproblem adaptive strategies in DRSOM II

At each iteration k, the DRSOM proceeds:

- Solving 2-D Quadratic trust-region model
- Computing quality of the approximation*

$$\rho^k := \frac{f(x^k) - f(x^k + d^{k+1})}{m_p^k(0) - m_p^k(d^{k+1})} = \frac{f(x^k) - f(x^k + d^{k+1})}{m_\alpha^k(0) - m_\alpha^k(\alpha^k)}$$

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

* Can be further improved by other acceptance criteria, e.g., Curtis et al. 2017

DRSOM: key assumptions and theoretical results (Zhang at al. SHUFE)

Assumption. (a) f has Lipschitz continuous Hessian. (b) DRSOM iterates with a fixed-radius strategy: $\Delta_k = \epsilon/\beta$) c) **If the Lagrangian multiplier $\lambda_k < \sqrt{\epsilon}$, assume $\| (H_k - \widetilde{H}_k)d_{k+1} \| \leq C \| d_{k+1} \|^2$ (Cartis et al.),** where $\widetilde{H}_k$ is the projected Hessian in the subspace (commonly adopted for approximate Hessian)

Theorem 1. If we apply DRSOM to QP, then the algorithms terminates in at most n steps to find a first-order stationary point

Theorem 2. (Global convergence rate) For *f* with second-order Lipschitz condition, DRSOM terminates in $O(\epsilon^{-3/2})$ 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.

Theorem 3. (Local convergence rate) If the iterate x_k converges to a strict local optimum x^* such that $H(x^*) \succ 0$, and if **<u>Assumption (c)</u>** is satisfied as soon as $\lambda_k \leq C_\lambda \| d_{k+1} \|$, then DRSOM has a local superlinear (quadratic) speed of convergence, namely: $\| x_{k+1} - x^* \| = O(\| x_k - x^* \|^2)$

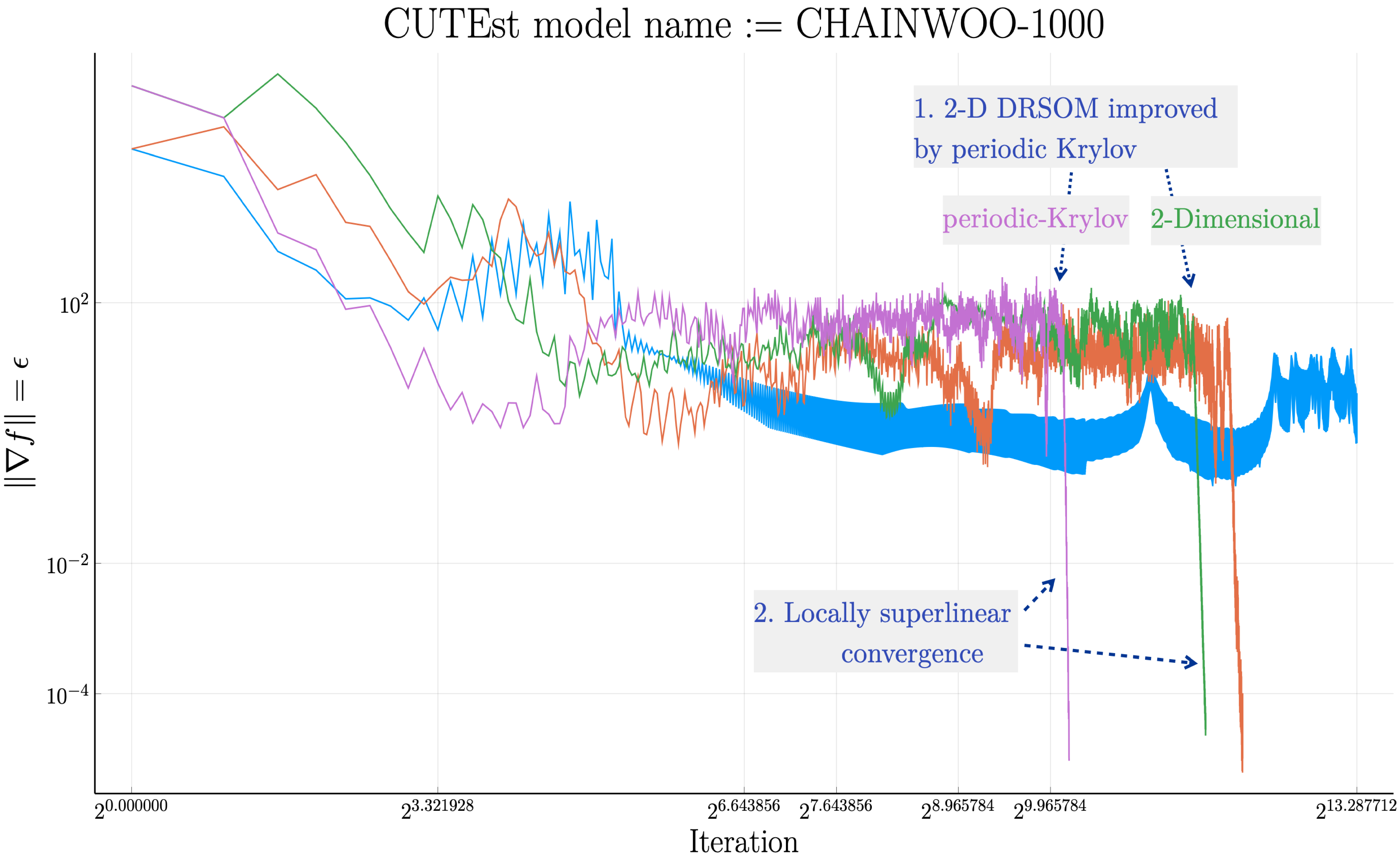
DRSOM: How to remove Assumption (c)?

- Global rate: ensure **Assumption (c)** holds ***periodically*** (whenever needed, e.g., switch to Krylov)
- Local rate: ensure **Assumption (c)** holds around $\boldsymbol{x}^*$,we have the desired results.

Specifically, expand subspace if Assumption (c) does not hold…

- Carmon et al. (2018) find the NC ($O(\epsilon^{-1/4})$ for each step) and proceed
- Run **Lanczos** (worst-case without sparsity $O(n^3)$)
- Trade-off between $O(\epsilon^{-7/4})$ (more dimension-free) and $O(\epsilon^{-3/2})$

DRSOM: convergence behavior, an example



Example from the CUTEst dataset

- *GD* and *LBFGS* both use a Line-search (Hager-Zhang)
- *DRSOM-F (2-D)*: original 2-dimensional version with g_k and d_k
- *DRSOM-F (periodic-Krylov)*, guarantees $\| (H_k - \widetilde{H}_k) d_{k+1} \| \leq C \| d_{k+1} \|^2$ periodically.

Part (3)

Computational Experiments

Nonconvex L_2-L_p Minimization in Compressed Sensing

- 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)

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

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

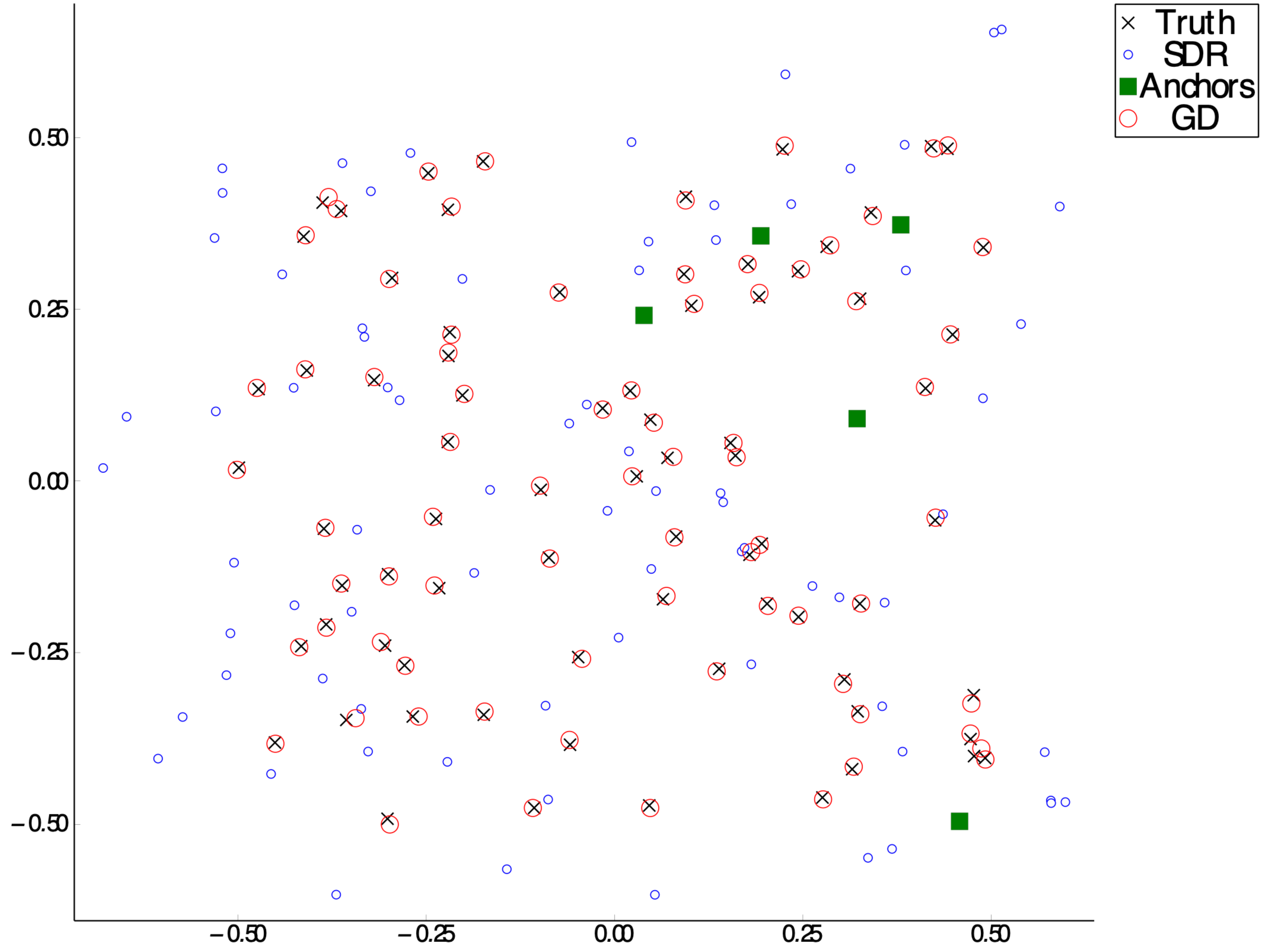
$$\|x_i - x_j\|^2 = d_{ij}^2, \forall (i,j) \in N_x$$
$$\|x_i - a_k\|^2 = \bar{d}_{ik}^2, \forall (i,k) \in N_a$$

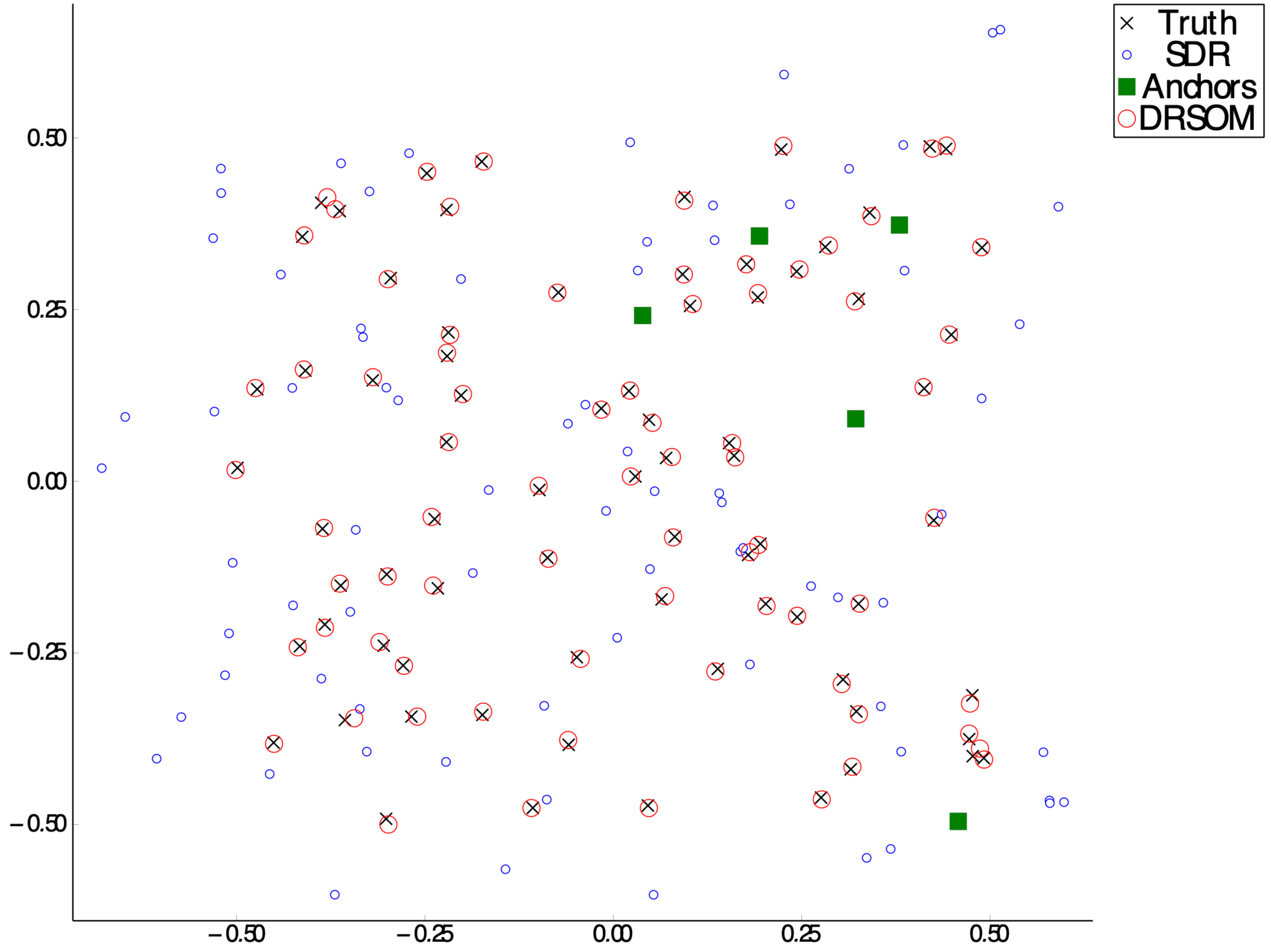
- We can solve SNL by the nonconvex nonlinear least square (NLS) problem

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

Sensor Network Location (SNL)

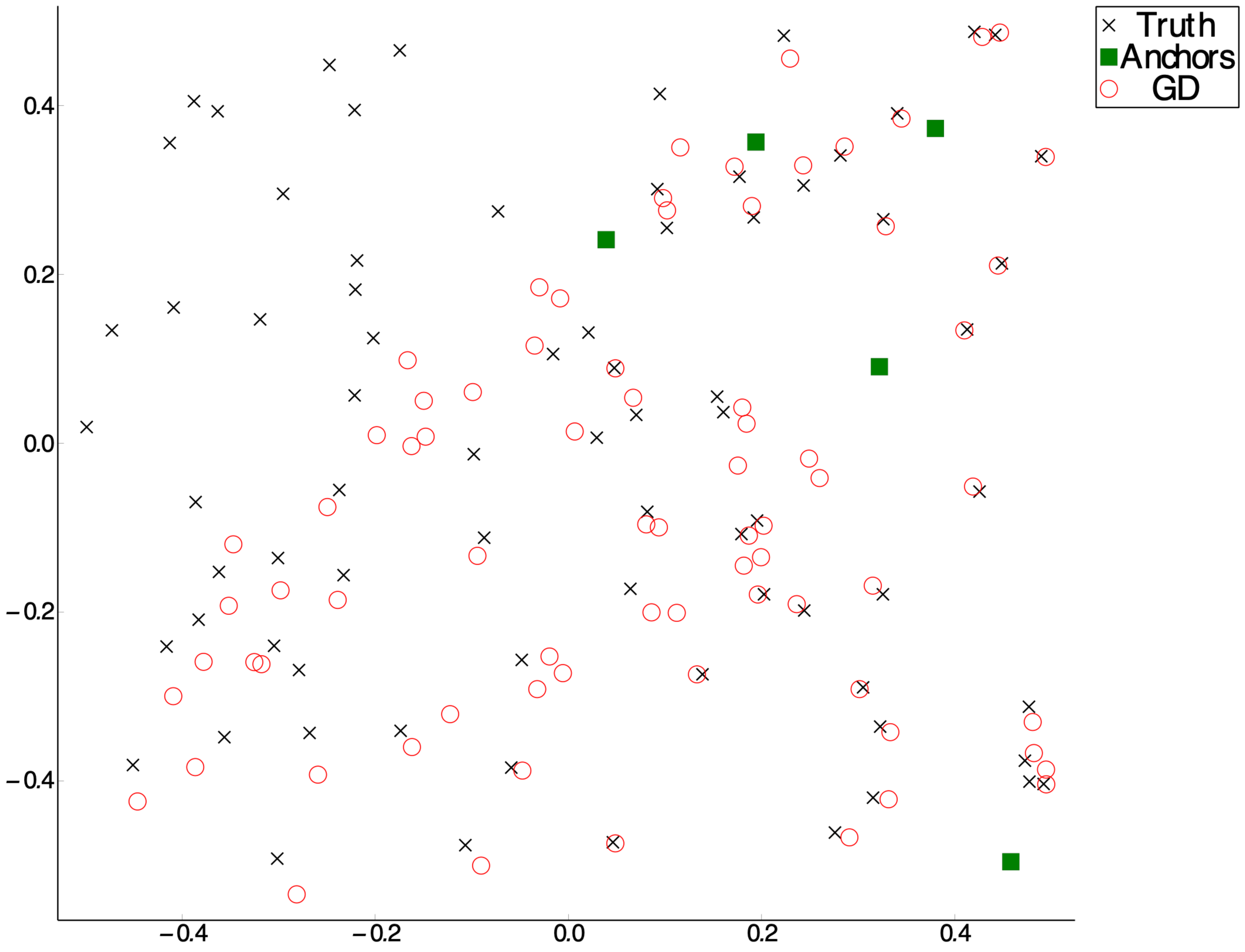
- Graphical results using SDP relaxation to initialize the NLS
- n = 80, m = 5 (anchors), radio range = 0.5, degree = 25, noise factor = 0.05
- Both Gradient Descent and DRSOM can find good solutions !

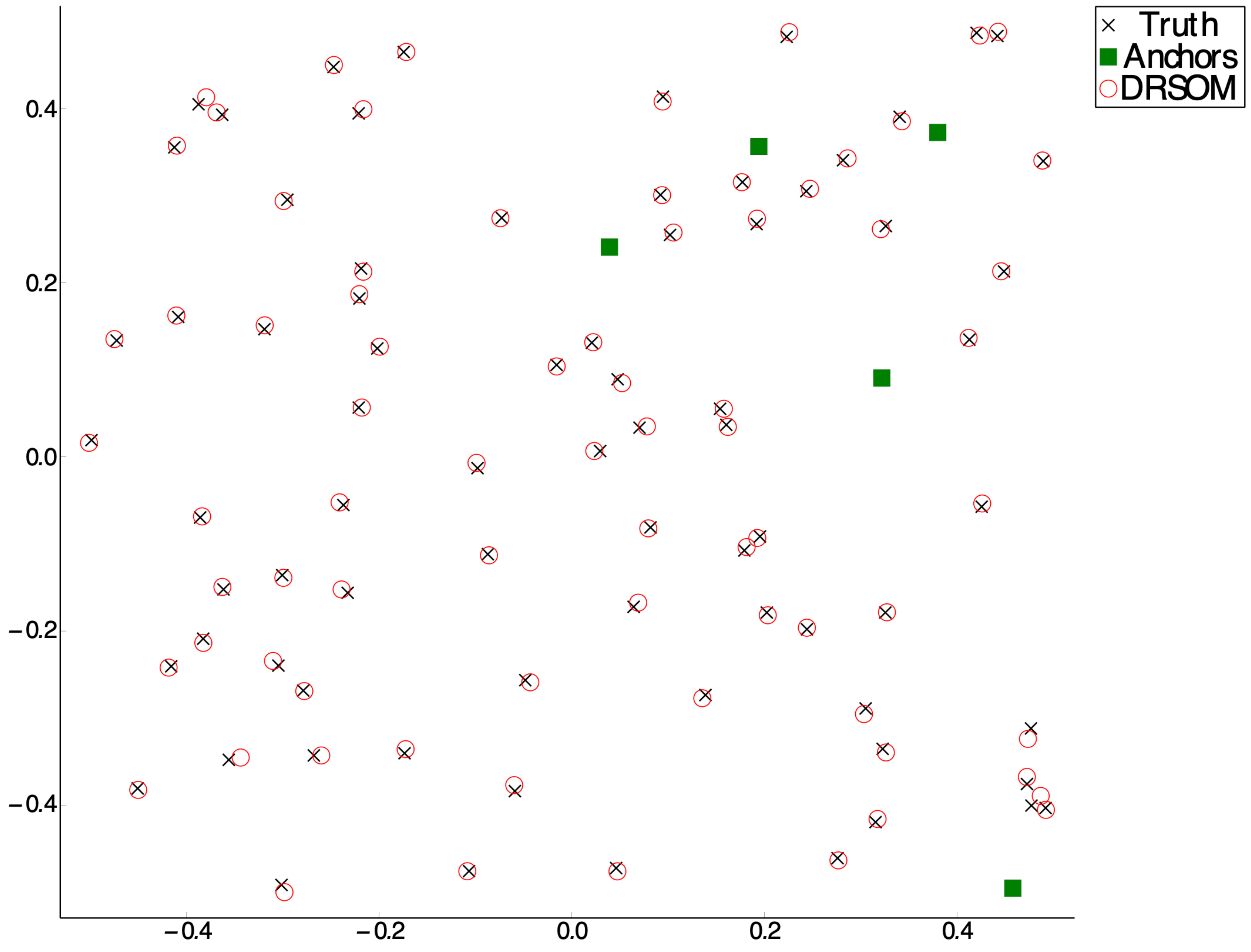




Sensor Network Location (SNL)

- Graphical results without SDP relaxation
- DRSOM can still converge to optimal solutions

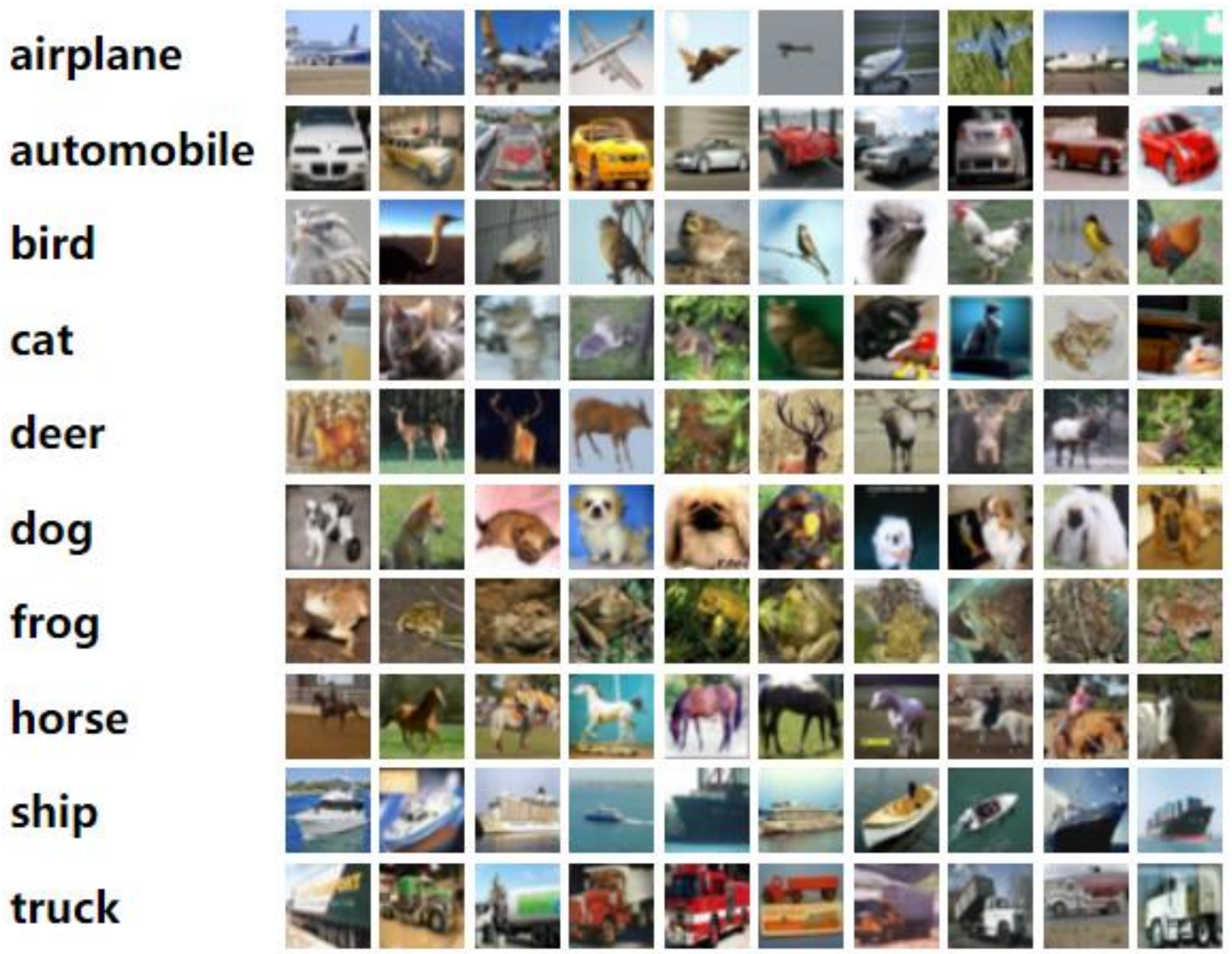




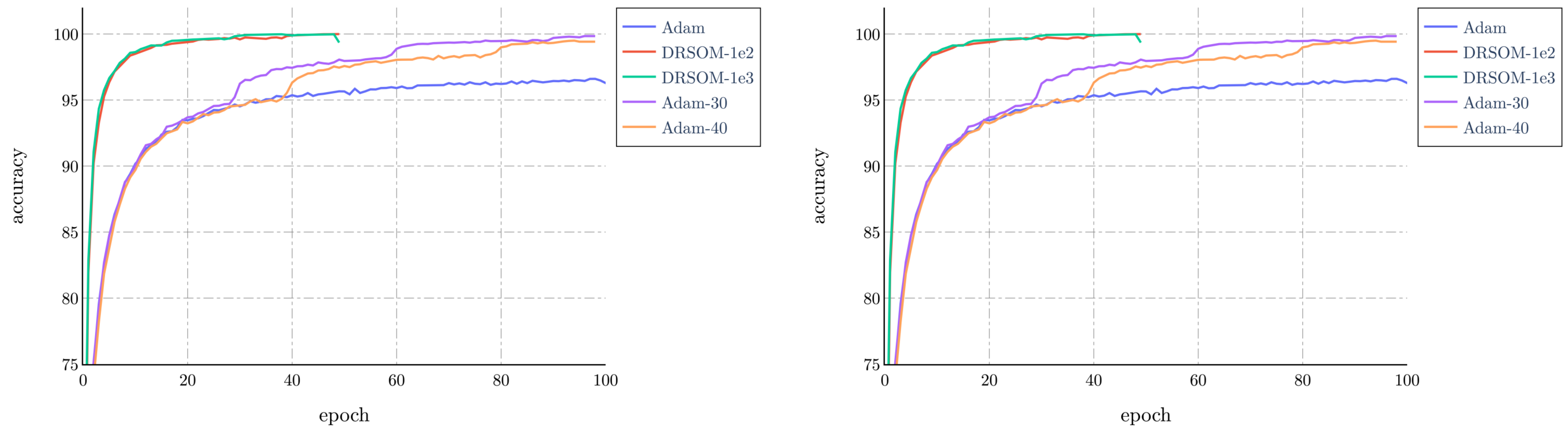
Neural Networks and Deep Learning

To use DRSOM in machine learning problems

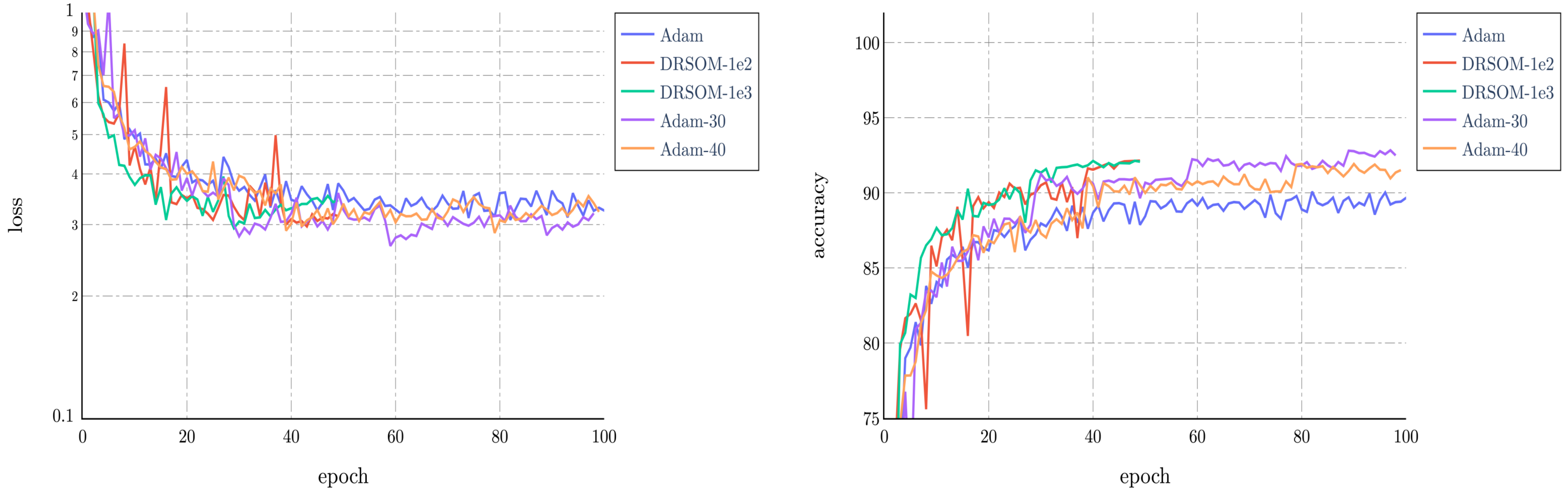
- We apply the mini-batch strategy to a vanilla DRSOM
- Use Automatic Differentiation to compute gradients
- Train ResNet18 Model with CIFAR 10
- Set Adam with initial learning rate 1e-3



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 little tuning

Cons

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

Good potential to be a standard optimizer for deep learning!

Policy Optimization

$$\max_{\theta \in \mathbb{R}^d} J(\theta) := \mathbb{E}_{\tau \sim p(\tau|\theta)}[\mathcal{R}(\tau)] = \int \mathcal{R}(\tau) p(\tau \mid \theta) d\tau$$

- **Vanilla policy gradient**: Apply gradient descent to find the policy that maximizes the expected return:

  $$\theta_{t+1} = \theta_t + \eta_t \hat{\nabla}_\theta J(\theta)$$

  where $\hat{\nabla}_\theta J(\theta)$ is estimated stochastic gradient. Examples include:

  - REINFORCE (Williams, Simple statistical gradient-following algorithms for connectionist reinforcement learning, 1992)

  - PGT (Sutton et al., Policy gradient methods for reinforcement learning with function approximation, 1999)

- **Policy gradient based on KL divergence**

  - Trust Region Policy Optimization (TRPO): Linearize objective function and update parameter under KL constraint (J. Schulman et al. “Trust region policy optimization”, 2015)

  - Proximal Policy Optimization (PPO) : Update the parameter via KL-regularized gradient ascent (J. Schulman et al. “Proximal policy optimization algorithms”, 2017)

  - Mirror descent policy optimization (Tomar et al. 2021, Shani et al. 2020)

- **Many other recent developments**

  - Momentum policy gradient (Feihu Huang et al. 2021), Hessian-aided policy gradient (Zebang Shen et al. 2019), Variance reduced policy gradient (Papini et al. 2018)

DRSOM for Policy Gradient (PG) (Liu et al. SHUFE)

- As mentioned above, the goal is to maximize the expected discounted trajectory reward:

$$\max_{\theta \in \mathbb{R}^d} J(\theta) := \mathbb{E}_{\tau \sim p(\tau|\theta)}[\mathcal{R}(\tau)] = \int \mathcal{R}(\tau) p(\tau \mid \theta) d\tau$$

- The gradient can be estimated by:

$$\hat{\nabla} J(\theta) = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \nabla \log p(\tau_i \mid \theta) \mathcal{R}(\tau_i)$$

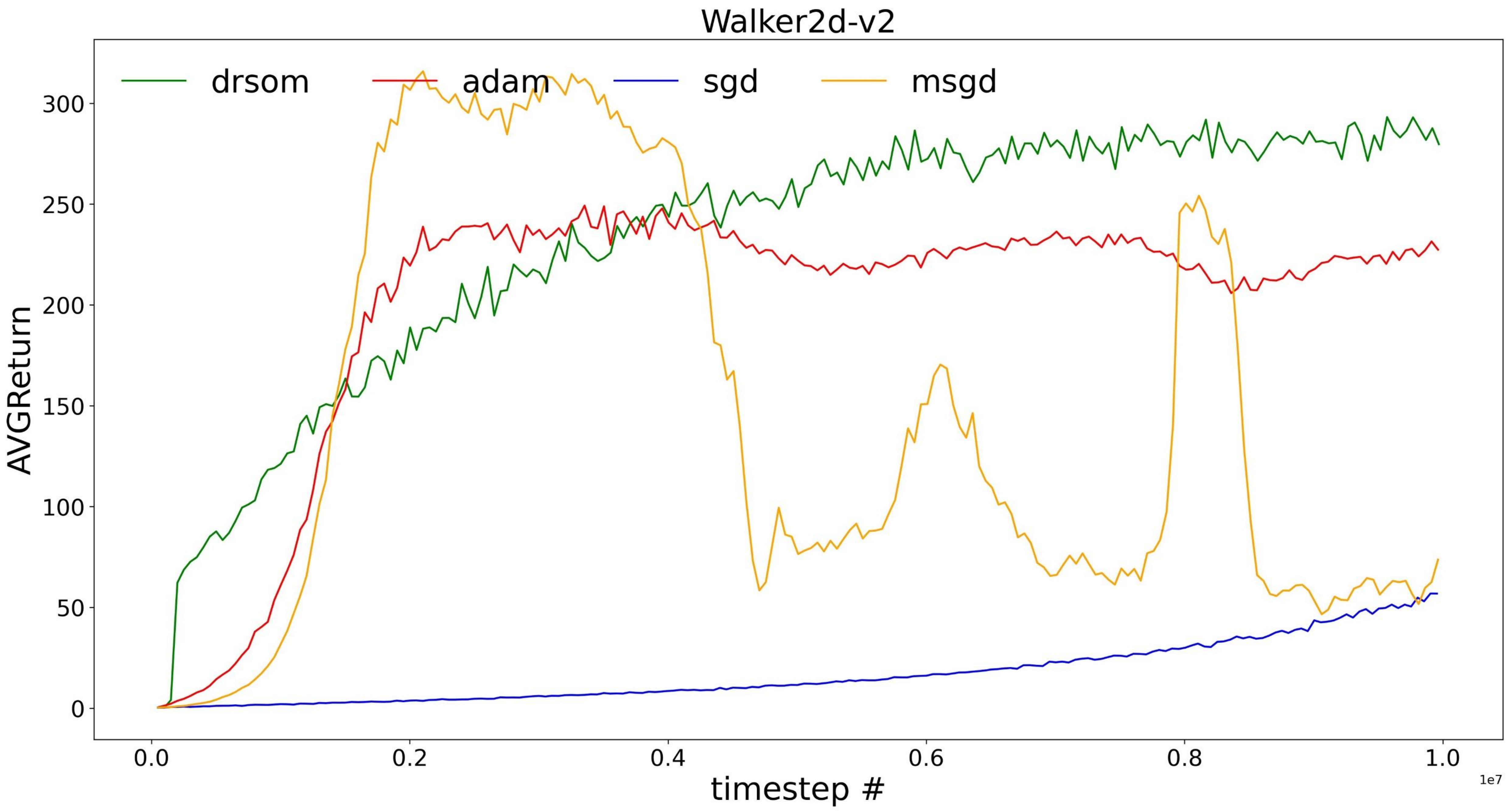
- With the estimated gradient, we can apply DRSOM to get the step size α, and update the parameter by:

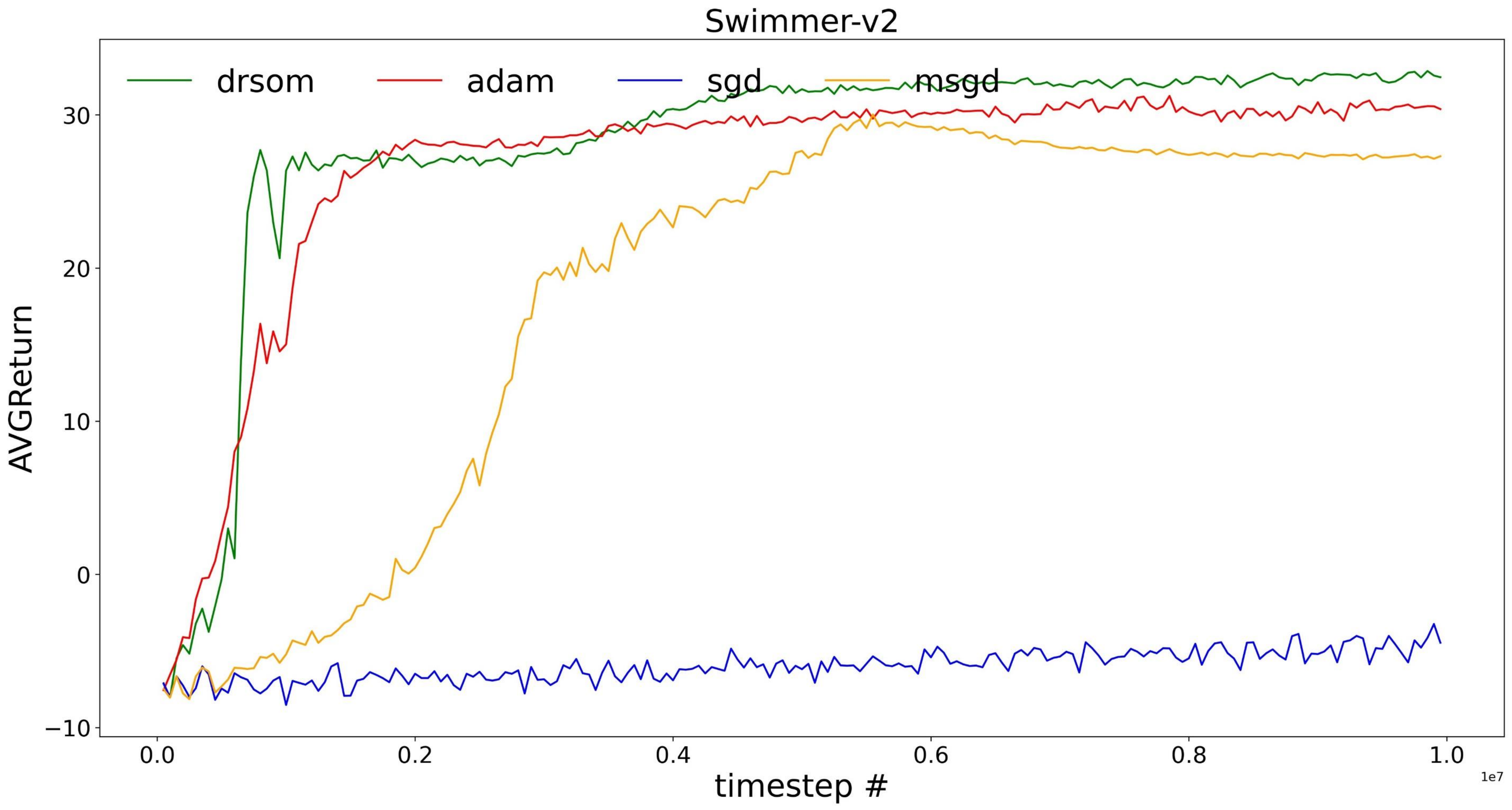
$$\theta_{t+1} = \theta_t + \alpha_t^1 \hat{\nabla} J(\theta_t) + \alpha_t^2 d_t$$

where d_t is the momentum direction.

DRSOM/ADAM/SGD Preliminary Results I

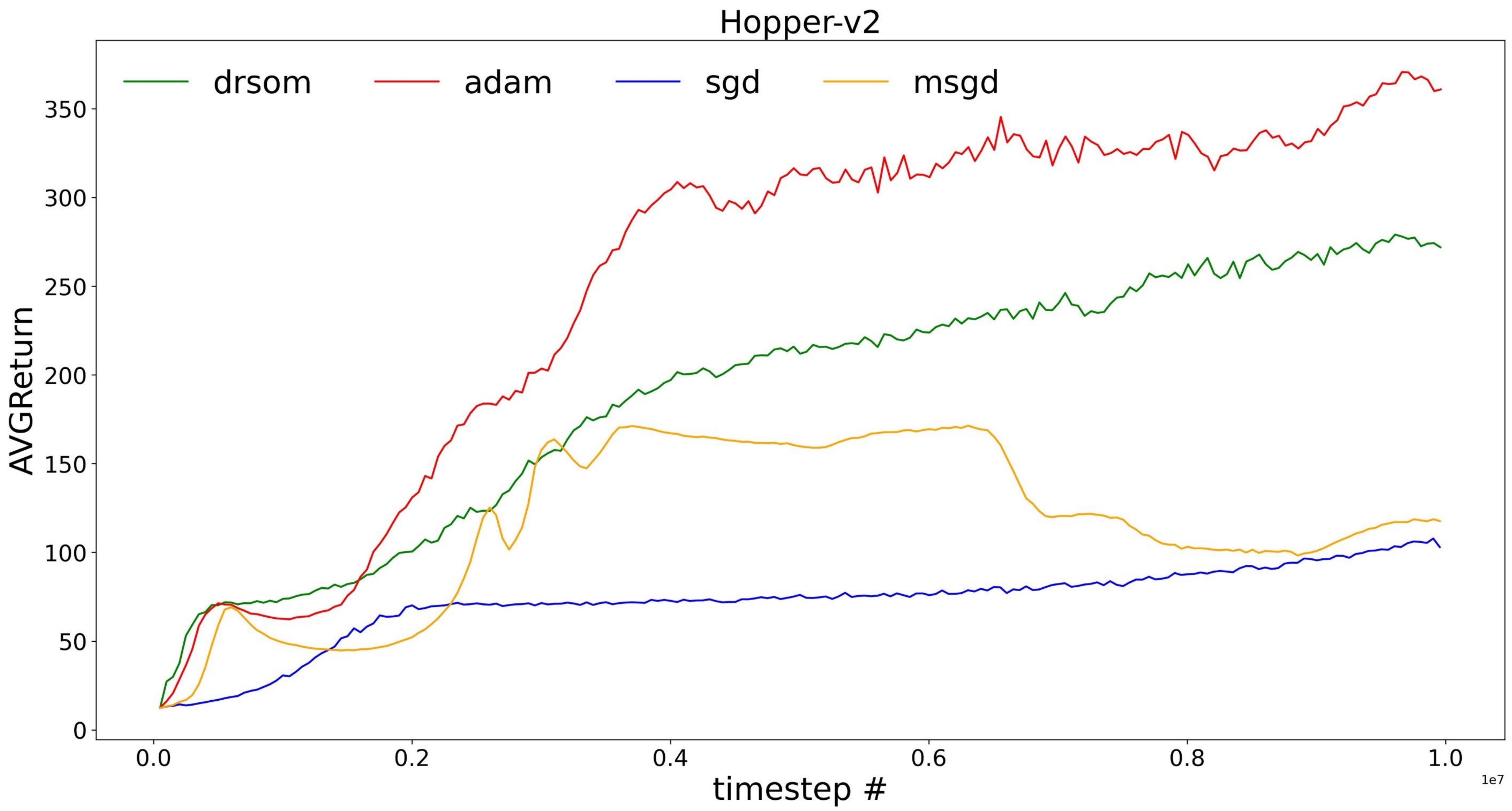
We compare the performance of DRSOM-based Reinforce with Adam-based reinforce and SGD-based reinforce(with(msgd) and without(sgd) momentum) on several GYM environments.
We set the learning rate of Adam and SGD both as 1e-3, and momentum of MSGD as 0.99

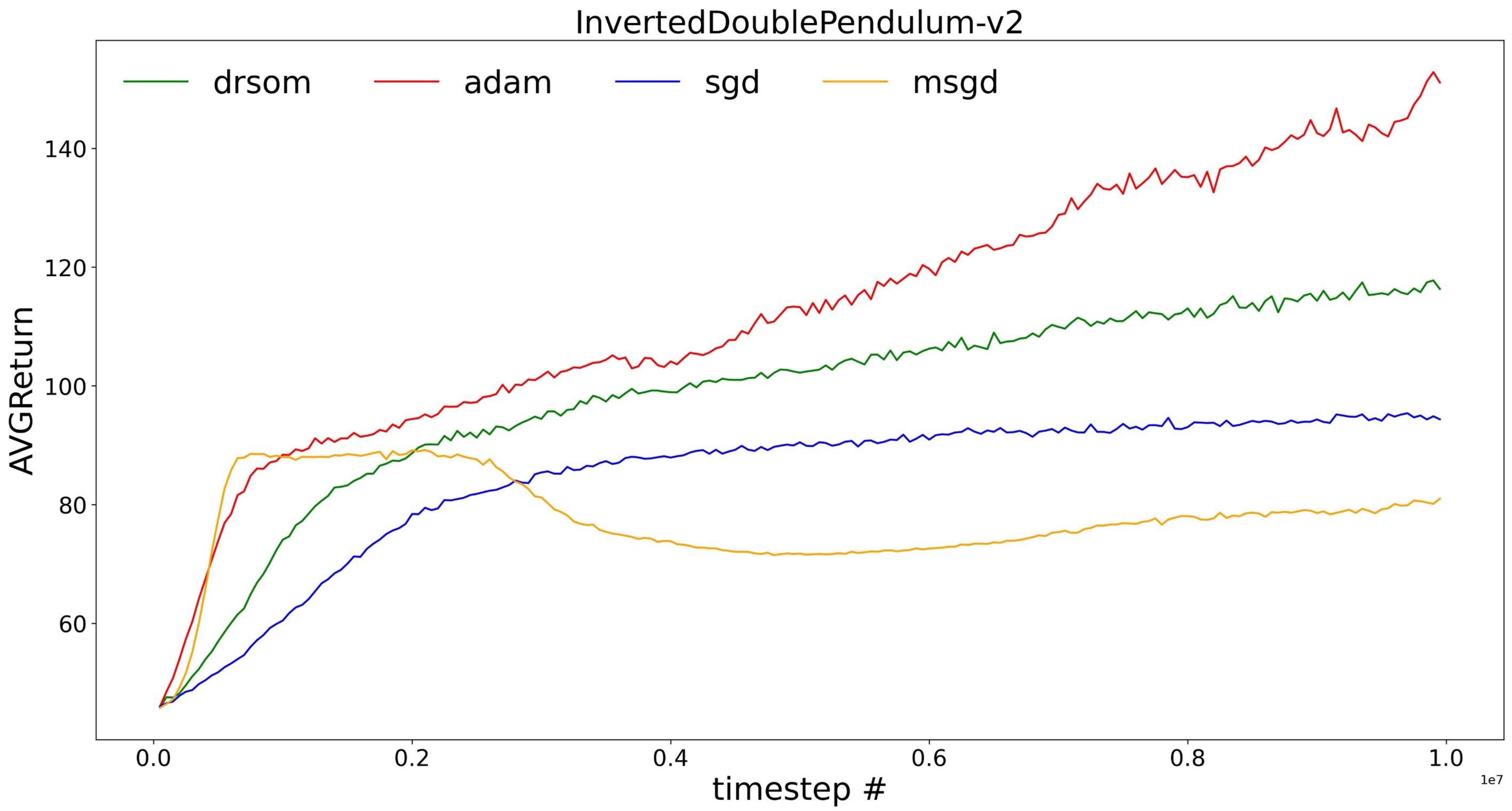




In these two cases, DRSOM converges faster and gain higher return than other algorithms. And also DRSOM seems to be more steady.

DRSOM/ADAM/SGD Preliminary Results II





In these two cases, DRSOM performs better than SGD but worse than ADAM.

DRSOM for TRPO I (Xue et al. SHUFE)

- **TRPO** attempts to optimize a surrogate function (based on the current iterate) of the objective function while keep a KL divergence constraint

$$
\begin{array}{ll}
\max_\theta & L_{\theta_k}(\theta) \\
\text{s.t.} & \mathrm{KL}\left(\Pr_\mu^{\pi_{\theta_k}} \| \Pr_\mu^{\pi_\theta}\right) \leq \delta
\end{array}
$$

- In practice, it linearizes the surrogate function, quadratizes the KL constraint, and obtain

$$
\begin{array}{ll}
\max_\theta & g_k^T(\theta - \theta_k) \\
\text{s.t.} & \frac{1}{2}(\theta - \theta_k)^T F_k (\theta - \theta_k) \leq \delta
\end{array}
$$

where F_k is the Hessian of the KL divergence.

DRSOM for TRPO II

- The problem admits a closed form solution, but requires solving a full dimension linear system,

$$F_k x = g_k$$

  leading to high computational cost !

- With the idea of DRSOM, we restrict $\theta_{k+1} \in \mathrm{span}\{g_k, d_k\}$, then update $\theta_{k+1} = \theta_k + \alpha_k^1 g_k + \alpha_k^2 d_k$. To choose the step size, we consider the following optimization problem:

$$\max_{\alpha \in \mathbb{R}^2} \; c_k^T \alpha$$

$$\text{s.t. } \frac{1}{2} \alpha^T G_k \alpha \leq \delta$$

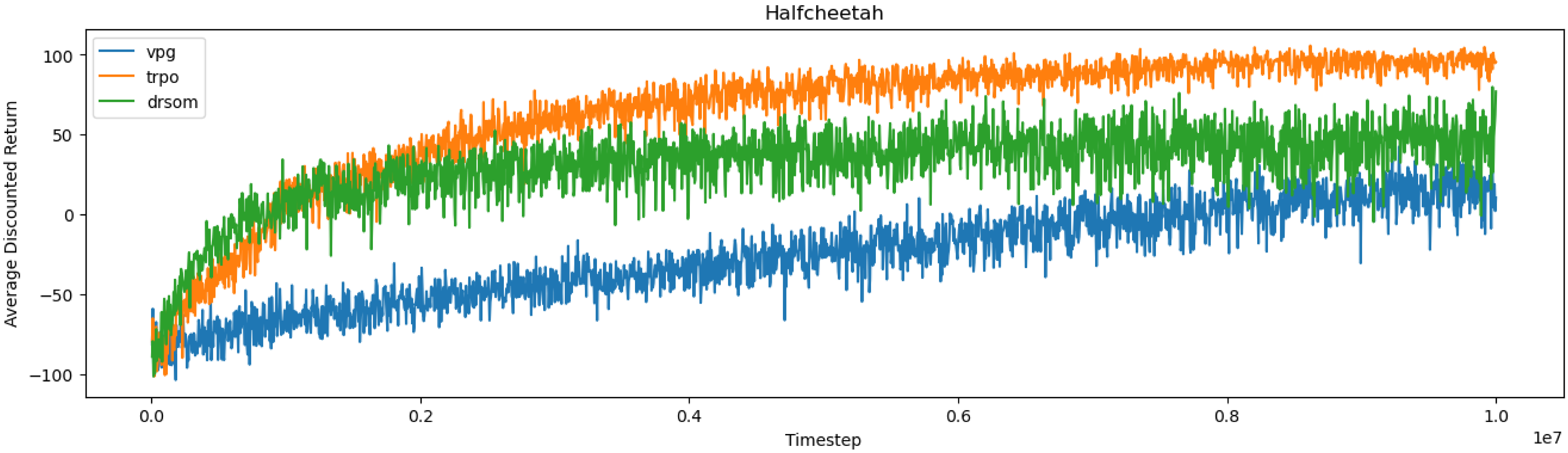
  where

$$c_k = \begin{pmatrix} \|g_k\|^2 \\ g_k^T d_k \end{pmatrix} \in \mathbb{R}^2 \text{ and } G_k = \begin{pmatrix} g_k^T H_k g_k & d_k^T H_k g_k \\ d_k^T H_k g_k & d_k^T H_k d_k \end{pmatrix} \in \mathcal{S}^2$$

Still has a closed form solution, but we only need to solve a 2 dimension linear system!

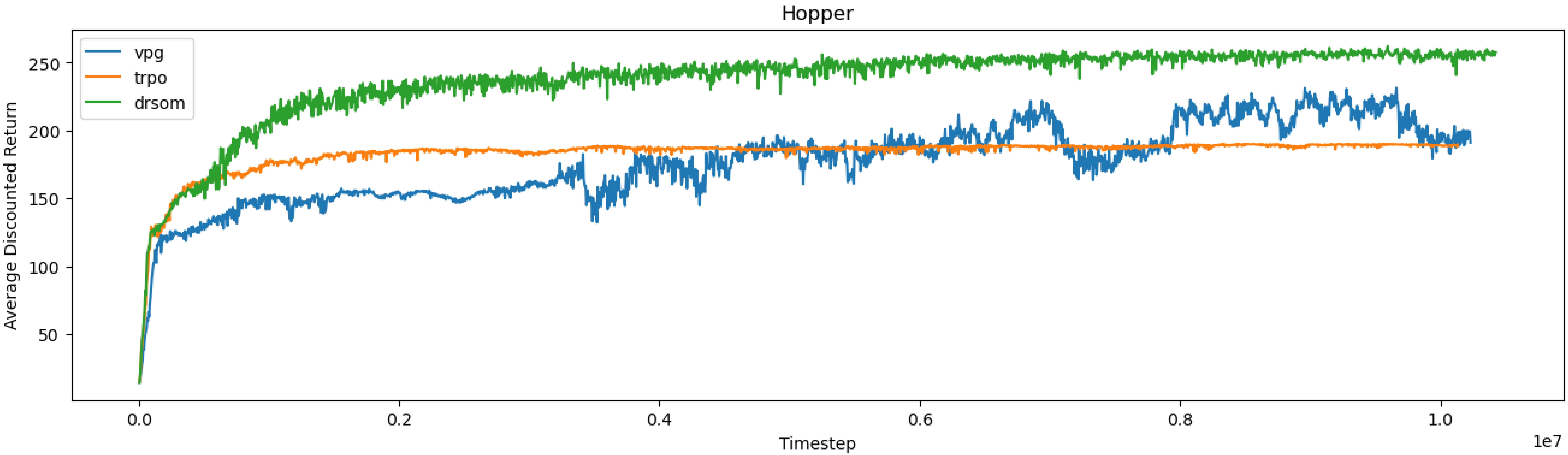
DRSOM/TRPO Preliminary Results I

- Although we only maintain the linear approximation of the surrogate function, surprisingly the algorithm works well in some RL environments



DRSOM/TRPO Preliminary Results II

- Sometimes even better than TRPO !



DRSOM for TRPO (Incorporating the Natural Gradient Direction)

- We then consider restricting $\theta_{k+1} \in \mathrm{span}\{F_k^{-1} g_k, d_k\}$, then update $\theta_{k+1} = \theta_k + \alpha_k^1 F_k^{-1} g_k + \alpha_k^2 d_k$ To choose the step size, we still consider the 2-dimensional optimization problem:

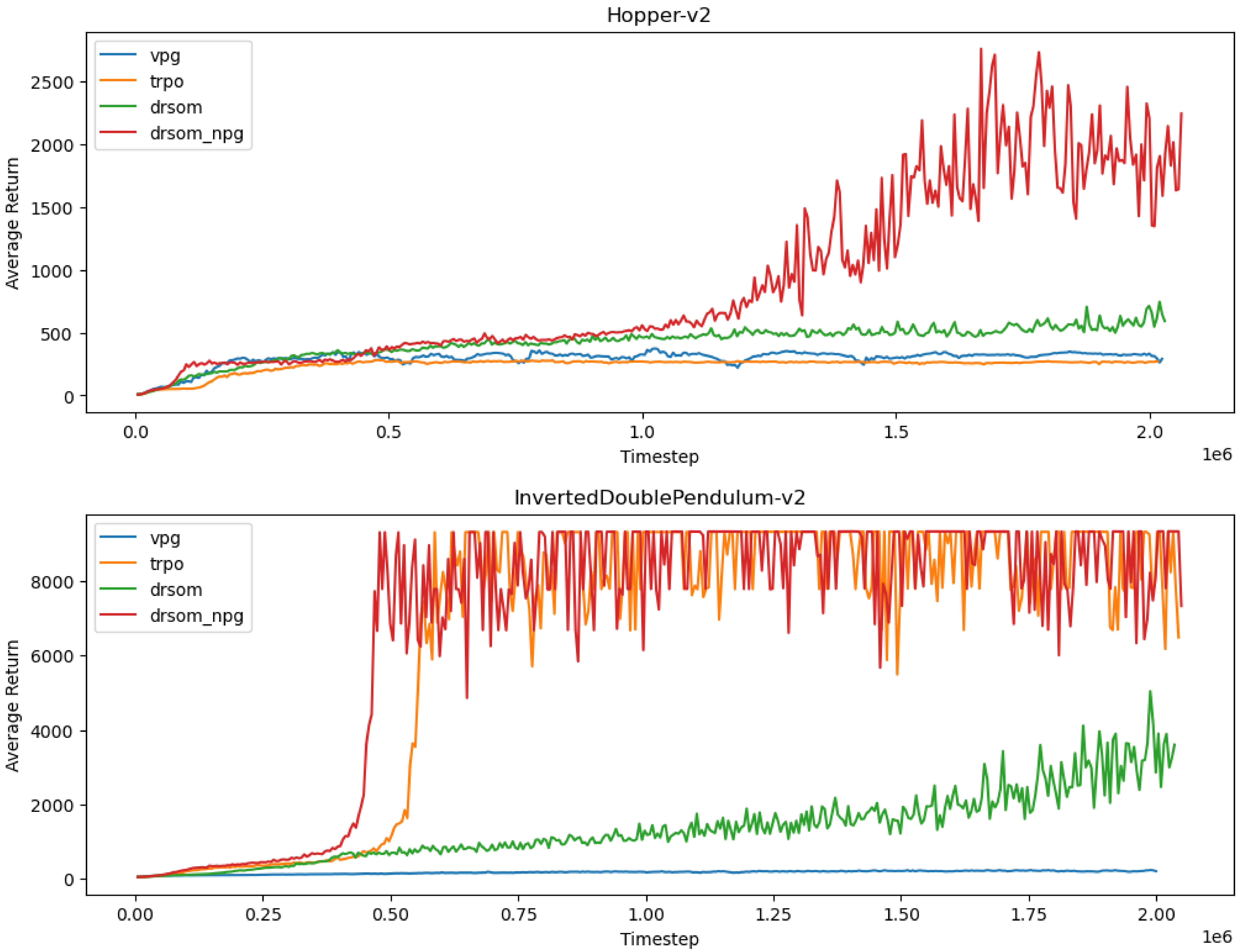
$$\max_{\alpha \in \mathbb{R}^2} \; c_k^T \alpha$$
$$\text{s.t.} \; \frac{1}{2} \alpha^T G_k \alpha \leq \delta$$

where

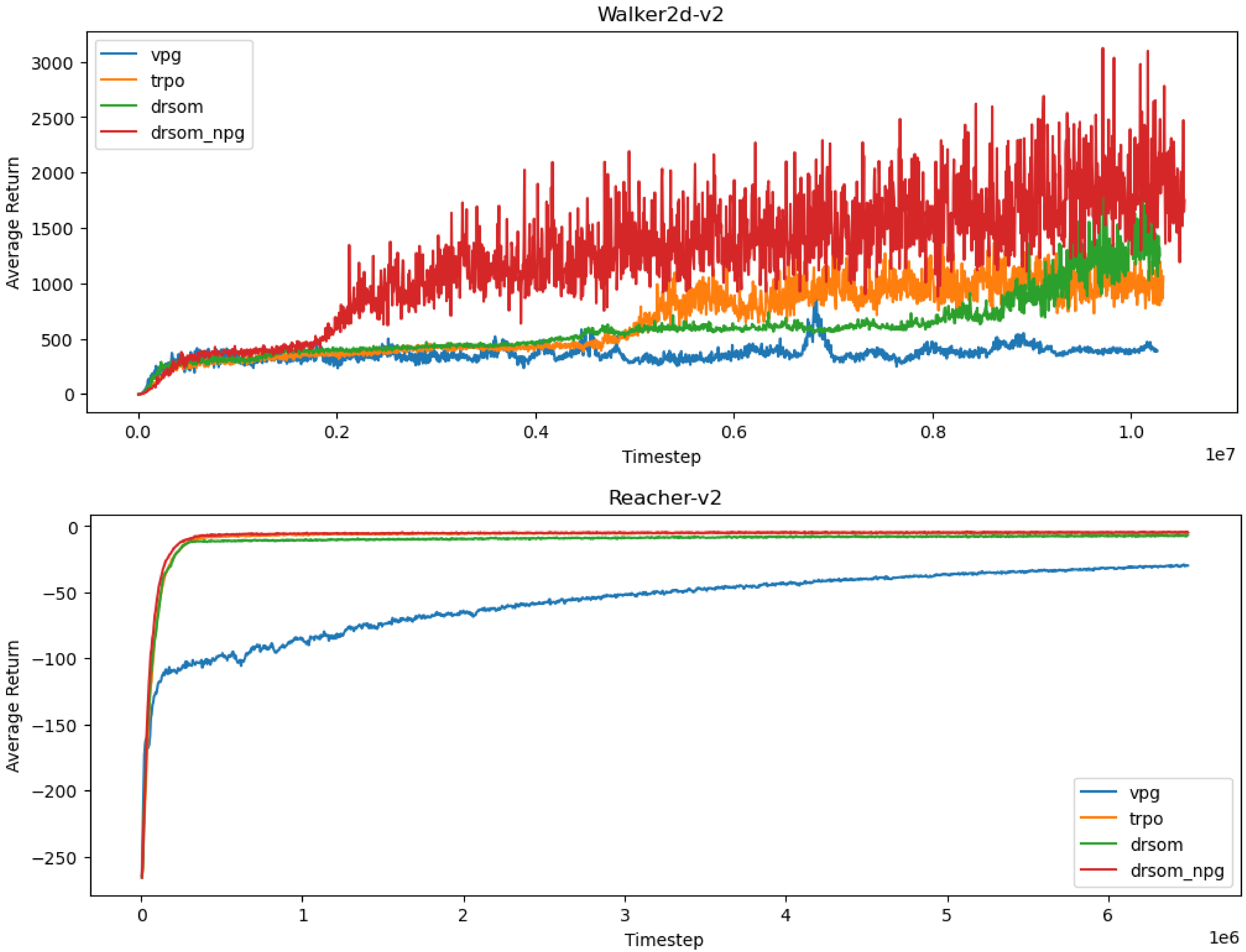
$$\boldsymbol{c}_k = \begin{pmatrix} \boldsymbol{g}_k^T \boldsymbol{F}_k^{-1} \boldsymbol{g}_k \\ \boldsymbol{g}_k^T \boldsymbol{d}_k \end{pmatrix} \in \mathbb{R}^2 \text{ and } \boldsymbol{G}_k = \begin{pmatrix} \boldsymbol{g}_k^T \boldsymbol{F}_k^{-1} \boldsymbol{g}_k & \boldsymbol{g}_k^T \boldsymbol{d}_k \\ \boldsymbol{g}_k^T \boldsymbol{d}_k & \boldsymbol{d}_k^T \boldsymbol{F}_k \boldsymbol{d}_k \end{pmatrix} \in \mathcal{S}^2$$

- It also can be seen that adding the momentum direction to TRPO improves the algorithm performance, at the cost of slightly more computation

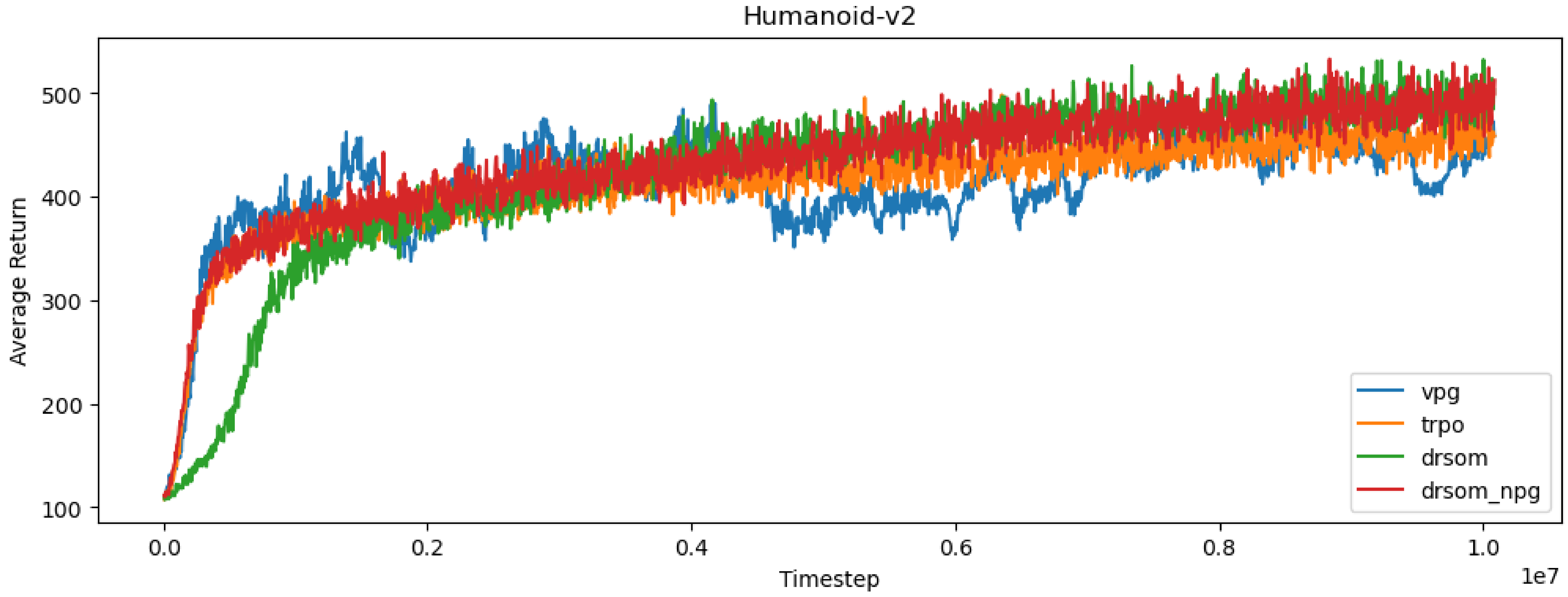
DRSOM/TRPO Preliminary Results III

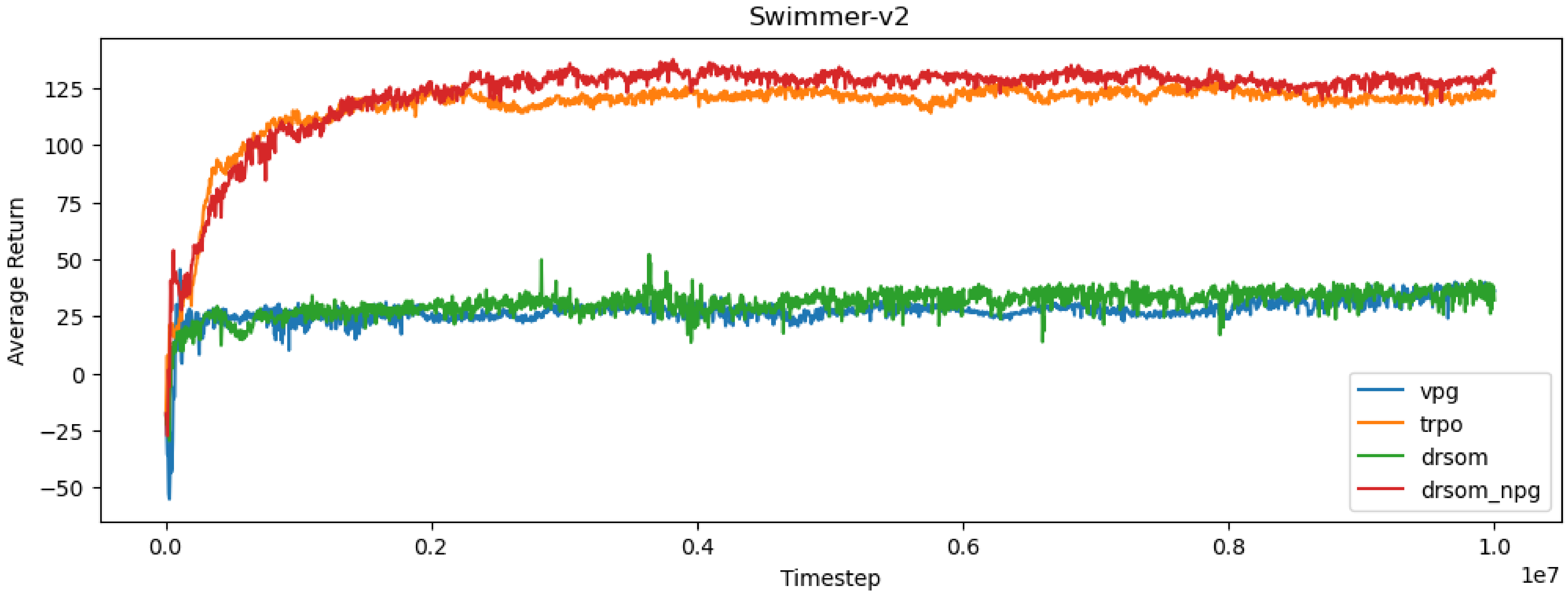


DRSOM/TRPO Preliminary Results IV



DRSOM/TRPO Preliminary Results V





DRSOM for Riemannian Optimization (Tang et al. NUS)

$$\min_{x \in \mathcal{M}} \quad f(x) \qquad \text{(ROP)}$$

- $\mathcal{M}$ is a Riemannian manifold embeded in Euclidean space $\mathbb{R}^n$.
- $f : \mathbb{R}^n \to \mathbb{R}$ is a second-order continuously differentiable function that is lower bounded in $\mathcal{M}$.

R-DRSOM: Choose an initial point $x_0 \in \mathcal{M}$, set $k = 0$, $p_{-1} = 0$;
for $k = 0, 1, \ldots, T$ **do**
Step 1. Compute $g_k = \mathrm{grad} f(x_k)$, $d_k = \mathrm{T}_{x_k \leftarrow x_{k-1}}(p_{k-1})$, $H_k g_k = \mathrm{Hess} f(x_k)[g_k]$ and $H_k d_k = \mathrm{Hess} f(x_k)[d_k]$;
Step 2. Compute the vector $c_k = \begin{bmatrix} -\langle g_k, g_k \rangle_{x_k} \\ \langle g_k, d_k \rangle_{x_k} \end{bmatrix}$ and the following matrices

$$Q_k = \begin{bmatrix} \langle g_k, H_k g_k \rangle_{x_k} & \langle -d_k, H_k g_k \rangle_k \\ \langle -d_k, H_k g_k \rangle_{x_k} & \langle d_k, H_k d_k \rangle_{x_k} \end{bmatrix}, \ , \ G_k := \begin{bmatrix} \langle g_k, g_k \rangle_{x_k} & -\langle d_k, g_k \rangle_{x_k} \\ -\langle d_k, g_k \rangle_{x_k} & \langle d_k, d_k \rangle_{x_k} \end{bmatrix}.$$

Step 3. Solve the following 2 by 2 trust region subproblem with radius $\triangle_k > 0$

$$\alpha_k := \arg \min_{\|\alpha_k\|_{G_k} \leq \triangle_k} f(x_k) + c_k^\top \alpha + \frac{1}{2} \alpha^\top Q_k \alpha;$$

Step 4. $x_{k+1} := \mathcal{R}_{x_k}\left(x_k - \alpha_k^1 g_k + \alpha_k^2 d_k\right)$;
end
Return x_k.

Max-CUT SDP

$$\textbf{Max-Cut:}\ \min\left\{-\langle L, X\rangle :\ \operatorname{diag}(X) = e,\ X \in \mathbb{S}^n_+\right\}. \tag{1}$$

$$\min\left\{-\left\langle L, RR^\top\right\rangle :\ \operatorname{diag}(RR^\top) = e,\ R \in \mathbb{R}^{n\times r}\right\}. \tag{2}$$

g67	Fval	-30977.7	-30977.7	-30977.7	-30977.7	-30977.7
n=10000	Residue	1.3e-10	2.4e-10	9.7e-10	2.6e-10	8.3e-09
m=20000	Time [s]	131.0	1371.4	177.8	1114.4	356.9
g70	Fval	-39446.1	-39446.1	-39446.1	-39446.1	-39446.1
n=10000	Residue	2.2e-10	3.7e-12	1.6e-09	2.3e-10	3.4e-09
m=9999	Time [s]	36.2	288.4	63.5	250.8	100.7
g72	Fval	-31234.2	-31234.2	-31234.2	-31234.2	-31234.2
n=10000	Residue	8.2e-11	1.8e-12	5.8e-10	2.0e-10	1.1e-08
m=20000	Time [s]	110.4	881.2	191.9	907.5	359.2
g77	Fval	-44182.7	-44182.7	-44182.7	-44182.7	-44182.7
n=14000	Residue	7.8e-11	1.4e-10	7.1e-10	1.2e-10	1.0e-08
m=28000	Time [s]	268.3	1576.9	450.4	2402.6	603.8
g81	Fval	-62624.8	-62624.8	-62624.8	-62624.8	-62624.8
n=20000	Residue	4.6e-11	1.3e-10	1.4e-09	7.9e-11	2.0e-08
m=40000	Time [s]	650.1	4283.9	1219.0	6087.4	1062.1

1D-Kohn-Sham Equation

$$\min \left\{ \frac{1}{2}\mathrm{tr}(R^\top LR) + \frac{\alpha}{4}\mathrm{diag}(RR^\top)^\top L^{-1}\mathrm{diag}(RR^\top) : \; R^\top R = I_p, \; R \in \mathbb{R}^{n\times r} \right\}, \quad (3)$$

where L is a tri-diagonal matrix with 2 on its diagonal and -1 on its subdiagonal and $\alpha > 0$ is a parameter. We terminate algorithms when $\|\mathrm{grad} f(R)\| < 10^{-4}$.

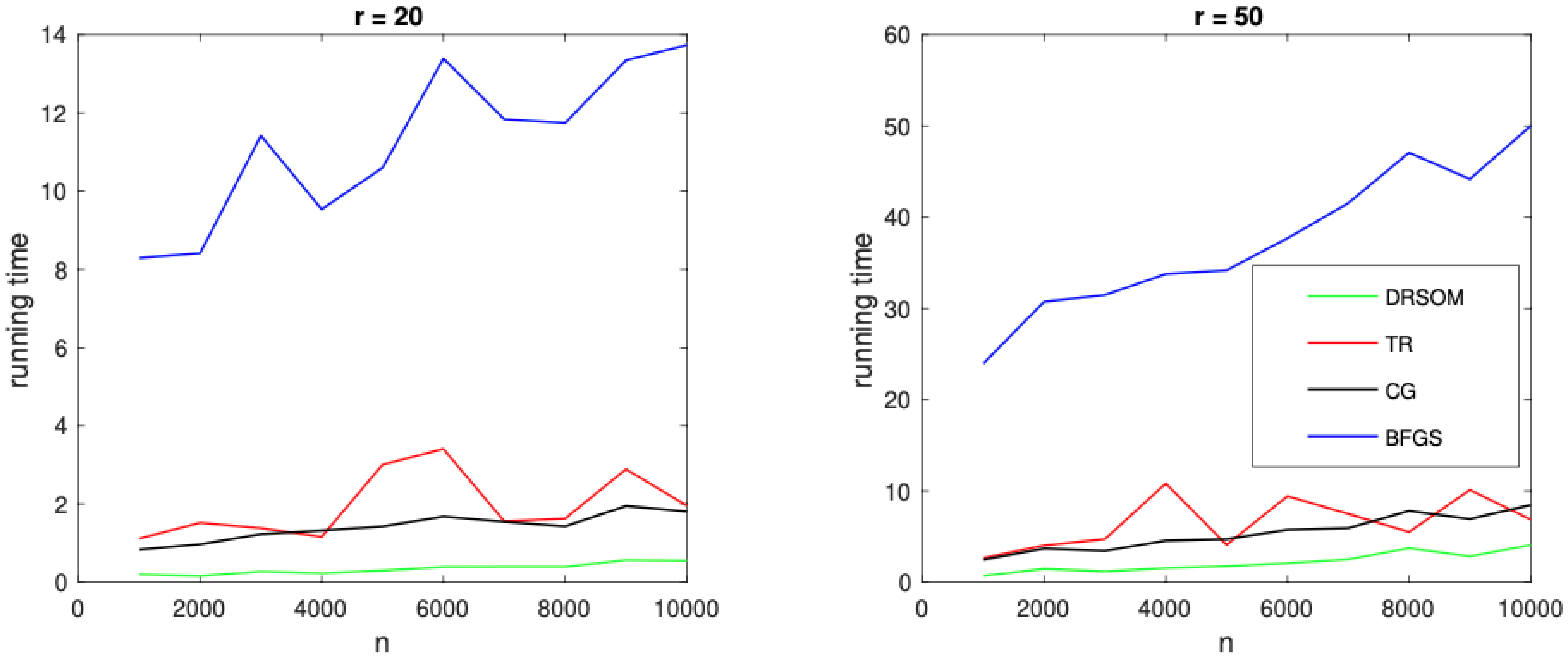


Figure 1: Results for Discretized 1D Kohn-Sham Equation. $\alpha = 1$.

DRSOM for LP Potential Reduction (Gao et al. SHUFE)

We consider a simplex-constrained QP model

$$\begin{array}{lll} \min_x & \frac{1}{2}\|Ax\|^2 & =: f(x) \\ \text{subject to} & e^\top x = 1 & \\ & x \geq 0 & \end{array}$$

$$\begin{array}{rcl} Ax - b\tau & = & 0 \\ -A^\top y - s + c\tau & = & 0 \\ b^\top y - c^\top x - \kappa & = & 0 \\ e_n^\top x + e_n^\top s + \kappa + \tau & = & 1 \end{array}$$

The self-dual embedding builds a bridge

We wish to solve a standard LP (and its dual)

$$\begin{array}{ll} \min_x & c^\top x \\ \text{subject to} & Ax = b \\ & x \geq 0 \end{array}$$

$$\begin{array}{ll} \max_{y,s} & b^\top y \\ \text{subject to} & A^\top y + s = c \\ & s \geq 0 \end{array}$$

- The homogeneous QP seems so restrictive!

Then we define the (nonconvex) potential function and apply DRSOM to it

- How to solve much more general LPs?

$$\phi(x) := \rho \log(f(x)) - \sum_{i=1}^{n} \log x_i$$

$$\nabla \phi(x) = \frac{\rho \nabla f(x)}{f(x)} - X^{-1} e$$

$$\nabla^2 \phi(x) = -\frac{\rho \nabla f(x) \nabla f(x)^\top}{f(x)^2} + \rho \frac{A^\top A}{f(x)} + X^{-2}$$

Combined with scaled gradient(Hessian) projection, the method solves LPs

DR-Potential Reduction: Preliminary Results

One feature of the DR-Potential reduction is the use of negative curvature of

$$\nabla^2 \phi(x) = -\frac{\rho \nabla f(x) \nabla f(x)^\top}{f(x)^2} + \rho \frac{A^\top A}{f(x)} + X^{-2}$$

- Computable using Lanczos iteration
- Getting LPs to high accuracy $10^{-6} \sim 10^{-8}$ if negative curvature is efficiently computed

Problem	PInfeas	DInfeas.	Compl.	Problem	PInfeas	DInfeas.	Compl.
ADLITTLE	1.347e-10	2.308e-10	2.960e-09	KB2	5.455e-11	6.417e-10	7.562e-11
AFIRO	7.641e-11	7.375e-11	3.130e-10	LOTFI	2.164e-09	4.155e-09	8.663e-08
AGG2	3.374e-08	4.859e-08	6.286e-07	MODSZK1	1.527e-06	5.415e-05	2.597e-04
AGG3	2.248e-05	1.151e-06	1.518e-05	RECIPELP	5.868e-08	6.300e-08	1.285e-07
BANDM	2.444e-09	4.886e-09	3.769e-08	SC105	7.315e-11	5.970e-11	2.435e-10
BEACONFD	5.765e-12	9.853e-12	1.022e-10	SC205	6.392e-11	5.710e-11	2.650e-10
BLEND	2.018e-10	3.729e-10	1.179e-09	SC50A	1.078e-05	6.098e-06	4.279e-05
BOEING2	1.144e-07	1.110e-08	2.307e-07	SC50B	4.647e-11	3.269e-11	1.747e-10
BORE3D	2.389e-08	5.013e-08	1.165e-07	SCAGR25	1.048e-07	5.298e-08	1.289e-06
BRANDY	2.702e-05	7.818e-06	1.849e-05	SCAGR7	1.087e-07	1.173e-08	2.601e-07
CAPRI	7.575e-05	4.488e-05	4.880e-05	SCFXM1	4.323e-06	5.244e-06	8.681e-06
E226	2.656e-06	4.742e-06	2.512e-05	SCORPION	1.674e-09	1.892e-09	1.737e-08
FINNIS	8.577e-07	8.367e-07	1.001e-05	SCTAP1	5.567e-07	8.430e-07	5.081e-06
FORPLAN	5.874e-07	2.084e-07	4.979e-06	SEBA	2.919e-11	5.729e-11	1.448e-10
GFRD-PNC	4.558e-05	1.052e-05	4.363e-05	SHARE1B	3.367e-07	1.339e-06	3.578e-06
GROW7	1.276e-04	4.906e-06	1.024e-04	SHARE2B	2.142e-04	2.014e-05	6.146e-05
ISRAEL	1.422e-06	1.336e-06	1.404e-05	STAIR	5.549e-04	8.566e-06	2.861e-05
STANDATA	5.645e-08	2.735e-07	5.130e-06	STANDGUB	2.934e-08	1.467e-07	2.753e-06
STOCFOR1	6.633e-09	9.701e-09	4.811e-08	VTP-BASE	1.349e-10	5.098e-11	2.342e-10

- Now solving small and medium Netlib instances in 10 seconds

  within 1000 iterations
- In MATLAB and getting transferred into C for acceleration

Ongoing Research and Future Directions

- How to enforce or remove Assumption c) in algorithms/analyses
- How to design a more adaptive-radius mechanism with the same complexity bound, e.g., the trust-region framework of Curtis et al., 2017
- Incorporate the second-order steepest-descent direction, the eigenvector of the most negative Hessian eigenvalue
- Indefinite and Randomized Hessian rank-one updating vs BFGS
- Dimension Reduced Non-Smooth/Semi-Smooth Newton
- Dimension Reduced Second-Order Methods for optimization with more complicated constraints

THANK YOU