Blockwise Direct-Search Methods

Haitian Li

Department of Applied Mathematics
The Hong Kong Polytechnic University

Joint with Dr. Zaikun Zhang

ASA2024, Xinxiang, Henan

Derivative-free optimization (DFO): what and when?

What is DFO?

Solve an optimization problem

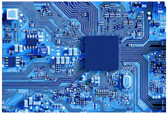
$$\min_{x \in \mathbb{R}^n} f(x)$$

using function values but not derivatives (classical or generalized).

When do we use DFO?

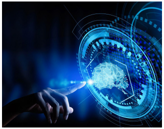
- Derivatives are not available even though f may be smooth.
- "not available": the evaluation is impossible or too expensive.

The applications of DFO



Circuit Design

Photovoltaic



Machine Learning

1. X. Zeng et al., BBGP-sDFO: Batch Bayesian and Gaussian Process Enhanced Subspace Derivative Free Optimization for High-Dimensional Analog Circuit Synthesis, *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 2023.
2. Shokralla et al., Parameter estimation of a photovoltaic array using direct search optimization algorithm. *J. Renew. Sustain. Energy*, 2017.
3. Ghanbari and Scheinberg, Black-box optimization in machine learning with trust region based derivative free algorithm, arXiv:1703.06925, 2017.

Two main classes of DFO methods

1. Direct-search methods based on
   - simplex (Nelder-Mead method)
   - directional search (BFO, PDS, NOMAD)
2. Model-based methods based on
   - trust region (Powell's methods)
   - line search

Methods not covered by these two classes:

Bayesian optimization, genetic algorithms, etc.

Model-based methods v.s. Direct-search methods

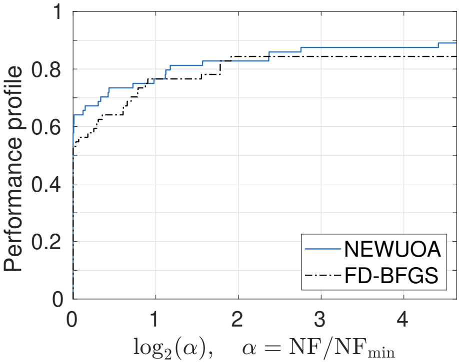
	Model-based	Direct-search
Performance	good	less satisfactory
Implementation	complicated	relatively simple

1. Model-based methods:
   - The optimization process is guided by models.
   - The coupling between modeling and optimization makes the implementation complicated.
2. Direct-search methods:
   - No need to construct models.
   - Iterate is decided by comparing the function values of samples.

An example of DFO solver: NEWUOA

- A derivative-free solver for unconstrained problems
- Developed by M.J.D. Powell
- Widely used by engineers and scientists
- The modernized version (https://github.com/libprima)

NEWUOA: performance is quite good



- FD-BFGS: Forward-finite-difference BFGS method.

NEWUOA: implementation and understanding is HARD

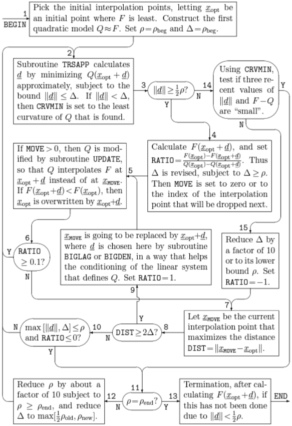


Figure 1: An outline of the method, where Y = Yes and N = No

Framework of NEWUOA

NEWUOA: implementation and understanding is HARD

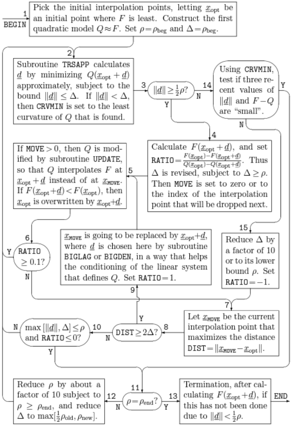


Figure 1: An outline of the method, where Y = Yes and N = No

Framework of NEWUOA

From Powell (2006)

The development of NEWUOA has taken nearly three years. The work was very frustrating ...

Outline

1. Classical direct-search methods

2. Blockwise direct-search methods

3. Implementation and experiments

4. Conclusions and future work

Outline

1. Classical direct-search methods
2. Blockwise direct-search methods
3. Implementation and experiments
4. Conclusions and future work

A classical direct-search framework

What is direct search?

- No explicit models are constructed based on function values.
- Iterations are only decided according to function values.

Algorithm 1: Direct Search (DS) based on sufficient decrease

Input: $x_0 \in \mathbb{R}^n$, $0 < \theta < 1 \leq \gamma$, $\alpha_0 > 0$, $c > 0$, a searching direction set $\mathcal{D} \subset \mathbb{R}^n$.

for $k = 0, 1, \ldots$ **do**

  if $f(x_k + \alpha_k d) < f(x_k) - c\rho(\alpha_k)$ for some $d \in \mathcal{D}$ **then**

    Set $x_{k+1} = x_k + \alpha_k d$ and $\alpha_{k+1} = \gamma\alpha_k$.

  else

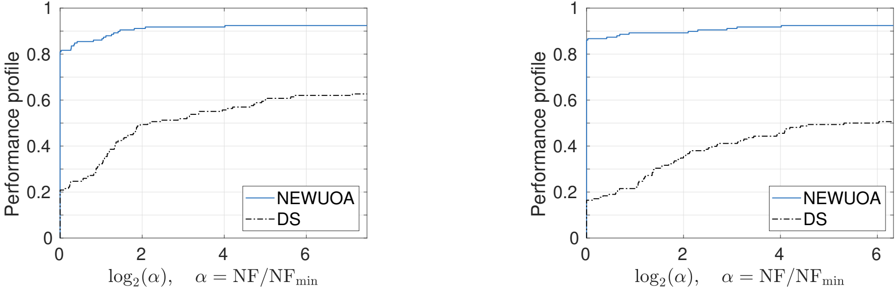
    Set $x_{k+1} = x_k$ and $\alpha_{k+1} = \theta\alpha_k$.

Unsatisfactory performance of direct-search methods

Simple, but performs unsatisfactorily!

Unsatisfactory performance of direct-search methods

Simple, but performs unsatisfactorily!



(a) $\tau = 10^{-3}$ (b) $\tau = 10^{-5}$

(Unconstrained CUTEst problems, $1 \leq n \leq 200$)

Outline

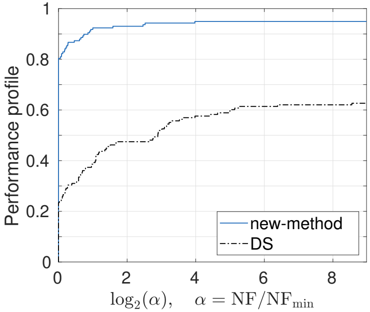
1. Classical direct-search methods

2. Blockwise direct-search methods

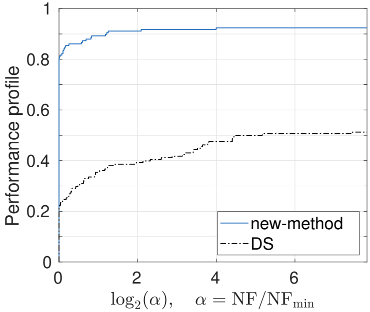
3. Implementation and experiments

4. Conclusions and future work

Obvious improvement!



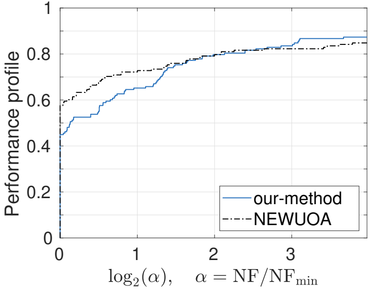
(a) $\tau = 10^{-3}$



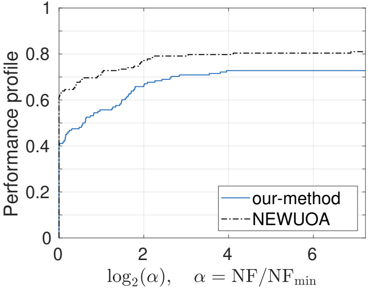
(b) $\tau = 10^{-5}$

(Unconstrained CUTEst problems, $1 \leq n \leq 200$)

Performance of the new method we introduce



(a) $\tau = 10^{-3}$



(b) $\tau = 10^{-5}$

(Unconstrained CUTEst problems, $1 \leq n \leq 200$)

Flaws of the classical direct-search method

Algorithm 1: Direct Search (DS) based on sufficient decrease

Input: $x_0 \in \mathbb{R}^n$, $0 < \theta < 1 \leq \gamma$, $\alpha_0 > 0$, $c > 0$, searching direction set $\mathcal{D} \subset \mathbb{R}^n$.

for $k = 0, 1, \dots$ **do**

    if $f(x_k + \alpha_k d) < f(x_k) - c\rho(\alpha_k)$ for some $d \in \mathcal{D}$ **then**

        Set $x_{k+1} = x_k + \alpha_k d$ and $\alpha_{k+1} = \gamma\alpha_k$.

    else

        Set $x_{k+1} = x_k$ and $\alpha_{k+1} = \theta\alpha_k$.

How to improve it?

Flaws of the classical direct-search method

Algorithm 1: Direct Search (DS) based on sufficient decrease

Input: $x_0 \in \mathbb{R}^n$, $0 < \theta < 1 \leq \gamma$, $\alpha_0 > 0$, $c > 0$, searching direction set $\mathcal{D} \subset \mathbb{R}^n$.

for $k = 0, 1, \ldots$ **do**

  if $f(x_k + \alpha_k d) < f(x_k) - c\rho(\alpha_k)$ for some $d \in \mathcal{D}$ **then**

    Set $x_{k+1} = x_k + \alpha_k d$ and $\alpha_{k+1} = \gamma\alpha_k$.

  else

    Set $x_{k+1} = x_k$ and $\alpha_{k+1} = \theta\alpha_k$.

How to improve it?

- Divide the searching direction set into many blocks.
- Each block has its own step size.

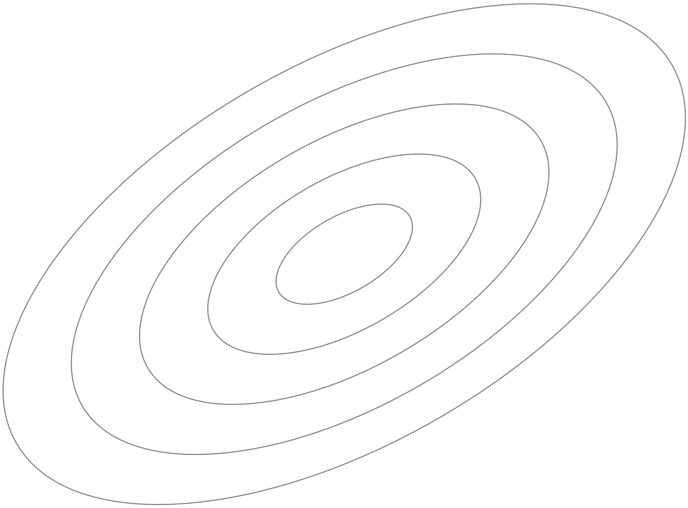
A cyclic framework of blockwise direct-search method

Algorithm 2: Cyclic Blockwise Direct Search (CBDS)

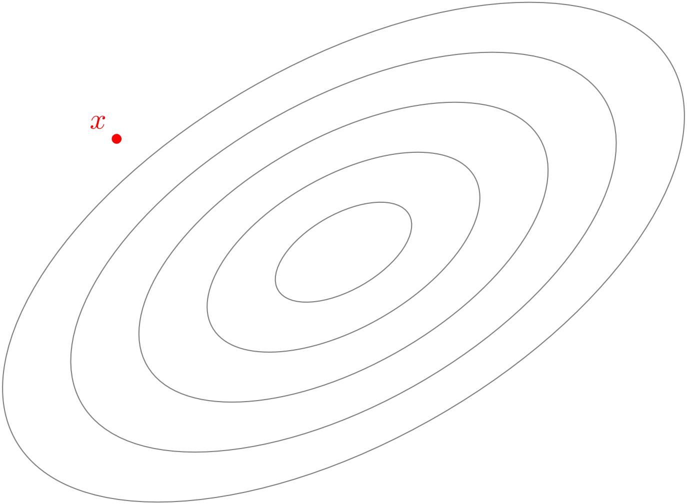
Input: $x_0 \in \mathbb{R}^n$, $0 < \theta < 1 \le \gamma$, $\alpha_0^1, \ldots, \alpha_0^m \in (0, \infty)$, $c > 0$, a searching direction set $\mathcal{D} = \cup_{i=1}^m \mathcal{D}^i \subset \mathbb{R}^n$.

for $k = 0, 1, \ldots$ **do**
- Set $y_k^1 = x_k$.
- **for** $i = 1, \ldots, m$ **do**
  - **if** $f(y_k^i + \alpha_k^i d_k^i) < f(y_k^i) - c\rho(\alpha_k^i)$ for some $d_k^i \in \mathcal{D}^i$ **then**
    - Set $y_k^{i+1} = y_k^i + \alpha_k^i d_k^i$ and $\alpha_{k+1}^i = \gamma \alpha_k^i$.
  - **else**
    - Set $y_k^{i+1} = y_k^i$ and $\alpha_{k+1}^i = \theta \alpha_k^i$.
- Set $x_{k+1} = y_k^{m+1}$.

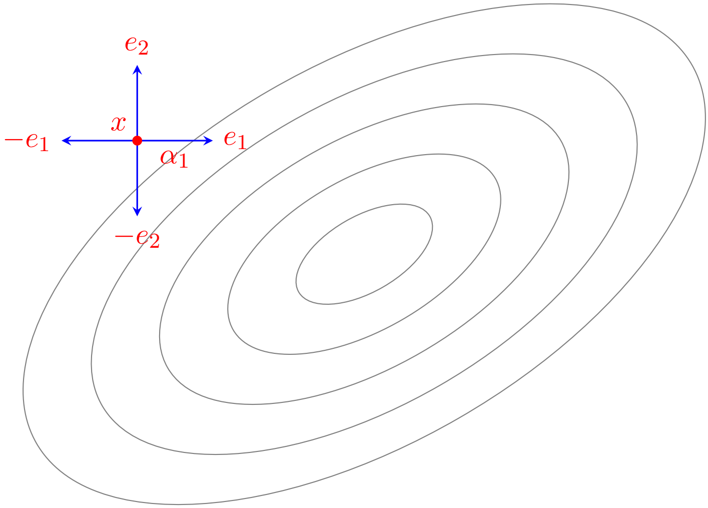
A simple example of the blockwise direct-search method



A simple example of the blockwise direct-search method

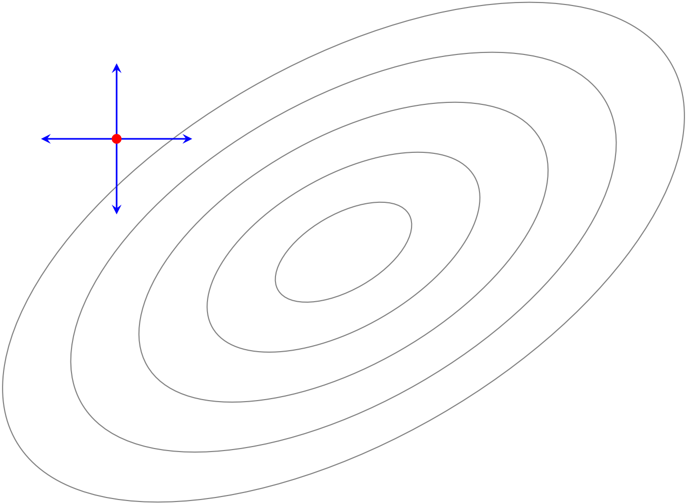


A simple example of the blockwise direct-search method

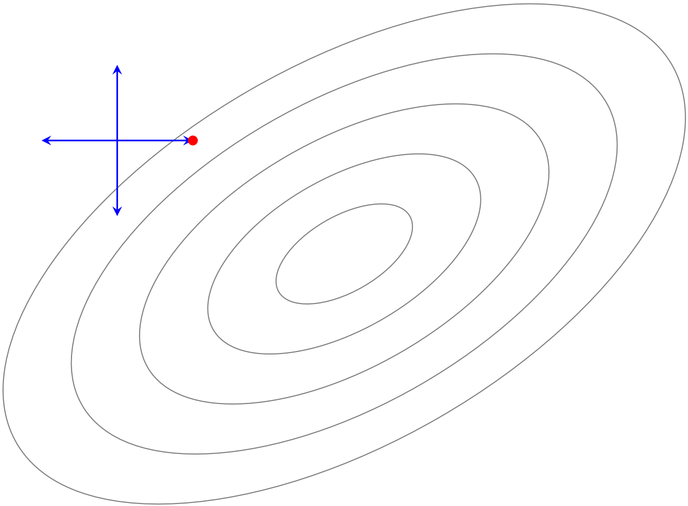


$\mathcal{D}_1 = \{e_1, -e_1\}$ and $\mathcal{D}_2 = \{e_2, -e_2\}$

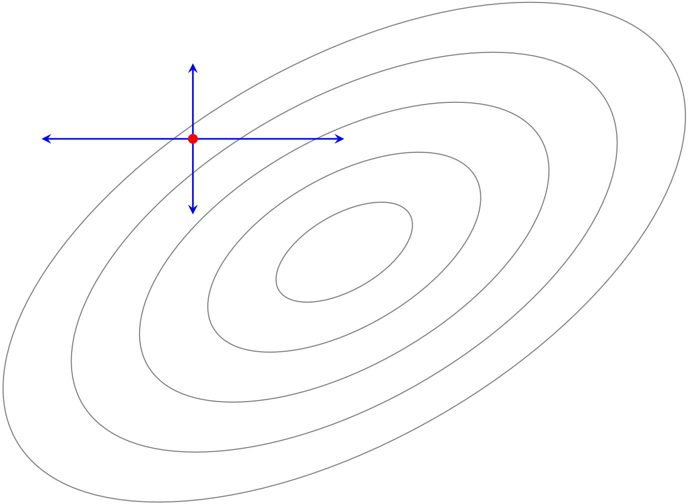
A simple example of the blockwise direct-search method



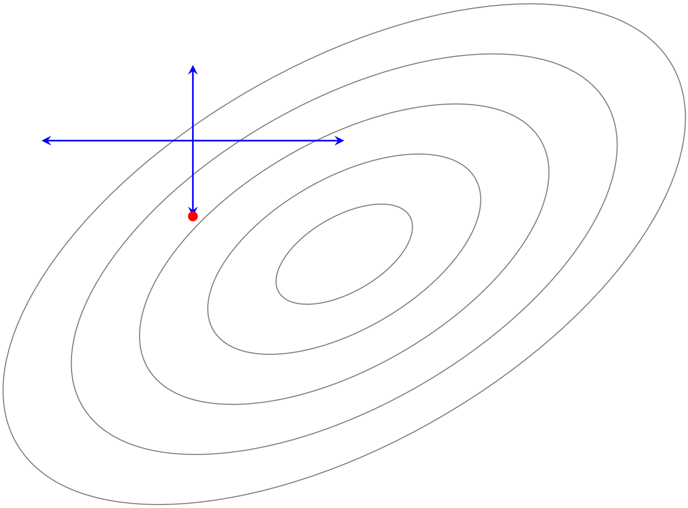
A simple example of the blockwise direct-search method



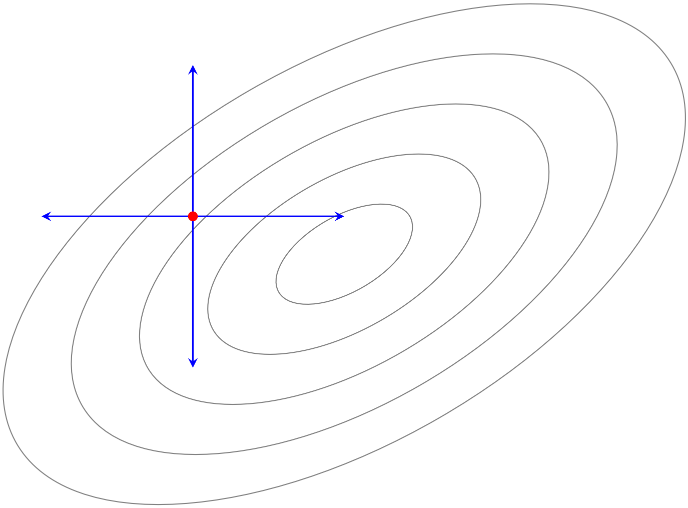
A simple example of the blockwise direct-search method



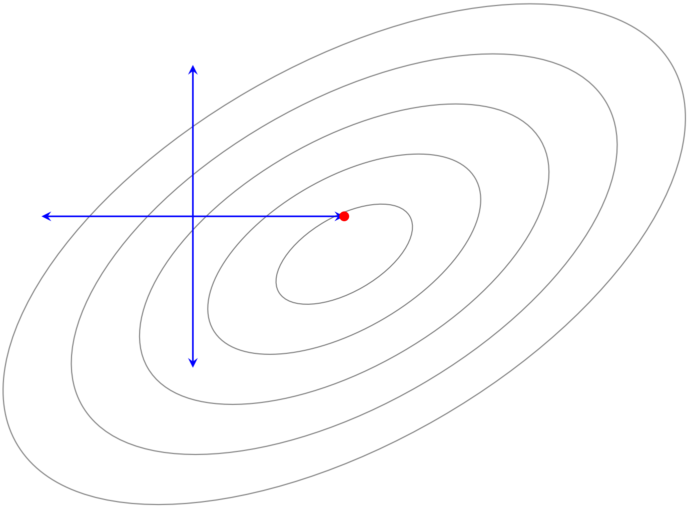
A simple example of the blockwise direct-search method



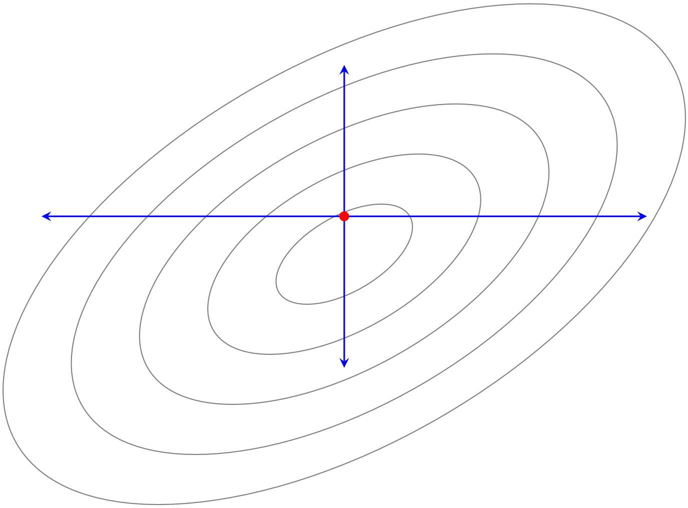
A simple example of the blockwise direct-search method



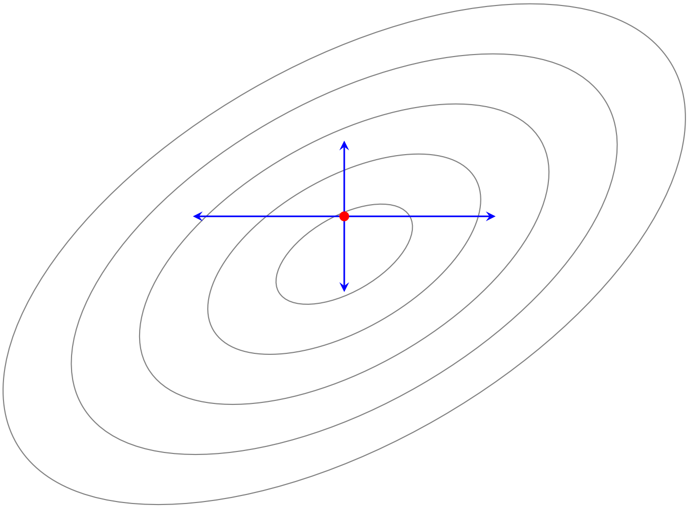
A simple example of the blockwise direct-search method



A simple example of the blockwise direct-search method



A simple example of the blockwise direct-search method



Flexibility of the framework

- The searching direction set: A positive spanning set.
- The division of blocks: any way that "fits the problem".
- The scheme of visiting blocks: Cyclic (Gauss-Seidel), Jacobi, random.

Flexibility of the framework

- The searching direction set: A positive spanning set.
- The division of blocks: any way that "fits the problem".
- The scheme of visiting blocks: Cyclic (Gauss-Seidel), Jacobi, random.

Our implementation takes the following setting as the default:

- $\mathcal{D} = \{e_1, -e_1, \ldots, e_n, -e_n\}$
- $\mathcal{D}^i = \{e_i, -e_i\}$
- Gauss-Seidel scheme

Outline

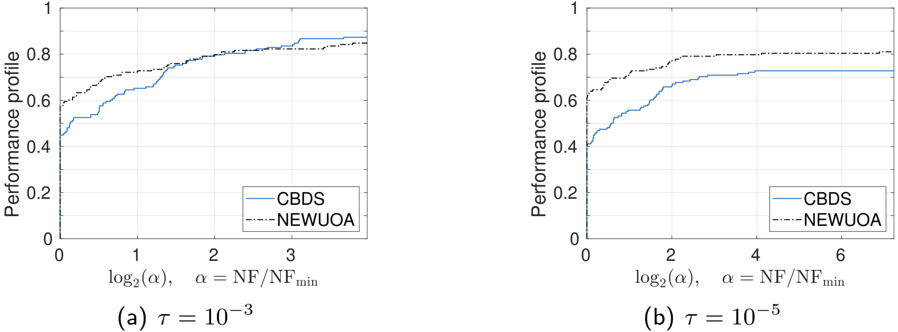
1. Classical direct-search methods

2. Blockwise direct-search methods

3. Implementation and experiments

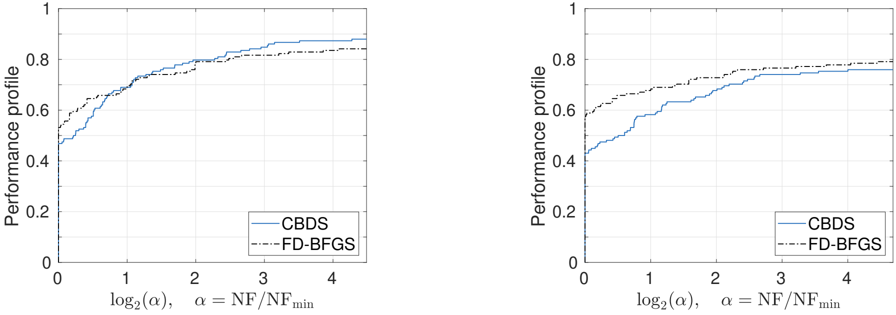
4. Conclusions and future work

Recapped



(Unconstrained CUTEst problems, $1 \leq n \leq 200$)

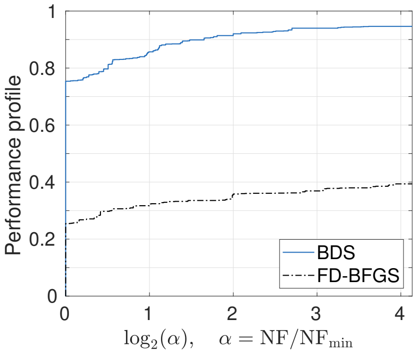
Comparison between BDS and existing DFO solvers



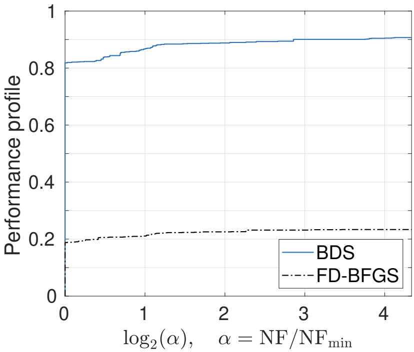
(a) $\tau = 10^{-3}$ (b) $\tau = 10^{-5}$

(Unconstrained CUTEst problems, $1 \leq n \leq 200$)

BDS v.s. FD-BFGS



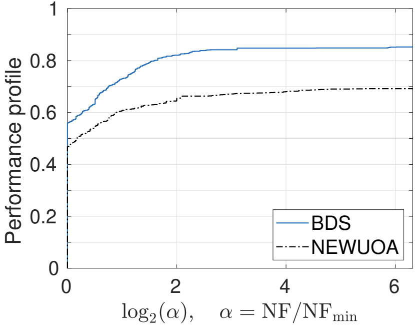
$\tau = 10^{-3}$



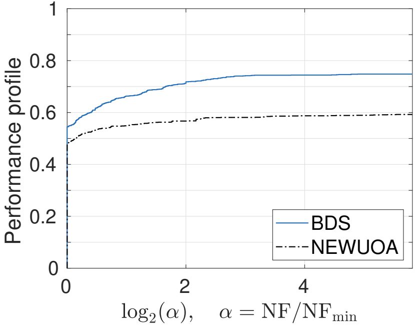
$\tau = 10^{-5}$

Set $h = \sqrt{(\max|f|, 1)\sigma}$ for FD-BFGS.

BDS v.s. NEWUOA

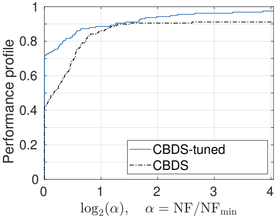


$\tau = 10^{-3}$

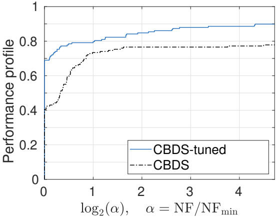


$\tau = 10^{-5}$

Self-tuning can improve the performance! [1]



$\tau = 10^{-3}$



$\tau = 10^{-5}$

In our experiments:

- problem set: unconstrained problems from CUTEst
- dimensions: $1 \le n \le 200$
- hyperparameters: γ, θ, c
- default values: $\gamma = 2$, $\theta = 0.5$, $c =$ eps (machine precision)

[1] Motivated by BFO, A Note on Using Performance and Data Profiles for Training Algorithms, ACM Transactions on Mathematical Software, 2019, Porcelli, M. and Toint, Ph. L.

Self-tuning can improve the performance!

$$\min_{x \in \mathbb{R}^n} f(P_\Omega(x)) + \lambda r(x),$$

where

- $f(x)$: the difference of the integral in the performance profile
- Ω: the feasible set for the hyperparameters
- $P_\Omega(x)$: the projection from x to Ω
- λ: the penalty parameter
- $r(x)$: the residue of x in Ω

Outline

1. Classical direct-search methods

2. Blockwise direct-search methods

3. Implementation and experiments

4. Conclusions and future work

Structured nonsmooth problems

$$\min_{x \in \mathbb{R}^n} f(x) + \Phi(x)$$

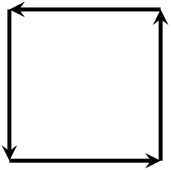
- f is smooth
- Φ is nonsmooth but separable with respect to the blocks

Examples:

- l_p-regularized problems
- bound-constrained problems

Is CBDS convergent?

- We do not know the answer yet.
- The analysis of cyclic methods is challenging.
- Is it possible that the vanilla version of CBDS is not convergent?
- Powell's non-convergent example for cyclic coordinate descent method.



limiting behavior

Conclusions

1. Blockwise Direct Search (BDS) is a substantial improvement over the classical direct search method (based on sufficient decrease)
2. BDS performs well in the following tests:
   - Noise-free problems with a moderate size $(6 \leq n \leq 200)$ and a convergence tolerance that is not too small $(10^{-1} \leq \tau \leq 10^{-5})$
   - Noisy problems with a moderate noise level $(10^{-3} \leq \sigma \leq 10^{-1})$ and a convergence tolerance that is comparable with the noise level
3. BDS is robust under noise without any noise-handling techniques
4. BDS can be tuned to improve its performance
5. BDS is open-source and easy to use

Future work

- Convergence and worst-case complexity (an adapted framework?)
- Make use of the existing iterates (finite difference or interpolation)
- Extend our implementation to other languages (Python, Julia, etc.)

BDS on GitHub

- tested continuously via GitHub Actions
- tested under different platforms

Thank you!

References I

- C. Audet and J. E. Dennis Jr.
  Analysis of generalized pattern searches.
  SIAM J. Optim., 13:889–903, 2002.
- C. Audet and J. E. Dennis Jr.
  Mesh adaptive direct search algorithms for constrained optimization.
  SIAM J. Optim., 17:188–217, 2006.
- C. Audet and D. Orban.
  Finding optimal algorithmic parameters using derivative-free optimization.
  SIAM Journal on Optimization, 17(3):642–664, 2006.
- A. S. Bandeira, K. Scheinberg, and L. N. Vicente.
  Convergence of trust-region methods based on probabilistic models.
  SIAM J. Optim., 24:1238–1264, 2014.

References II

- N. I. M. Gould, D. Orban, and Ph. L. Toint.
  CUTEst: a constrained and unconstrained testing environment with safe threads for mathematical optimization.
  Comput. Optim. Appl., 60:545–557, 2015.
- S. Gratton, C. W. Royer, L. N. Vicente, and Z. Zhang.
  Direct search based on probabilistic descent.
  SIAM J. Optim., 25:1515–1541, 2015.
- Tianchen Gu, Wangzhen Li, Aidong Zhao, Zhaori Bi, Xudong Li, Fan Yang, Changhao Yan, Wenchuang Hu, Dian Zhou, Tao Cui, et al.
  Bbgp-sdfo: Batch bayesian and gaussian process enhanced subspace derivative free optimization for high-dimensional analog circuit synthesis.
  IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2023.

References III

- T. G. Kolda, R. M. Lewis, and V. Torczon.
  Optimization by direct search: New perspectives on some classical and modern methods.
  SIAM Rev., 45:385–482, 2003.
- M. Porcelli and Ph. L. Toint.
  BFO, a trainable derivative-free brute force optimizer for nonlinear bound-constrained optimization and equilibrium computations with continuous and discrete variables.
  ACM Trans. Math. Software, 44:6:1–6:25, 2017.
- M. J. D. Powell.
  The NEWUOA software for unconstrained optimization without derivatives.
  In G. Di Pillo and M. Roma, editors, *Large-scale Nonlinear Optimization*, pages 255–297. Springer, Boston, 2006.

References IV

- Michael JD Powell.
  On search directions for minimization algorithms.
  Mathematical programming, 4:193–201, 1973.