Advanced Subspaces for Large-Scale Optimization with Local Approximation Strategy

Yitong He, Xi'an Jiaotong University, yitong.he@stu.xjtu.edu.cn
(joint work with Pengcheng Xie (Lawrence Berkeley National Laboratory))

Introduction

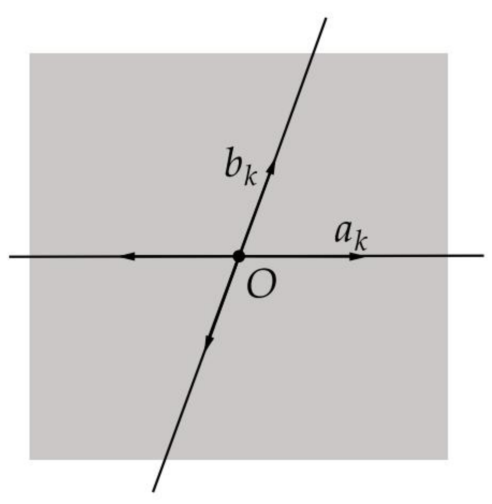
Main goal: find x^* such that

$$f(x^*) = \min_{x \in \mathbb{R}^n} f(x)$$

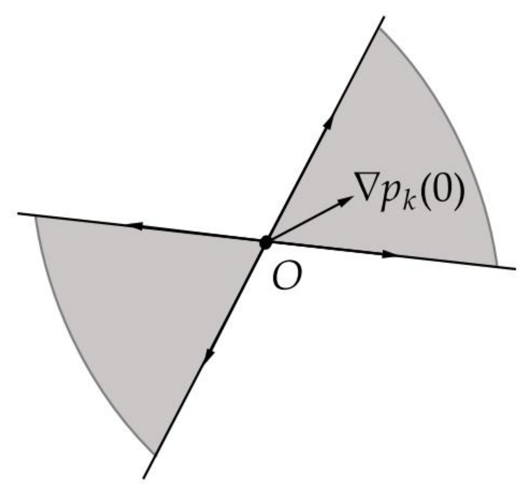
Algorithm 1 Subspace-based optimization method

Input: object function $f : \mathbb{R}^n \to \mathbb{R}$, $\boldsymbol{x}_0 \in \mathbb{R}^n$, $k = 0$.

1: Choose a subspace $\mathcal{S}_k \subset \mathbb{R}^n$.
2: Calculate $\boldsymbol{x}_{k+1} \approx \text{argmin}\{f(\boldsymbol{x}) :\ \boldsymbol{x} \in \boldsymbol{x}_k + \mathcal{S}_k\}$.
3: Let $k = k + 1$.



$S_k = span\{a_k, b_k\}$



$S_k = \bigcup_{s \in \mathcal{R}} span\{\nabla p_k(s)\}$

Main Results

The model-driven truncated subspaces adopted in **MD-LAMBO** is defined as follows. Table 1 and 2 lists the sixteen choices for $\mathcal{S}_k$.

$$p_k(\boldsymbol{s}) = Q_k(\boldsymbol{s}) + \varepsilon\|\boldsymbol{s}\|^3 \xrightarrow{\mathcal{F}} \begin{cases} \mathcal{S}_k = \bigcup_{\boldsymbol{s}\in\mathcal{R}} span\{\nabla p_k(\boldsymbol{s})\} \\ \mathcal{S}_k = span\left\{\bigcup_{\boldsymbol{s}\in\mathcal{R}}\{\nabla p_k(\boldsymbol{s})\}\right\} \end{cases}$$

Table 1: Type-1 truncated subspaces

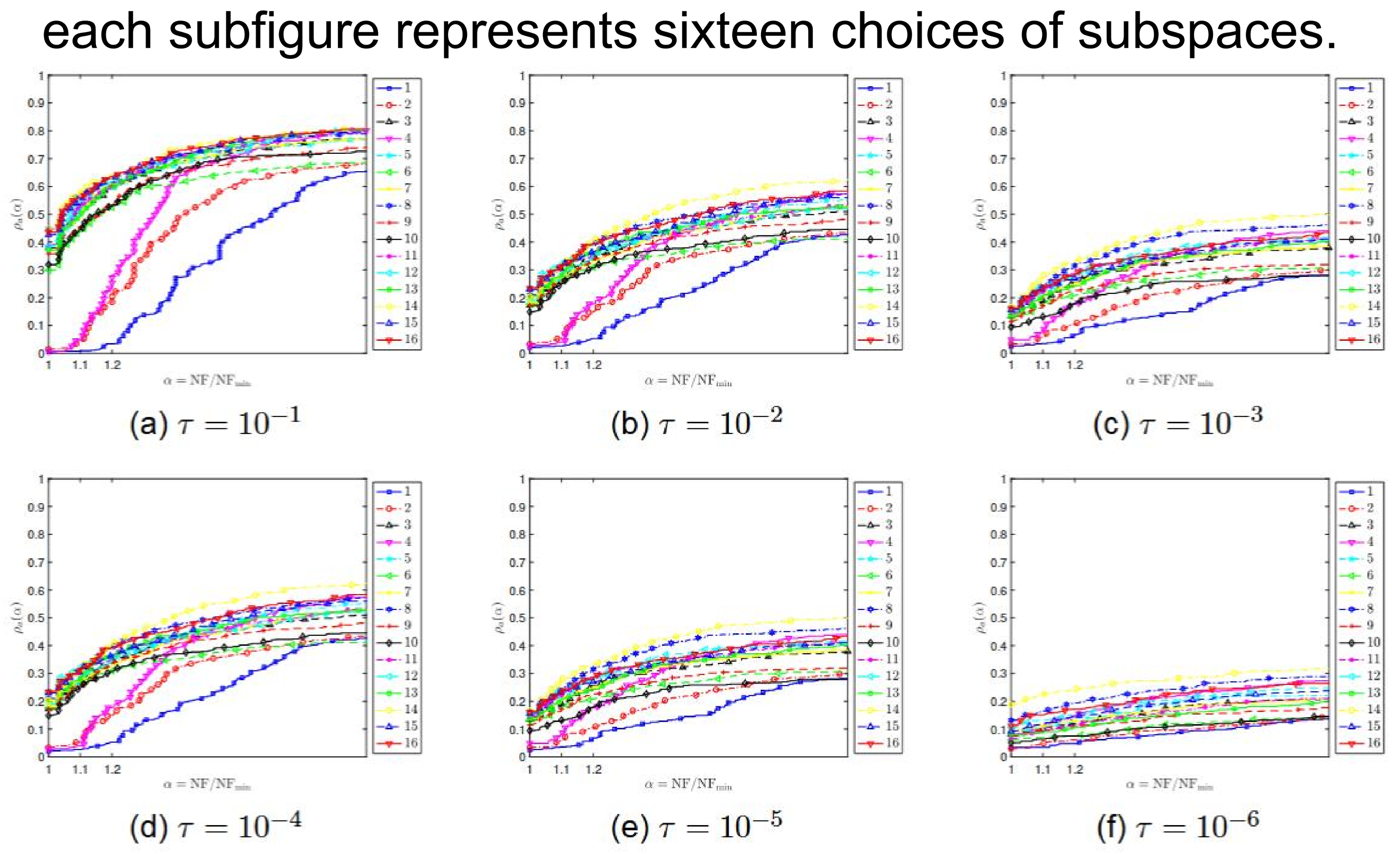
$\mathcal{R}$	$\mathcal{B}(\boldsymbol{0};1)$	$span\{\nabla p_k(\boldsymbol{0})\}$	$\{\boldsymbol{s}_1, \boldsymbol{s}_2\}$	$span\{\boldsymbol{s}_1, \boldsymbol{s}_2\}$
$\varepsilon = 0$	$\mathcal{S}_{\mathcal{B}}^{(1,0)}$	$\mathcal{S}_{l}^{(1,0)}$	$\mathcal{S}_{\boldsymbol{s}_1,\boldsymbol{s}_2}^{(1,0)}$	$\mathcal{S}_{\langle\boldsymbol{s}_1,\boldsymbol{s}_2\rangle}^{(1,0)}$
$\varepsilon > 0$	$\mathcal{S}_{\mathcal{B}}^{(1,\varepsilon)}$	$\mathcal{S}_{l}^{(1,\varepsilon)}$	$\mathcal{S}_{\boldsymbol{s}_1,\boldsymbol{s}_2}^{(1,\varepsilon)}$	$\mathcal{S}_{\langle\boldsymbol{s}_1,\boldsymbol{s}_2\rangle}^{(1,\varepsilon)}$

Table 2: Type-2 truncated subspaces

$\mathcal{R}$	$\mathcal{B}(\boldsymbol{0};1)$	$span\{\nabla p_k(\boldsymbol{0})\}$	$span\{\boldsymbol{s}_1\} \cup span\{\boldsymbol{s}_2\}$	$span\{\boldsymbol{s}_1, \boldsymbol{s}_2\}$
$\varepsilon = 0$	$\mathcal{S}_{\mathcal{B}}^{(2,0)}$	$\mathcal{S}_{l}^{(2,0)}$	$\mathcal{S}_{\langle\boldsymbol{s}_1\rangle,\langle\boldsymbol{s}_2\rangle}^{(2,0)}$	$\mathcal{S}_{\langle\boldsymbol{s}_1,\boldsymbol{s}_2\rangle}^{(2,0)}$
$\varepsilon > 0$	$\mathcal{S}_{\mathcal{B}}^{(2,\varepsilon)}$	$\mathcal{S}_{l}^{(2,\varepsilon)}$	$\mathcal{S}_{\langle\boldsymbol{s}_1\rangle,\langle\boldsymbol{s}_2\rangle}^{(2,\varepsilon)}$	$\mathcal{S}_{\langle\boldsymbol{s}_1,\boldsymbol{s}_2\rangle}^{(2,\varepsilon)}$

Numerical Results

The following figure displays the performance profiles of **MD-LAMBO** under six distinct accuracy τ. The sixteen curves in each subfigure represents sixteen choices of subspaces.



(a) $\tau = 10^{-1}$ (b) $\tau = 10^{-2}$ (c) $\tau = 10^{-3}$

(d) $\tau = 10^{-4}$ (e) $\tau = 10^{-5}$ (f) $\tau = 10^{-6}$

Figure 1: Performance profile of subspaces

Different subspaces drives the algorithm to perform differently. The highly overlap between some curves 6 and 10 indicates that algorithm with truncated subspace has approximately the same capability of problem-solving as the one with classic subspace while the former sometimes performs better.

MD-LAMBO

Model-**D**riven **L**ocal **A**pproximation **M**odel-**B**ased **O**ptimization

Input: object function f, parameters $\gamma_{inc}, \gamma_{dec}$, radius Δ_0, thresholds η, $k = 0$.

At *k*th iteration:

$p_k(x) \approx f(x)$ — Obtaining model function

↓

$S_k = \mathcal{F}(p_k)$ — Generating low-dimensional subspace

↓

$\hat{S}_k = P_k^T(S_k)$, $m_k = \tau_k(p_k)$ — Obtaining projected subspace and model

↓

$s^* = argmin\{m_k(s) : s \in \hat{S}_k\}$ — Solving trust-region subproblem with **STCG**

↓

$x_{k+1} = x_k + P_k s^*$, $\Delta_{k+1} = \gamma\Delta_k$ — Updating the iterates according to the solution

At each iteration, the algorithm solves a trust-region subproblem in $\mathcal{S}_k$ with **STCG**, a modification of truncated conjugate gradient method such that the solution achieves at least one half of maximal function decrease.

Theorem 1 *Let $\boldsymbol{y}^*$ be the solution of trust region subproblem in subspace $\hat{\mathcal{S}}$ and $\boldsymbol{s}^*$ be the global minimizer, then the function value decrease satisfies*

$$m_k(\boldsymbol{0}) - m_k(\boldsymbol{y}^*) \geq \frac{1}{2}\left(m_k(\boldsymbol{0}) - m_k(\boldsymbol{s}^*)\right).$$

With some extra assumptions, such decrease satisfies Cauchy decrease condition, which is one important guarantee of the following convergence result.

Theorem 2 *Suppose $f \in \mathcal{C}^{2+}(\mathbb{R}^n)$, subspace $\mathcal{S}_k$ satisfies $\mathcal{S}_k \neq \mathcal{S}_{\boldsymbol{s}_1,\boldsymbol{s}_2}^{(1,\varepsilon)}$ $(\varepsilon \geq 0)$, $\{\boldsymbol{x}_k\}_{k\geq 0}$ is a sequence obtained by MD-LAMBO. Then for some $\boldsymbol{x}^* \in \mathbb{R}^n$ such that $\nabla f(\boldsymbol{x}^*) = \boldsymbol{0}$, we have*

$$\|f(\boldsymbol{x}_k) - f(\boldsymbol{x}^*)\| \to 0,\ k \to \infty.$$

Discussion

1. By numerical results, algorithms with truncated subspcaces performs as good as classic subspaces.
2. Due to extra boundary conditions, in solving trust-region subproblem, **STCG** always terminates faster than **TCG**. In this way it saves CPU time.

Question 1: how does ε affect the algorithm?
Question 2: how does the swich in different subspaces during consecutive iterations affect the algorithm?

References

[1] Yuan, Ya-xiang. On the Truncated Conjugate Gradient Method. Mathematical Programming. 87, 561-573 (1999).
[2] Cartis, C., Roberts, L. Scalable subspace methods for derivative-free nonlinear least-squares optimization. Math. Program. 199, 461–524 (2023).
[3] Zhang, Z. Scalable Derivative-Free Optimization Algorithms with Low-Dimensional Subspace Techniques. arXiv e-prints, Art. no. arXiv:2501.04536, doi:10.48550/arXiv.2501.04536 (2025).