SOLNP+: A Zero-Order Optimization Solver

SOLNP+ Group: Jiyuan Tan, Tianhao Liu, Jinsong Liu

June 24, 2023

PART I:

Background

Zero-Order or Derivative-Free Nonlinear Programming

$$
\begin{aligned}
&\min_{s\in\mathbb{R}^d} f(s) \\
s.t.\quad & h_1(s) = 0, \\
& l_h \leqslant h_2(s) \leqslant u_h, \\
& l_s \leqslant s \leqslant u_s.
\end{aligned}
$$

$$\xrightarrow{\text{Adding Slack}}$$

$$
\begin{aligned}
&\min_{x\in\mathbb{R}^d} \quad f(x) \\
s.t.\quad & g(x) = 0, \\
& l_x \leqslant x \leqslant u_x.
\end{aligned}
$$

- All functions are smooth functions.
- The solver only has access to **zero-order information**.
- Function evaluation may be **expensive**.
- There may be some **noises** in function evaluation.

Features of Derivative-Free Optimization(DFO)

- Less prior information: DFO do not need first-order or second-order information
- **Robustness** to noise
- Wide applications in **Practice**

Applications of Derivative-Free Algorithms

- **Aircraft design**: Giunta, Anthony A., et al. *Multidisciplinary optimisation of a supersonic transport using design of experiments theory and response surface modelling*, 1997.
- **Parameter estimation in time series**: Bennedsen, Mikkel, Eric Hillebrand, and Jingying Zhou Lykke, *Global temperature projections from a statistical energy balance model using multiple sources of historical data*, 2022.
- **Selecting the tuning parameters of deep neural network**: Snoek, Jasper, Hugo Larochelle, and Ryan P. Adams, *Practical bayesian optimization of machine learning algorithms*, 2012.
- **Prompt Optimization and Large-Model Training…**

Popular Methods

- **Model-Based Trust-Region Methods**: minimization of model functions in an adaptively chosen trust region.
  - Example: Powell's Derivative-Free Optimization (PDFO) solvers.
- **Model-Based Descend Method**: use model functions to find a descent direction and search along the direction.
  - Example: Implicit filtering, SOLNP+.
- **Direct search method**: compare the current points with some near points to reduce function value.
  - Example: Nonlinear Optimization with Mesh Adaptive Directional Search (NOMAD), Nelder-Mead Simplex method.
- **Generic, Simulated Annealing,⋯**

SOLNP+: History

- First proposed by Y in 1989.
- Originally implemented (SOLNP) in **Matlab**, 1989.
- **R** implementation (**Rsolnp**) by Alexios Ghalanos and Stefan Theussl, 2011.
- **New and C implementation (SOLNP+) with improvements, 2022; and addition of Randomized BCG and DRSOM for unconstrained optimization by Tan et al., 2023**

SOLNP+: Interface

- **SOLNP+** currently provides the following interface
  - **Matlab**
  - **Python**
  - **Julia**
- **Github link: https://github.com/COPT-Public/SOLNP_plus**

PART II:

Nonlinear Constrained Optimization

SOLNP+: Overview

- Use **finite difference** to approximate the gradient.
- Approximate the constraints by **linear function**.
- Use **Augmented Lagrangian Method** (ALM) to solve the nonlinear constrained problem.
- Use **Sequential Quadratic Programming** (SQP) with **BFGS update** to solve ALM subproblems.

SOLNP+ : Approximate Gradient and Constraints

- Use **finite difference** to calculate the approximated gradient.

$$[\nabla_\delta f(x)]_i = \frac{f(x+\delta e_i) - f(x)}{\delta}, \; e_i = [0, \; \cdots, 1, \; \cdots 0].$$

- Approximate the nonlinear constraints by **affine/linear** function:

$$g(x) = 0 \quad \Rightarrow \quad g(x_k) + \nabla_{\delta_k} g(x_k)^T (x - x_k) = 0.$$

SOLNP+ Outer Iteration: ALM Framework

- **Modified Augmented Lagrangian function**

$$L_k(x,y) = f(x) - y^T\left[g(x) - (g(x_k) + \nabla_{\delta_k} g(x_k)^T(x - x_k))\right] + \frac{\rho_k}{2}\left\|g(x) - (g(x_k) + \nabla_{\delta_k} g(x_k)^T(x - x_k))\right\|_2^2.$$

- **Primal Update** (Robinson, 1972):

$$\begin{aligned} \min\ & L_k(x, y_k) \\ \text{s.t. } & g(x_k) + \nabla_{\delta_k} g(x_k)^T(x - x_k) = 0, \\ & l_x \le x \le u_x, \end{aligned}$$

where y_k is the approximated Lagrange multiplier with respect to the **linear constraints**.

Solve ALM Subproblem: Find Feasible Solution

- The linearized problem may not be **feasible**.
- Find (approximated) **feasible solution** x_k^0 by solving the following LP.

$$
\begin{aligned}
&\min \ \tau \\
&\text{s.t. } g(x_k)(1-\tau) + \nabla_{\delta_k} g(x_k)^T (x - x_k) = 0, \\
&\qquad l_x \leq x \leq u_x, \\
&\qquad \tau \geq 0
\end{aligned}
$$

- When τ is small, we find **a near feasible** start point.
- Start from x_k^0, move along the direction that is in the null space of $\nabla_{\delta_k} g(x_k)^T$.

SOLNP+ Inner Iteration: SQP and BFGS Update

- SOLNP+ generates the following **sequential quadratic programming** (SQP) to solve the ALM subproblem.

$$\min \frac{1}{2}(x - x_k^i)^T H_k^i (x - x_k^i) + \nabla_{\delta_k} L_k(x_k^i, y_k)^T (x - x_k^i)$$

$$\text{s.t. } g(x_k) + \nabla_{\delta_k} g(x_k)^T (x - x_k) = b_k,$$

$$l_x \le x \le u_x.$$

where $b_k = g(x_k) + \nabla_{\delta_k} g\left(x^k\right)^T \left(x_k^0 - x_k\right)$ and **BFGS update**:

$$H_k^{i+1} = H_k^i + \frac{tt^T}{t^T s} - \frac{(H_k^i s)(H_k^i s)^T}{s^T H_k^i s}, \quad H_1^0 = I.$$

and $t = \nabla_{\delta_k} L_k(x_k^{i+1}, y_k) - \nabla_{\delta_k} L_k(x_k^i, y_k)$ d $\quad s = x_k^{i+1} - x_k^i.$

Solve SQP: QP Subproblem

- Let $\hat{x}_k^i$ be the solution of the following **QP**.

$$
\begin{aligned}
\min\ & \frac{1}{2}(x - x_k^i)^T H_k^i (x - x_k^i) + \nabla_{\delta_k} L_k(x_k^i, y_k)^T (x - x_k^i) \\
\text{s.t. } & g(x_k) + \nabla_{\delta_k} g(x_k)^T (x - x_k) = b_k, \\
& l_x \le x \le u_x.
\end{aligned}
$$

- SOLNP+ applies **IPM** to solve it and performs **line search** between $\hat{x}_k^i$ and x_k^i to reduce $L_k(x, y_k)$
- Use **Lagrange multiplier** of linear constraints as an approximation to the true multiplier.

SOLNP+: Dealing with Feasibility

- Sometimes **nonlinear feasibility** is difficult to satisfy
- When poor feasibility is detected, SOLNP+ will switch to another formulation for satisfying feasibility

$$
\begin{aligned}
\min\ & f(x) \\
\text{s.t. } & g(x) = 0 \\
& l_s \le h(x) \le u_s \\
& l_x \le x \le u_x
\end{aligned}
$$

Use ℓ_1 penalty $\Longrightarrow$

$$
\begin{aligned}
\min\ & f(x) + \mu_k e^\top u + \mu_k e^\top v \\
\text{s.t. } & g(x) + u - v = 0 \\
& h(x) - s = 0 \\
& l_s \le s \le u_s, \quad u, v \ge 0 \\
& l_x \le x \le u_x
\end{aligned}
$$

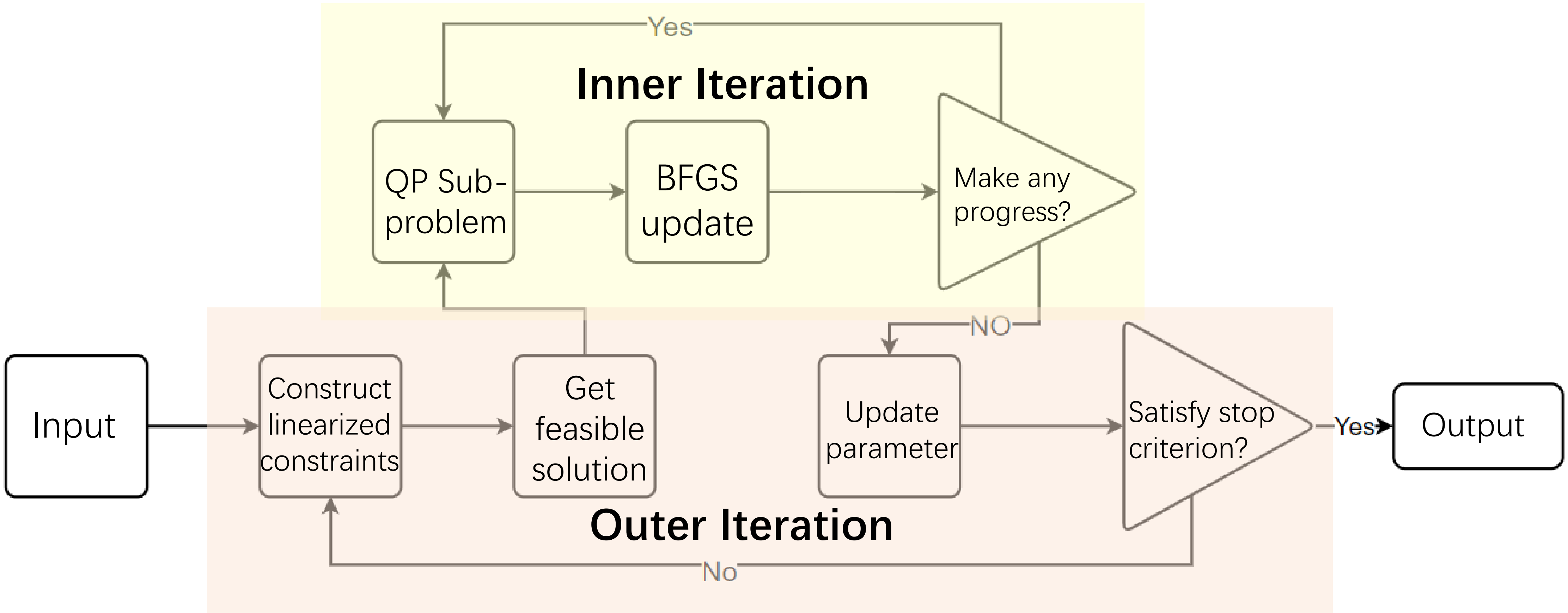
where μ_k is adjusted according to feasibility change.

Computation Aspects for SOLNP+

- **Heuristics** to update the penalty parameter ρ_k , the augmented Lagrangian parameter
- **Restart** when the algorithm cannot make any progress.
- **Line search** to improve quality of solution.
- **Adaptively choose** δ_k to increase robustness.
- **Feasibility reformulation**

SOLNP+ Solver

SOLNP+ is written in ANSI C and under active development.



Computational Results I: Noiseless functions

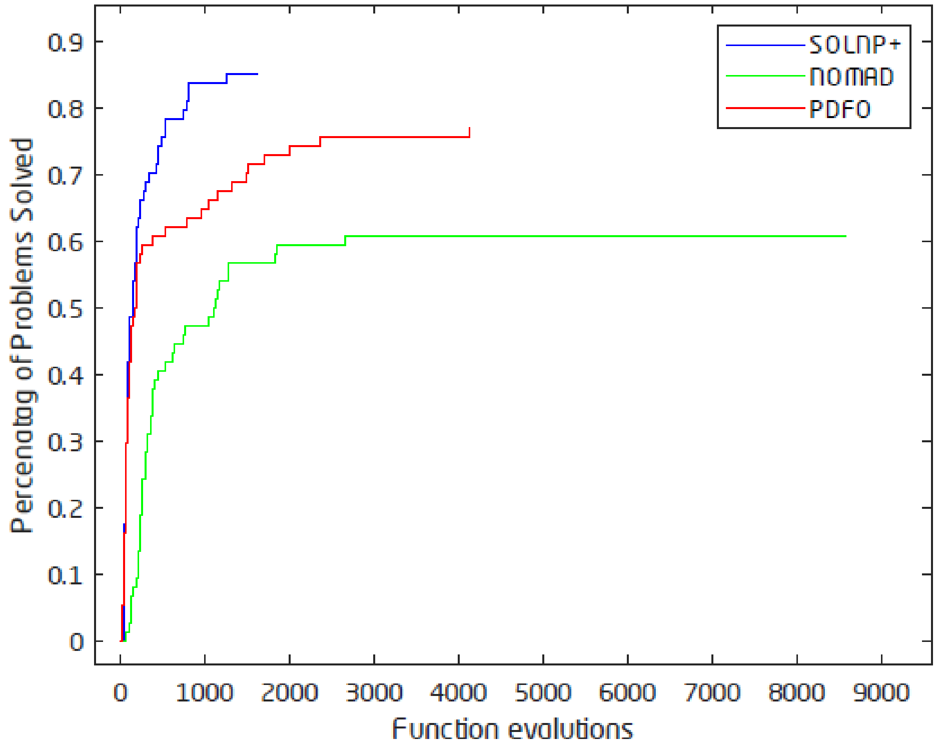


Figure 1: Test result of 74 problems in Hock and Schittkowski Hock and Schittkowski [1980] problems. Total running time of SOLNP+, NOMAD, PDFO are 1.410250e+00s, 2.251209e+03s and 5.324220e+00s.

TM Ragonneau and Z Zhang. Pdfo: Cross-platform interfaces for powells derivative-free optimization solvers (version 1.1), 2021.
Le Digabel, Sébastien. "Algorithm 909: NOMAD: Nonlinear optimization with the MADS algorithm." *ACM Transactions on Mathematical Software (TOMS)* 37.4 (2011): 1-15.and Christophe Tribes.

Computational Results I: Noiseless functions

Prob.	Dim.	Number of Evaluations			Objective Function Value		
		SOLNP+	NOMAD	PDFO	SOLNP+	NOMAD	PDFO
HS11	2	41	312	53	-8.49787e+00	-8.49846e+00	-8.49846e+00
HS26	3	81	326	146	1.43427e-06	3.56000e+00	2.11600e+01
HS38	4	165	625	460	1.62759e-05	2.25010e-13	7.87702e+00
HS40	4	74	239	76	-2.50025e-01	-2.40655e-01	-2.50000e-01
HS46	5	272	252	537	4.30387e-09	3.33763e+00	9.24220e-06
HS56	7	158	383	263	-3.45603e+00	-1.00000e+00	-3.45616e+00
HS78	5	82	296	110	-2.91974e+00	2.73821e+00	-2.91970e+00
HS79	5	75	353	101	7.87804e-02	1.72669e-01	7.87768e-02
HS80	5	104	312	96	5.39484e-02	2.59025e-01	5.39498e-02
HS81	5	138	328	153	5.39470e-02	1.21224e-01	5.39498e-02
HS84	5	217	1818	54	-5.28034e+06	-5.28019e+06	-5.28033e+06
HS93	6	148	1109	2367	1.35083e+02	1.35525e+02	1.35076e+02
HS106	8	530	2670	4000	7.08435e+03	7.66634e+03	8.94823e+03

Table 1: Test results on Hock and Schittkowski Hock and Schittkowski [1980] problems. The blue color means that the solver returns an approximate optimal solution with better objective value.

Computational Results II: Functions with Noise

- We consider the following problem,

$$\begin{aligned} &\min_{x\in\mathbb{R}^d} \quad f(x) \\ &s.t. \;\; g(x) = 0, \\ &\qquad\;\; l_x \leqslant x \leqslant u_x. \end{aligned}$$

with observed value

$$\begin{aligned} \widehat{f}(x) &= f(x)(1+\sigma N_1(x)), \\ \widehat{g}(x) &= g(x)(1+\sigma N_2(x)) \end{aligned}$$

where $N_i(x) \sim N(0, I)\ i.i.d.$, $\sigma = 10^{-4}$.

- If the infeasibility error of the point is less than 10^{-3}, we regard it as feasible point.

Computational Results II: Noise functions

Prob.	Dim	Average Number of Evaluations				Average Objective Function Value			
		SOLNP	SOLNP+	NOMAD	PDFO	SOLNP	SOLNP+	NOMAD	PDFO
HS11	2	118.13(20/50)	151.80	238.42	43.54	4.03901e+03	-8.49903e+00	-8.49988e+00	-8.42549e+00
HS26	3	125.55(21/50)	304.48	213.24	44.26	1.67703e+01	2.75509e-05	3.49606e+00	2.11602e+01
HS28	3	66.60	147.68	326.26	60.68	8.04709e+00	2.33612e-07	1.98141e+00	1.67747e-04
HS38	4	37.00	711.40	702.12	261.58	7.77777e+03	1.28702e+00	1.57504e-01	7.93643e+00
HS40	4	512.17(44/50)	114.18	179.08	67.14	-2.04409e-01	-2.50388e-01	-2.37238e-01	-2.49996e-01
HS46	5	127.00(28/50)	394.60	280.70	101.02	2.76249e+00	2.36928e-05	3.33766e+00	1.60209e+00
HS56	7	21.93(36/50)	374.06	377.60	133.98 (1/50)	-1.00014e+00	-3.45475e+00	-9.99998e-01	-3.45015e+00
HS78	5	–(50/50)	168.50	208.34	73.58	–	-2.91984e+00	-2.77044e+00	-2.91955e+00
HS79	5	889.00(47/50)	104.30	273.48	79.62 (2/50)	3.75856e+00	7.87920e-02	4.27542e+01	7.87840e-02
HS80	5	–(50/50)	100.06	221.14	68.88	–	5.39374e-02	7.29409e-02	5.39545e-02
HS81	5	1194.00(49/50)	246.50	223.58	125.20 (1/50)	2.71448e-01	5.43039e-02	9.10489e-02	5.39526e-02
HS84	5	17.96	427.96	589.86(36/50)	54.11 (41/50)	-2.35125e+06	-5.22708e+06	-5.25703e+06	-5.24458e+06
HS93	6	19.00(39/50)	805.14	469.20	86.38	1.37064e+02	1.35927e+02	1.35562e+02	1.35922e+02
HS106	8	45.00(49/50)	855.52(2/50)	1473.64	82.30	1.49936e+04	1.39741e+04	7.80392e+03	1.49971e+04

Table 2: Test results with noise on Hock and Schittkowski Hock and Schittkowski [1980] problems. Each experiment is repeated 50 times. The blue color means that the solver returns a solution with better objective value."(fail time/total time)" means the number of times for which the solvers return an infeasible solution. The average is taken for all the feasible solutions returned by the solver. Average test time of SOLNP, SOLNP+, NOMAD and PDFO on these problems are 7.14111e-01, 4.48086e-02, 1.51465e+02, and 1.26806e-01 seconds.

Computational Results III: Feasibility Problem

Solver	Solved	Time/s	Function Evaluation
SOLNP+	53/129	601.9	213298
PDFO	55/129	3056.7	324265

Test Result on the feasibility problems with dimension less than 200 of Cutest problem set. Solved means the solvers return a point with infeasibility less than 0.001.

Gould, Nicholas IM, Dominique Orban, and Philippe L. Toint. "CUTEst: a constrained and unconstrained testing environment with safe threads for mathematical optimization." *Computational optimization and applications* 60 (2015): 545-557.

Computational Results IV: Tumor Growth Problem

$$\begin{aligned}
\min_{t_1,\cdots,t_n,a_1,\cdots,a_n} \quad & P^* = P(t_{\text{end}}) + Q(t_{\text{end}}) + Q_P(t_{\text{end}}) \\
s.t. \qquad & 0 \le t_i \le t_{\text{end}}, \qquad i = 1, \cdots n, \\
& 0 \le a_i \le 1, \qquad i = 1, \cdots n, \\
& 0 \le \max_{t \in [0, t_{\text{end}}]} C(t) \le v_{\max}, \\
& 0 \le \int_0^{t_{\text{end}}} C(t) \mathrm{d}t \le v_{\text{cum}}.
\end{aligned}$$

At time t_i , we give drug of dosage a_i to the patient, P^* is the size of tumor at the end of the treatment; $C(t)$ is the drug concentration. The goal of this problem is to minimize the tumor size

Tumor Growth Problem continued

- P^* is calculated by the ODE

$$\frac{\mathrm{d}C}{\mathrm{d}t} = -\theta_1 C$$
$$\frac{\mathrm{d}P}{\mathrm{d}t} = \theta_4 P(1 - \frac{P + Q + Q_P}{K}) + \theta_5 Q_P - \theta_3 P - \theta_1\theta_2 CP$$
$$\frac{\mathrm{d}Q}{\mathrm{d}t} = \theta_3 P - \theta_1\theta_2 CQ$$
$$\frac{\mathrm{d}Q_P}{\mathrm{d}t} = \theta_1\theta_2 CQ - \theta_5 Q_P - \theta_6 Q_P,$$

Georgios Arampatzis, Daniel Walchli, Pascal Weber, Henri Rastas, and Petros Koumoutsakos. (μ, λ)-ccma-es for constrained optimization with an application in pharmacodynamics. In *Proceedings of the Platform for Advanced Scientific Computing Conference*, pages 1–9, 2019

Tumor Growth Problem III

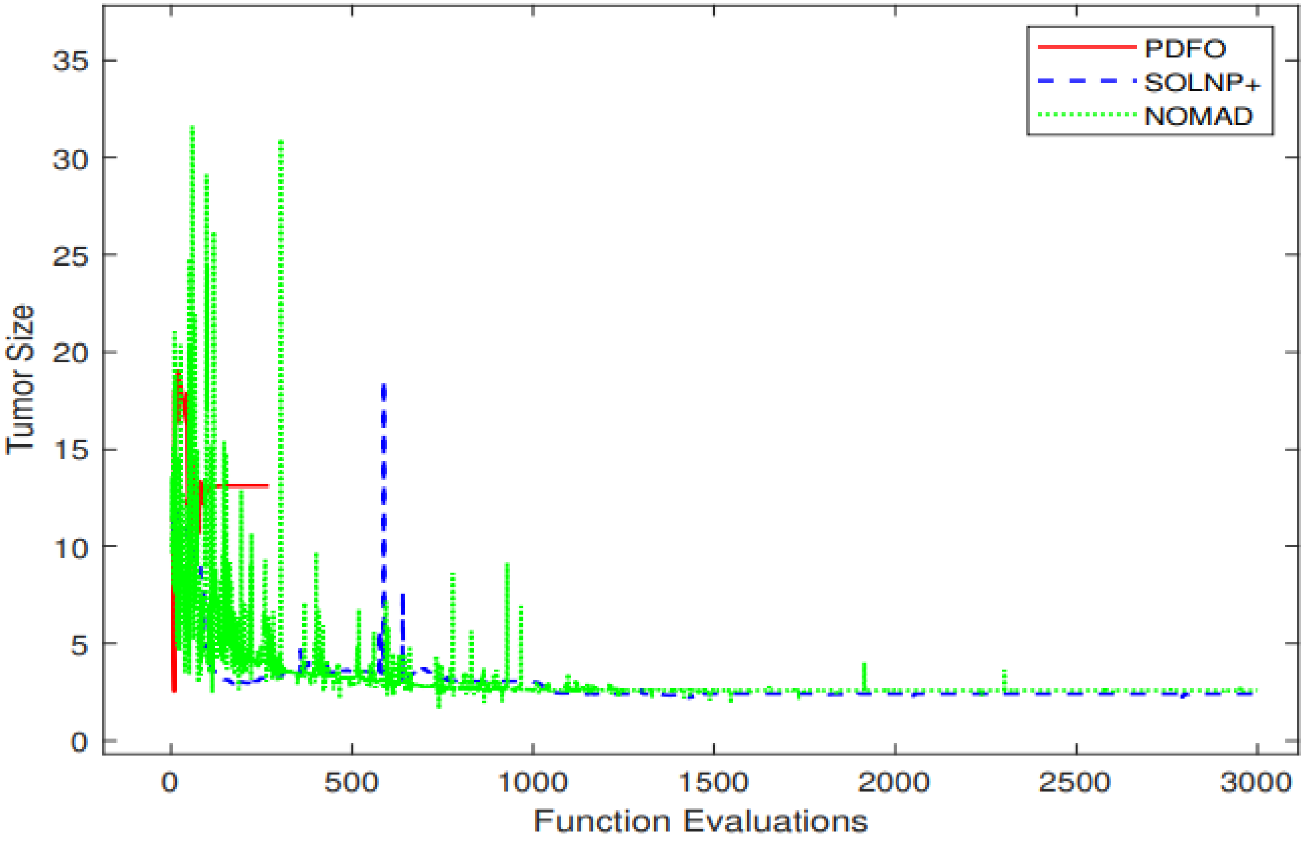


Figure 1: Convergence histories of the objective value.

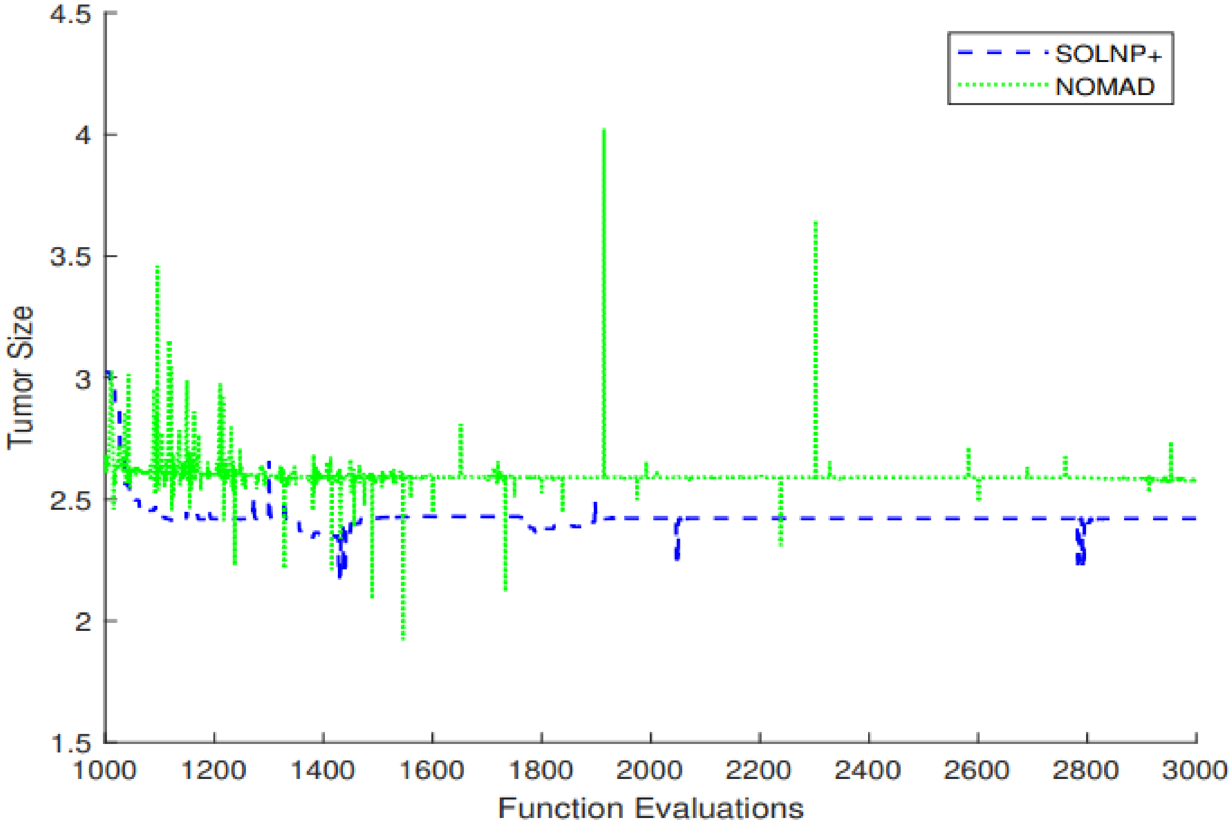


Figure 2: Convergence histories of the objective value after 1000 evaluations.

PART III:

Large-Scaled Unconstrained Optimization

Shortcoming of Finite-Difference

- For finite-difference methods and other traditional zero-order methods, the number of function query **increases linearly** with the problem dimension.
- Traditional methods may not perform well for solving **high dimensional** problems.
- The dimension in machine learning problems can be **extremely high**.

RMP: Multi-Point Random Perturbation

- Multipoint ZO Gradient Estimates (Duchi, 2014)

$$\hat{\nabla} f(x) := \frac{\phi(n)}{\delta b} \sum_{i=1}^{b} \left[\left(f(x + \delta u_i) - f(x) \right) u_i \right]$$

$$\mathbb{E}[\hat{\nabla} f(x)] = \nabla f_\delta(x)$$

$$f_\delta(x) = \mathbb{E}_u[f(x + \delta u)]$$

where u_i is i.i.d. **random direction**.

- Advantage: **Fewer** function queries to evaluate the gradient.

SOLNP+: Adopt Two Strategies

- With gradient estimates, SOLNP+ implements ZO version of
  - ZO-RMP (Ghadimi, 2013; Duchi, 2014), or
  - ZO-BCD, more recent research see (Cai, 2021)
- DRSOM (Zhang, 2022) with interpolation

S. Ghadimi and G. Lan, “Stochastic first-and zeroth-order methods for nonconvex stochastic programming,” *SIAM J. Optimiz.*, vol. 23, no. 4, pp. 2341–2368, 2013. doi: 10.1137/120880811

J.C.Duchi, M. I. Jordan, M. J. Wainwright, and A. Wibisono, “Optimal rates for zero-order convex optimization: The power of two function evaluations,”IEEE Trans.Inf Theory, vol.61,no.5,pp.2788-2806,2015.doi: 10.1109/TIT.2015.2409256.

Cai, HanQin, et al. "A zeroth-order block coordinate descent algorithm for huge-scale black-box optimization." *International Conference on Machine Learning*. PMLR, 2021.

Zhang, Chuwen, et al. "DRSOM: A Dimension Reduced Second-Order Method and Preliminary Analyses." *arXiv preprint arXiv:2208.00208* (2022).

SOLNP+: Apply RMP and RBCD

- Randomized Multiple Point Estimator:
  - $u_i \sim N(0, R^d)$
  - u_i Rademacher Random Variable(each element ∈ $\{-1, +1\}$)

- Randomized Block Coordinate Descent Estimator:
  - Randomly select some block
  - Use e_i (vector with zero components except that dimension i is 1), $i \in Block$ to estimate gradient

- Update rule: $x_{k+1} = x_k - \eta \hat{\nabla} f(x_k)$

R. Sun and Y. Y, "Worst-case complexity of cyclic coordinate descent: O(n^2) gap with randomized version." Mathematical Programming, Volume 185, 487-520, 2021.

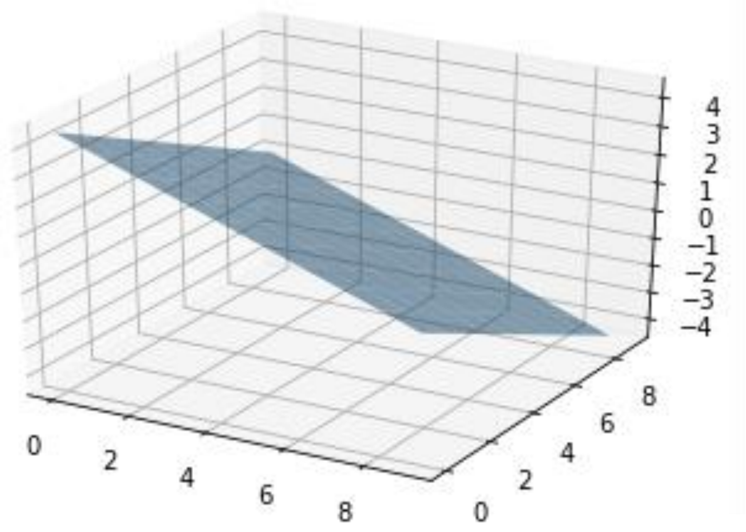
SOLNP+: ZO-DRSOM I

- Dimension reduced second-order method (DRSOM) automatically adjust stepsize α_k of directions $x_{k+1} = x_k + \begin{bmatrix} -g_k & d_k \end{bmatrix} \begin{bmatrix} \alpha_k^1 \\ \alpha_k^2 \end{bmatrix} = x_k + D_k \alpha_k$
  - gradient $g_k = \nabla f(x_k)$
  - momentum $d_k = x_k - x_{k-1}$
  - other directions added in D_k ...
- DRSOM solves small **trust region subproblem** (TRS) to determine α_k

$$\begin{aligned} \min_{\alpha \in \mathbb{R}^2} \ & f(x_k) + g_k^\top D_k \alpha + \frac{1}{2} \alpha^\top D_k^\top H_k D_k \alpha \\ \text{s.t.} \ & \|D_k \alpha\|_2 \leq \Delta \end{aligned}$$

SOLNP+: ZO-DRSOM II

- We use two way to calculate the gradient.
  - ZO-RMP-DRSOM: Use Randomized Multi-Point Method
  - ZO-RBCD-DRSOM: Use Randomized Block Coordinate Descent
- ZO-DRSOM uses interpolation in the 2-dimensional subspace to approximate $D_k^T H_k D_k$ in TRS.



SOLNP+: ZO-DRSOM III

- Advantage of ZO-DRSOM
  - Able to use partial second-order information
  - Do not need to do interpolation in whole space
  - Adaptively choose the step size.

Experiments in Large Problems: Rosenbrock

- Rosenbrock function is a well-known nonconvex functions in the form of

$$f(\boldsymbol{x}) = \sum_{i=1}^{n-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$$

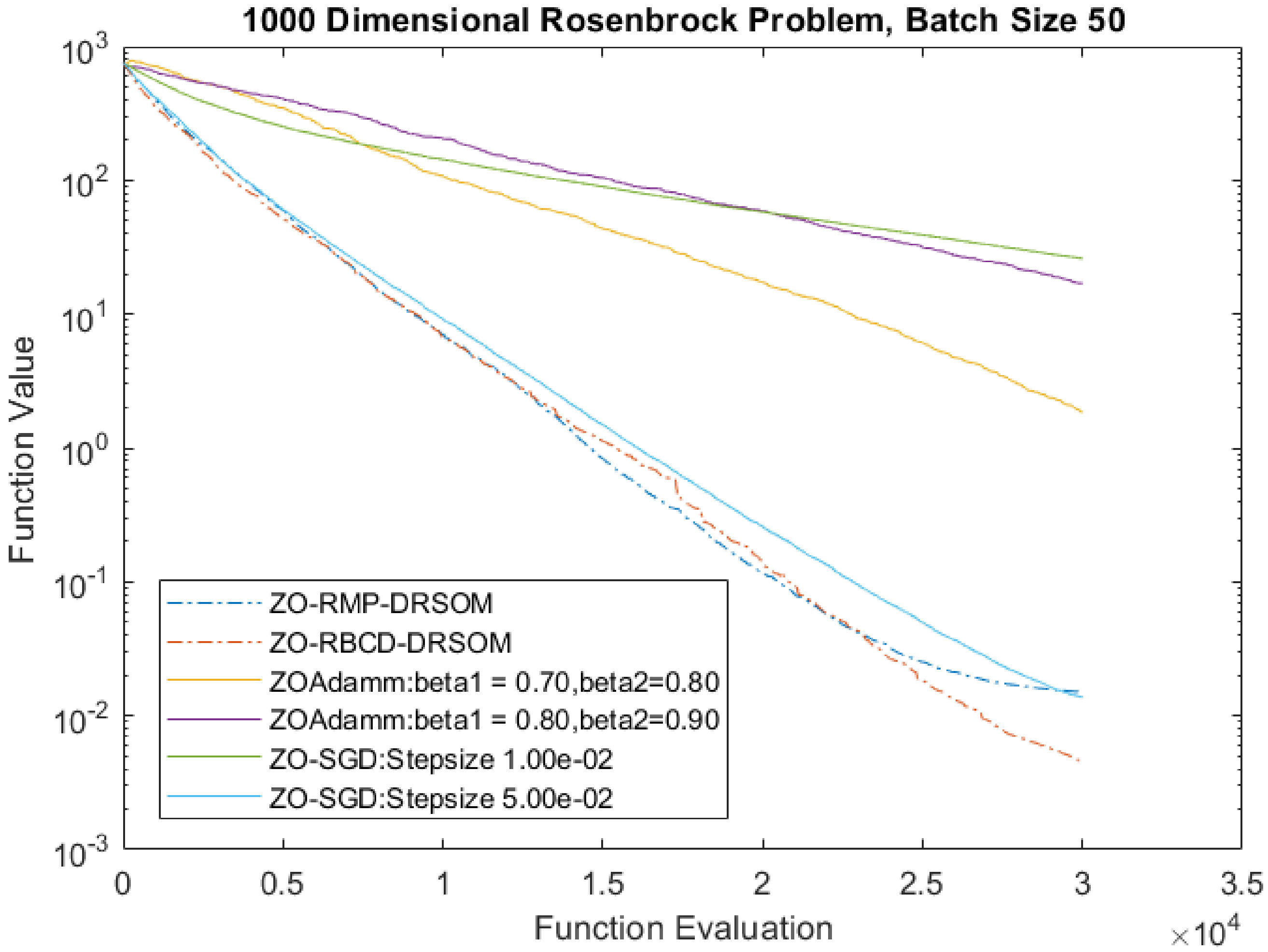
- ZO-Adamm, ZO-SGD and ZO-DRSOM are tested in a 1200 dimensional Rosenbrock problem.
  - Batch size 50
  - Each experiment is repeated for 10 times.

Chen, Xiangyi, et al. "Zo-adamm: Zeroth-order adaptive momentum method for black-box optimization." *Advances in neural information processing systems* 32 (2019).

Experiments in Large Problems: Rosenbrock

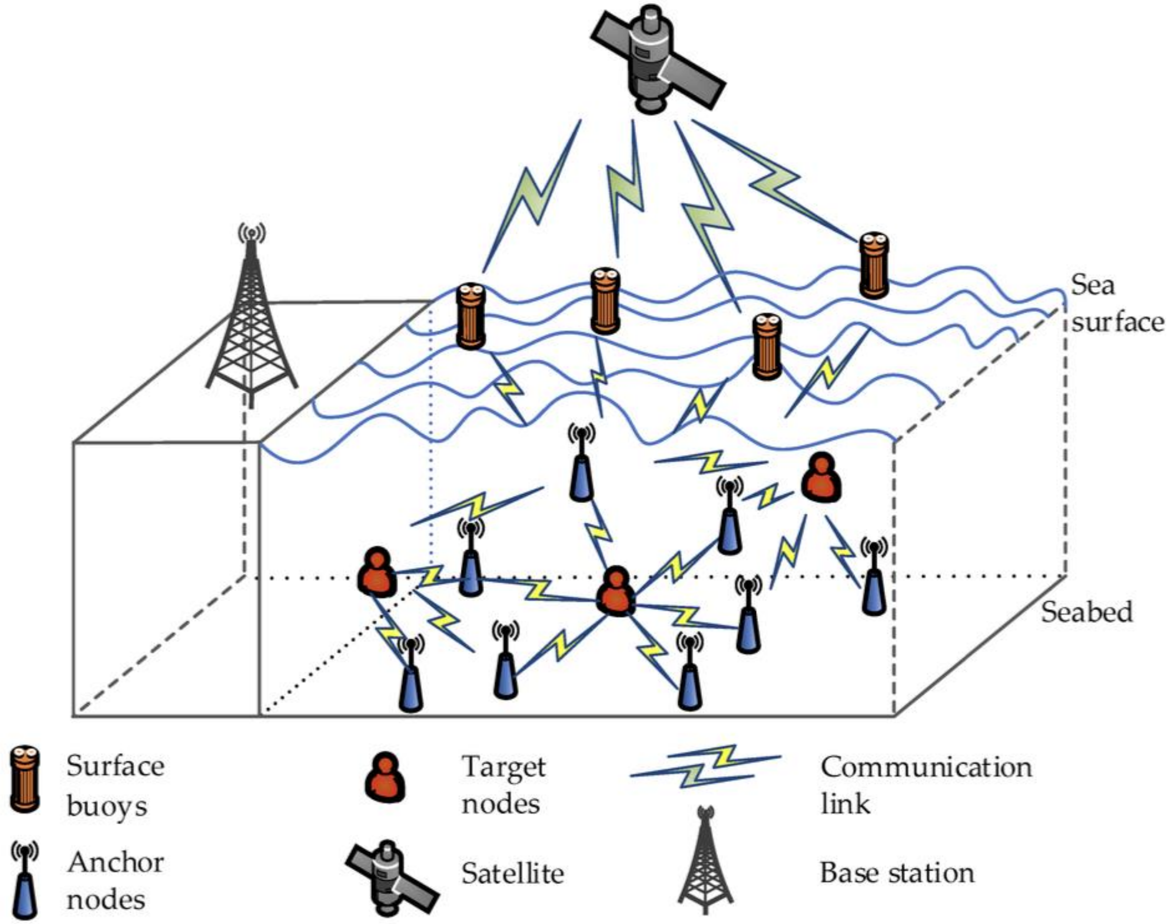
- ZO-RMP-DRSOM, ZO-RBCD-DRSOM and ZO-SGD decrease most smoothly. However, inappropriate parameters lead to worse performance of ZO-SGD and

  ZO-ADAMM.



Experiments in Large-Scale Problems: SNL I

- Sensor network localization (SNL) is widely used in GPS and location services.
- We known
  - Some distances between sensors
  - Some distances between sensors and anchors
  - Position of anchors
- We want to know
  - Position of sensors



Experiments in Large-Scale Problems: SNL II

- SNL is the problem to recover unknown locations of sensors given some distances.
- Given anchors $a_k \in \mathbb{R}^d$, distance $d_{ij} \in N_x$ and $\hat{d}_{kj} \in N_a$, we want to find $x_i \in \mathbb{R}^d$ such that

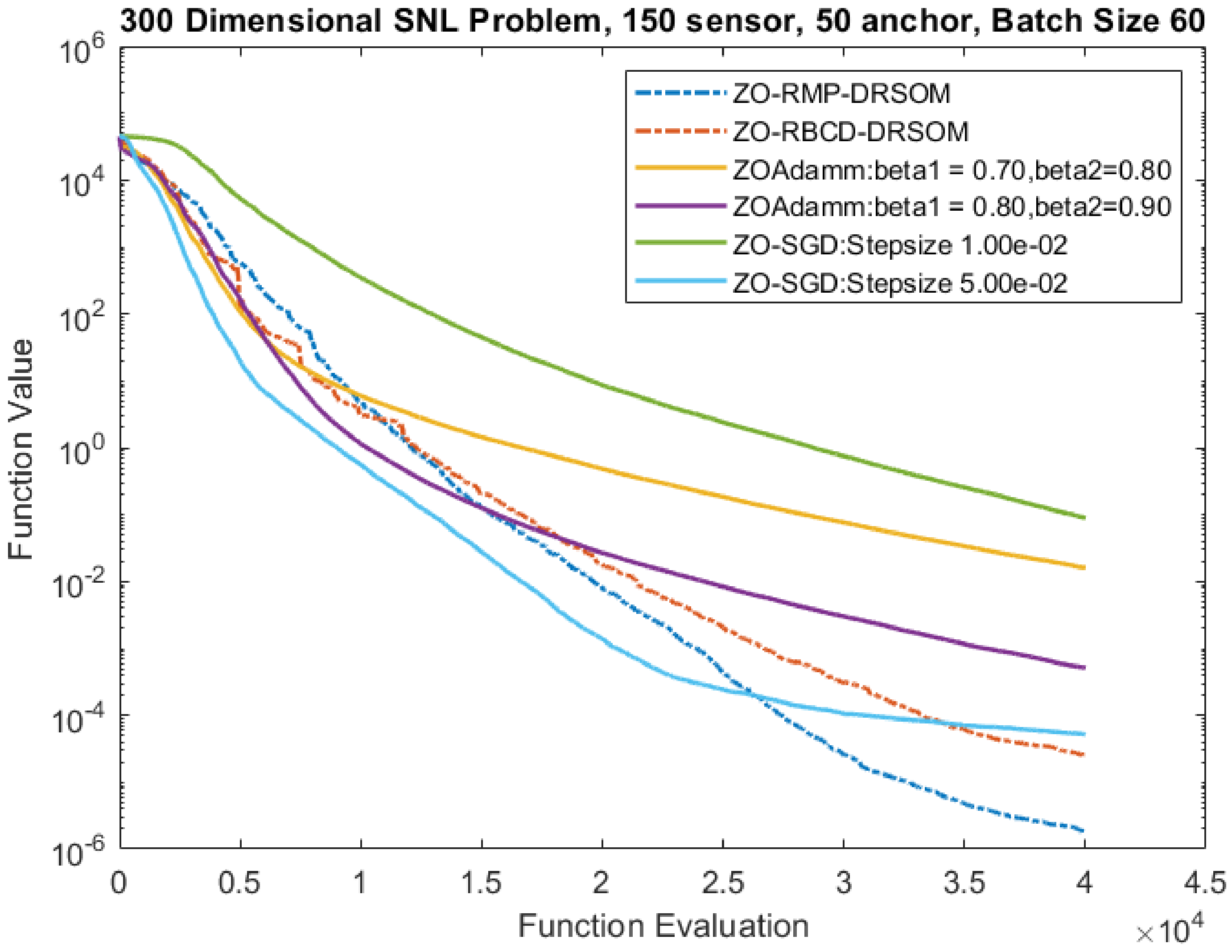
$$\|x_i - x_j\|_2^2 = d_{ij}^2, \quad \forall (i,j) \in N_x$$
$$\|a_k - x_j\|_2^2 = \hat{d}_{kj}^2, \quad \forall (k,j) \in N_a$$

- Reformulate SNL as nonconvex optimization problem

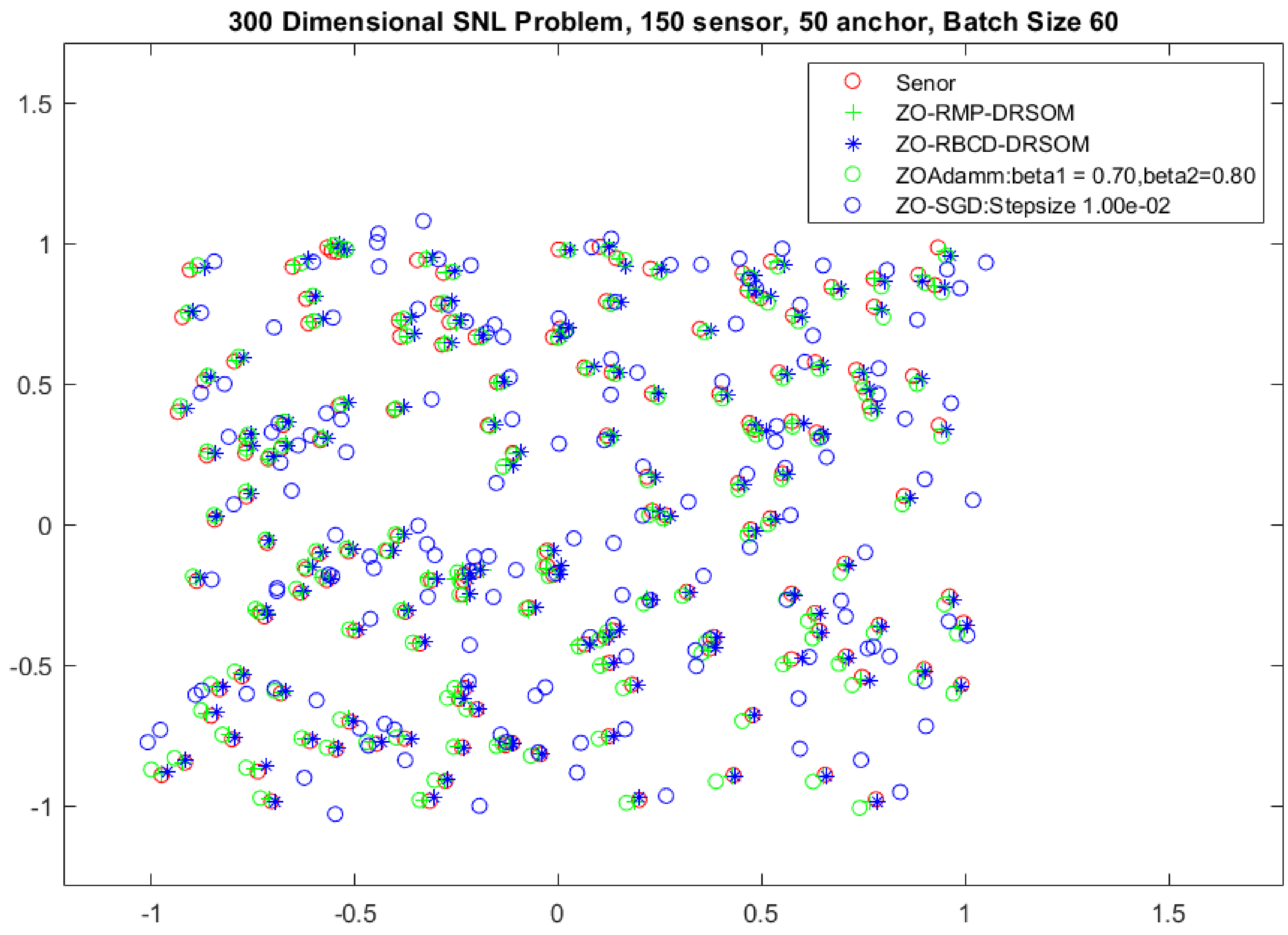
$$\min_{x_i \in \mathbb{R}^d, \forall i} \; f(x) = \sum_{(i,j) \in N_x} (\|x_i - x_j\|_2^2 - d_{ij}^2)^2 + \sum_{(k,j) \in N_a} (\|a_k - x_j\|_2^2 - \hat{d}_{kj}^2)^2$$

Experiments in Large-Sacle Problems: SNL III

- ZO-Adam, ZO-SGD, ZO-RBCD-DRSOM and ZO-RMP-DRSOM are tested in a 150-sensor SNL problem.
  - Batch size 60
  - Each experiment is repeated for 10 times.



Experiments in Large-Scale Problems: SNL IV



Advantage of SOLNP+

- Able to make use of dual information.
- Provide estimation of both primal and dual solutions.
- It seems Faster in speed.
- It seems Robust under noise.

Thank you!