Blockwise Direct-Search Methods

Haitian Li

Department of Applied Mathematics
The Hong Kong Polytechnic University

Joint work with Zaikun Zhang

August 17, 2023

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

What is DFO?

Solve an optimization problem

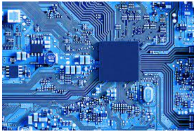
$$\min_{x \in \mathbb{R}^n} f(x)$$

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

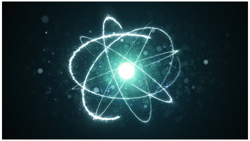
Why do we use DFO?

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

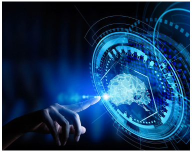
Examples of DFO problems



Circuit Design



Nuclear Energy



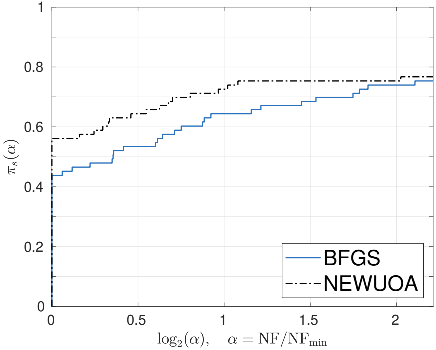
Machine Learning

1. Ciccazzo et al., Derivative-free robust optimization for circuit design. *J. Optim. Theory Appl.*, 2015.
2. More et al., Nuclear energy density optimization. *Phys. Rev.*, 2010.
3. Ghanbari and Scheinberg, Black-box optimization in machine learning with trust region based derivative free algorithm, arXiv:1703.06925, 2017.

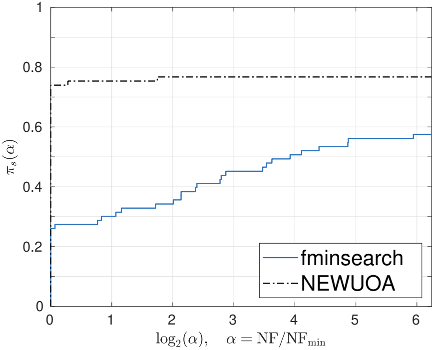
A powerful DFO solver: NEWUOA

- A derivative-free solver for unconstrained problems
- Developed by Powell in Fortran 77
- Modernized by Dr. Zhang (PRIMA)
- Widely used by engineers ans scientists

NEWUOA: performance is excellent



NEWUOA v.s. BFGS



NEWUOA v.s. simplex

(Unconstrained CUTEst problems, $6 \leq n \leq 100$)

NEWUOA: implementation is HARD

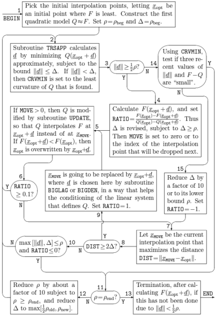


Figure 1: An outline of the method, where Y = Yes and N = No

Outline of NEWUOA's code

NEWUOA: implementation is HARD

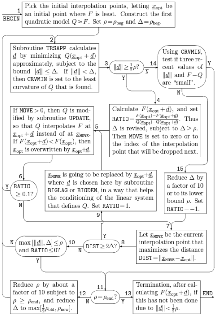


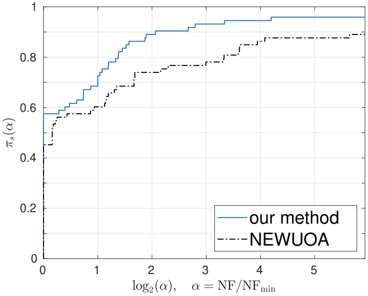
Figure 1: An outline of the method, where Y = Yes and N = No

Outline of NEWUOA's code

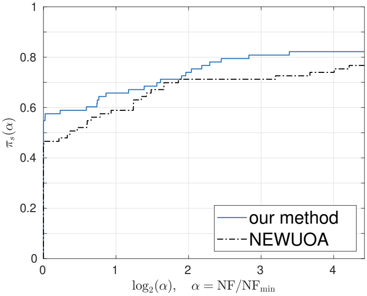
From Powell (2006)

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

Performance of the new method we introduce



(a) $\tau = 10^{-2}$



(b) $\tau = 10^{-4}$

(Unconstrained CUTEst problems, $6 \leq n \leq 100$)

Three months v.s. Three frustrating years!

562 lines of MATLAB code v.s. 2497 lines of Fortran code!

Outline

1. Classical direct-search methods
2. Blockwise direct-search methods
3. Implementation and Experiments
4. Conclusions and Future work

Outline

A classical direct-search framework

What is direct search?

- Not construct any models of objective functions explicitly.
- Only relying on function values to decide the point to visit.

Algorithm 1: Direct Search (DS)

Input: Initial point $x_0 \in \mathbb{R}^n$, parameters of step size $0 < \theta < 1 \leq \gamma$, initial step size $\alpha_0 \in (0, \infty)$, and searching set $\mathcal{D} \subset \mathbb{R}^n$.

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

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

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

  else

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

A classical direct-search framework

What is direct search?

- Not construct any models of objective functions explicitly.
- Only relying on function values to decide the point to visit.

Algorithm 1: Direct Search (DS)

Input: Initial point $x_0 \in \mathbb{R}^n$, parameters of step size $0 < \theta < 1 \leq \gamma$, initial step size $\alpha_0 \in (0, \infty)$, and searching set $\mathcal{D} \subset \mathbb{R}^n$.

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

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

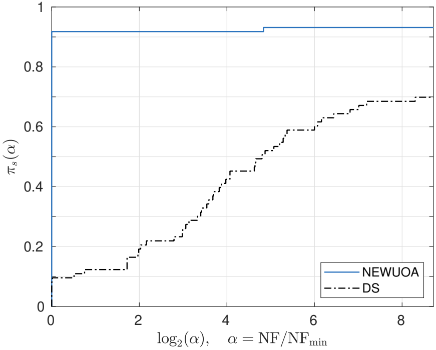
    Set $\alpha_{k+1} = \gamma \alpha_k$ and $x_{k+1} = x_k + \alpha_k d$.

  else

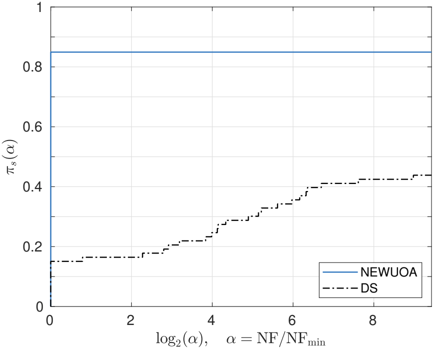
    Set $\alpha_{k+1} = \theta \alpha_k$ and $x_{k+1} = x_k$.

Simple, but performs poorly!

Unsatisfactory performance of direct-search methods



$\tau = 10^{-2}$



$\tau = 10^{-4}$

Flaws of the classical direct-search method

- One step size for every iteration is unfair.
- Rewarding "bad" directions does not make sense.

How to improve it?

Flaws of the classical direct-search method

- One step size for every iteration is unfair.
- Rewarding "bad" directions does not make sense.

How to improve it?

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

Outline

The framework of blockwise direct-search method

Algorithm 2: Cyclic Blockwise Direct Search (CBDS)

Input: Initial point $x_0 \in \mathbb{R}^n$, parameters of step sizes $0 < \theta < 1 \leq \gamma$, initial step sizes $\alpha_0^1, \ldots, \alpha_0^m \in (0, \infty)$, and blockwise searching sets $\mathcal{D}^1, \ldots, \mathcal{D}^m \subset \mathbb{R}^n$.

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

  Set $y_k^1 = x_k$.

  for $i = 1, \ldots, m$ **do**

    if $f(y_k^i + \alpha_k^i d_k^i) < f(y_k^i) - \rho(\alpha_k^i)$ for some $d_k^i \in \mathcal{D}^i$ **then**

      Set $\alpha_{k+1}^i = \gamma \alpha_k^i$ and $y_k^{i+1} = y_k^i + \alpha_k^i d_k^i$.

    else

      Set $\alpha_{k+1}^i = \theta \alpha_k^i$ and $y_k^{i+1} = y_k^i$.

  Set $x_{k+1} = y_k^{m+1}$.

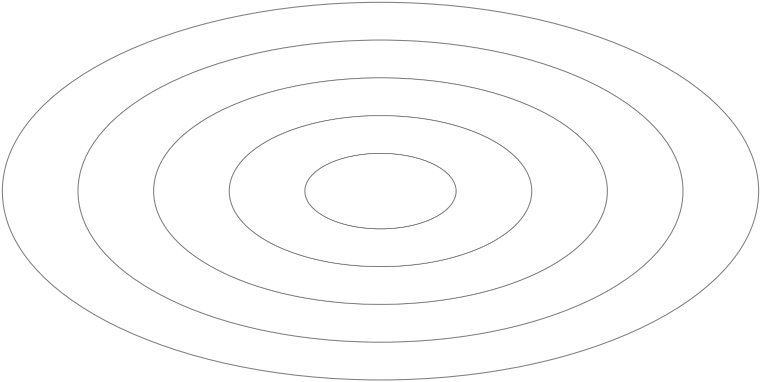
Why it still takes so long to implement?

Actually, there are so many options needed to select carefully. After testing for such a long time, we find that

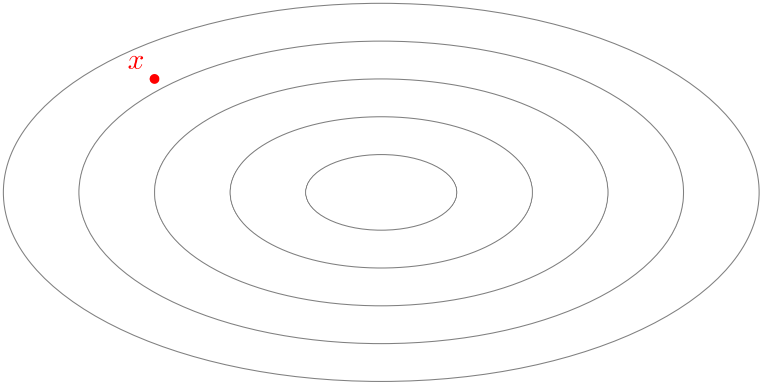
- Gauss-Seidel
- Coordinate directions

BDS source: github.com/blockwise-direct-search

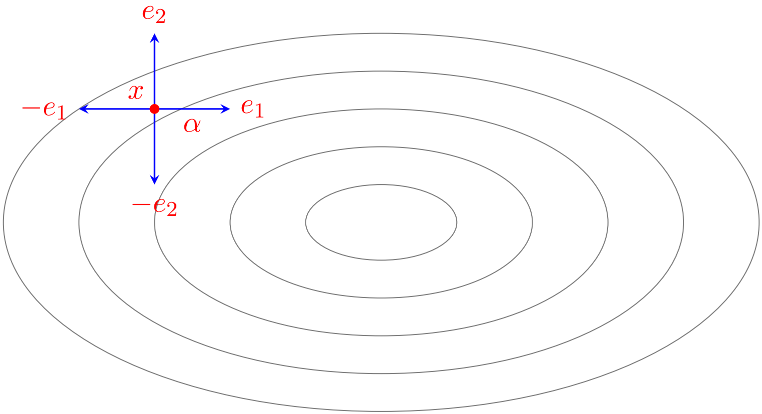
A simple example of the blockwise direct-search method



A simple example of the blockwise direct-search method

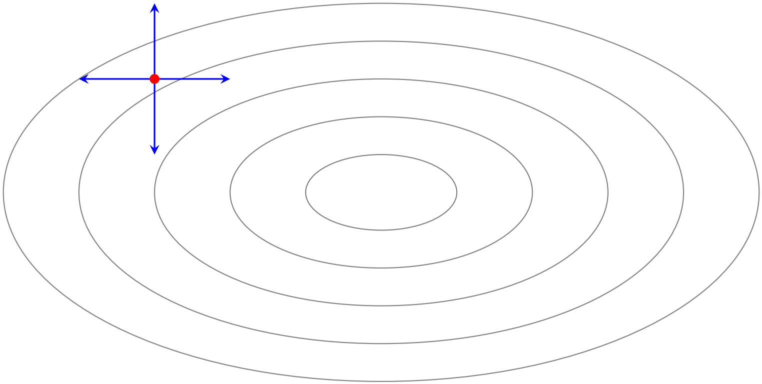


A simple example of the blockwise direct-search method

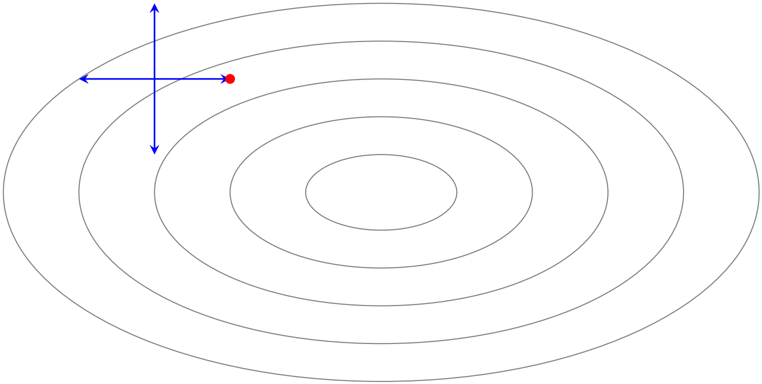


$\mathcal{D}_1 = \{e_1, -e_1\}$ and $\mathcal{D}_2 = \{e_2, -e_2\}$

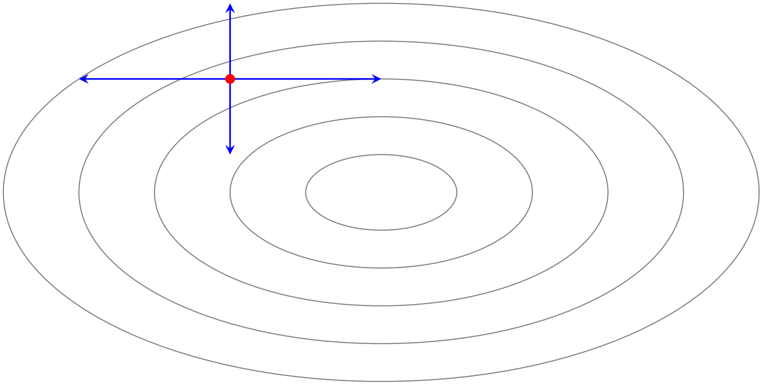
A simple example of the blockwise direct-search method



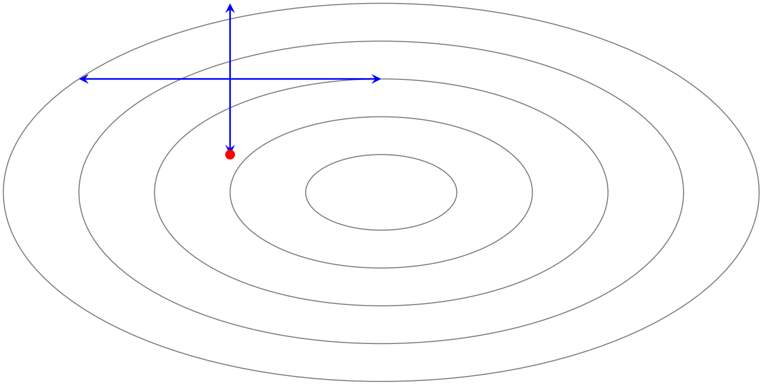
A simple example of the blockwise direct-search method



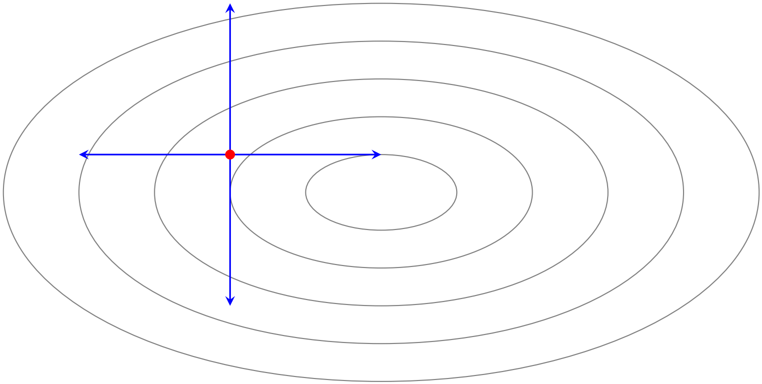
A simple example of the blockwise direct-search method



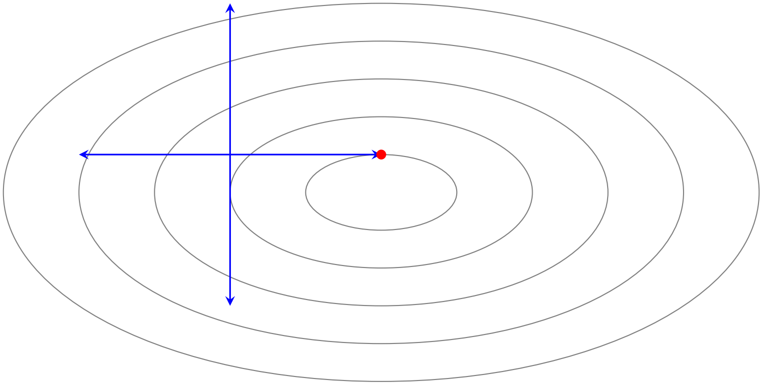
A simple example of the blockwise direct-search method



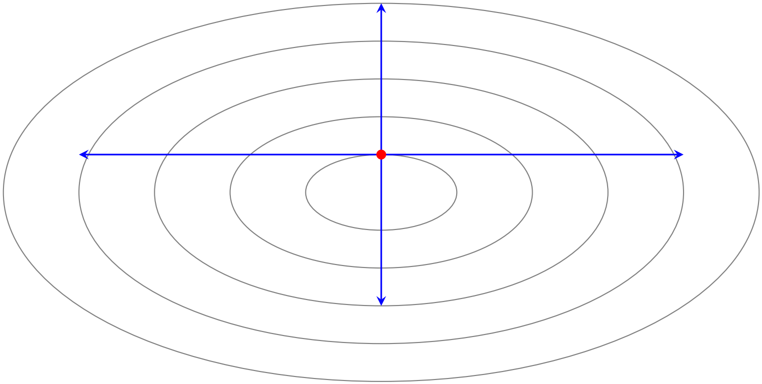
A simple example of the blockwise direct-search method



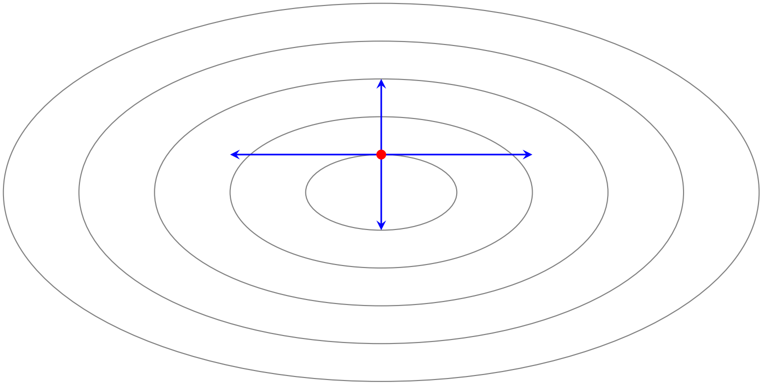
A simple example of the blockwise direct-search method



A simple example of the blockwise direct-search method



A simple example of the blockwise direct-search method



Outline

Comparison between BDS and existing DFO solvers

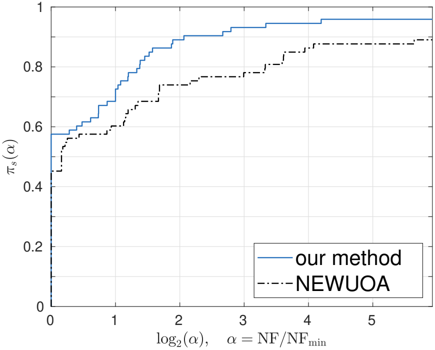
Observed value:

$$\widetilde{f}(x) = \begin{cases} f(x), & \text{there is no noise,} \\ f(x)[1+\epsilon(x)], & \text{there is noise,} \end{cases}$$

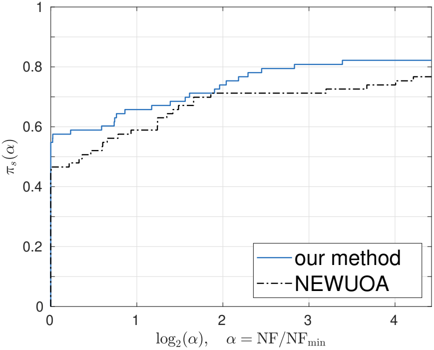
where $\epsilon(x) \sim \mathcal{N}(0, \sigma^2)$.
In our experiments:

- problem set: unconstrained problems from CUTEst
- dimensions: $6 \le n \le 100$
- noise level: $\sigma = 10^{-3}$
- budget: $1000n$ function evaluations
- number of random experiments: 10

Experiment (recapped)

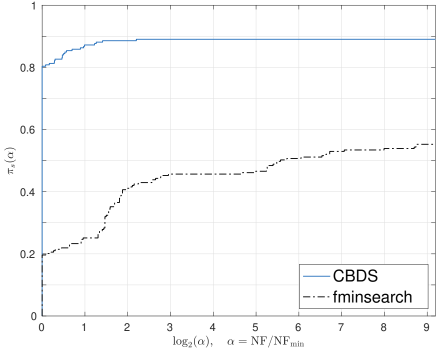


$\tau = 10^{-2}$

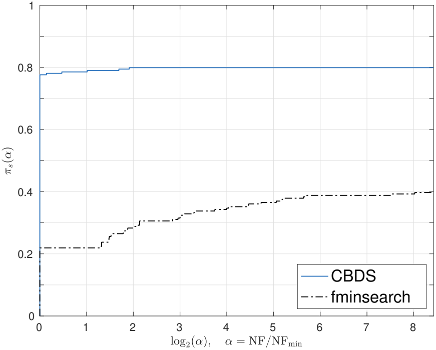


$\tau = 10^{-4}$

The comparison between CBDS and simplex

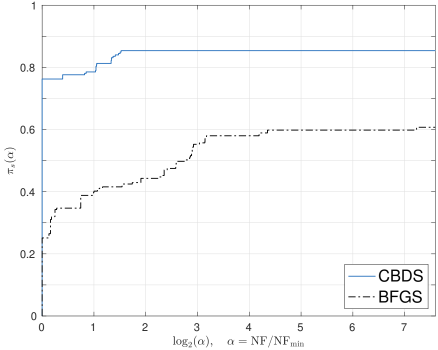


$\tau = 10^{-2}$

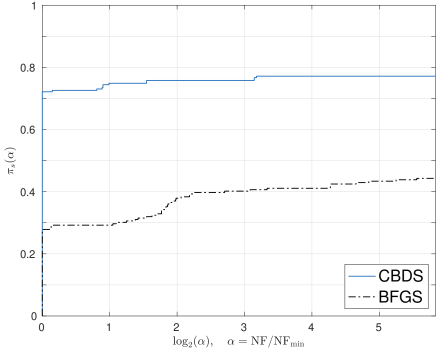


$\tau = 10^{-4}$

The comparison between CBDS and BFGS

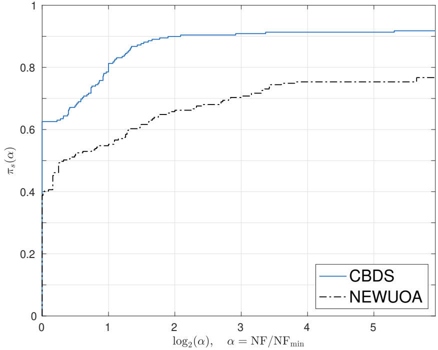


$\tau = 10^{-2}$

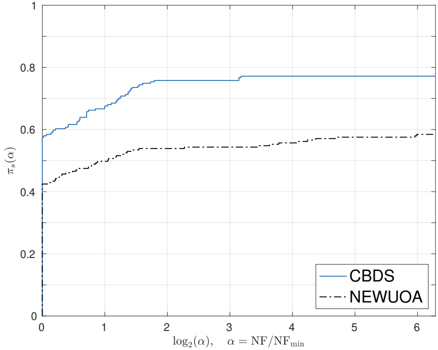


$\tau = 10^{-4}$

The comparison between CBDS and NEWUOA



$\tau = 10^{-2}$



$\tau = 10^{-4}$

Outline

Is CBDS convergent?

- In fact, 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.

Conclusions

What we have achieved:

- Our project is battle-tested
- Our method is efficient and robust to noise without any techniques

Future work:

- Convergence and worst-case complexity (of an adapted framework?)
- Finite difference or interpolation using existing iterates
- Rosenbrock's rotation

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.
- 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.
- A. Ciccazzo, V. Latorre, G. Liuzzi, S. Lucidi, and F. Rinaldi.
  Derivative-free robust optimization for circuit design.
  J. Optim. Theory Appl., 164:842–861, 2015.

References II

- H. Ghanbari and K. Scheinberg.
  Black-box optimization in machine learning with trust region based derivative free algorithm.
  arXiv:1703.06925, 2017.
- 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.

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. Kortelainen, T. Lesinski, J. More, W. Nazarewicz, J. Sarich, N. Schunck, M. V. Stoitsov, and S. M. Wild.
  Nuclear energy density optimization.
  Phys. Rev. C, 82(2):024313, 2010.
- 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.