Blockwise Direct-Search Methods

Haitian Li

Department of Applied Mathematics
The Hong Kong Polytechnic University

The 9th Graduate Forum of Mathematical Programming Branch of Operations Research of China

Superivsor: Dr. Zaikun Zhang and Prof. Xiaojun Chen

November 4, 2023

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

What is DFO?

Solve an optimization problem

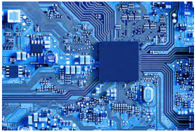
$$\min_{x \in \mathbb{R}^n} f(x)$$

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

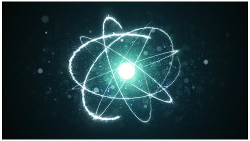
Why do we use DFO?

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

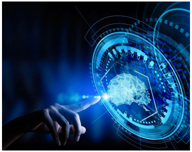
Examples of DFO problems



Circuit Design



Nuclear Energy



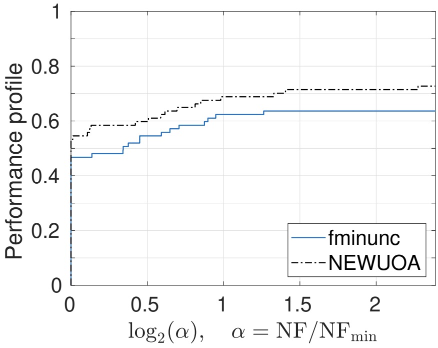
Machine Learning

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.

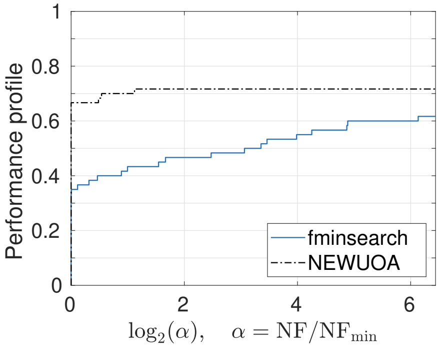
A powerful DFO solver: NEWUOA

- A derivative-free solver for unconstrained problems
- Developed by M.J.D. Powell
- Widely used by engineers and scientists

NEWUOA: performance is excellent



NEWUOA v.s. BFGS



NEWUOA v.s. simplex

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

NEWUOA: implementation and understanding is HARD

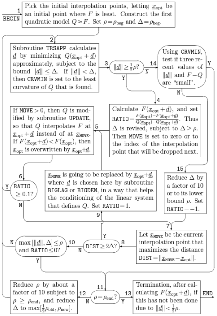


Figure 1: An outline of the method, where Y = Yes and N = No

Outline of NEWUOA's code

NEWUOA: implementation and understanding is HARD

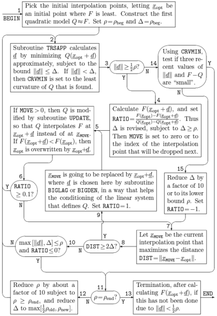


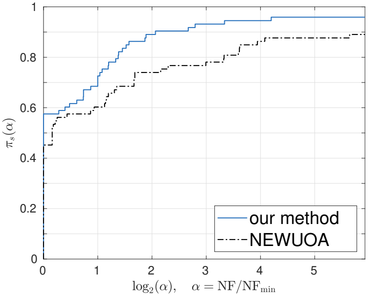
Figure 1: An outline of the method, where Y = Yes and N = No

Outline of NEWUOA's code

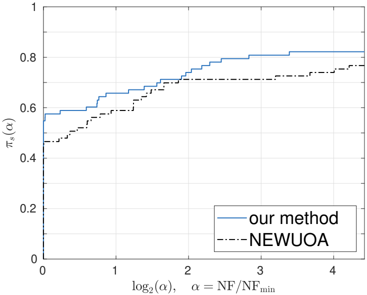
From Powell (2006)

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, $6 \le n \le 100$)

Six months v.s. Three frustrating years!

492 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 simple comparisons to decide the point to visit.

Algorithm 1: Direct Search (DS)

Input: $x_0 \in \mathbb{R}^n$, $0 < \theta < 1 \leq \gamma$, $\alpha_0 \in (0, \infty)$, 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 simple comparisons to decide the point to visit.

Algorithm 1: Direct Search (DS)

Input: $x_0 \in \mathbb{R}^n$, $0 < \theta < 1 \le \gamma$, $\alpha_0 \in (0, \infty)$, and searching set $\mathcal{D} \subset \mathbb{R}^n$.

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

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

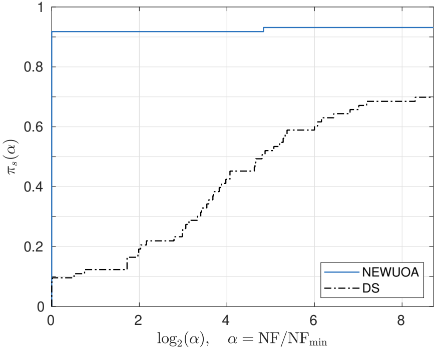
    Set $\alpha_{k+1} = \gamma \alpha_k$ and $x_{k+1} = x_k + \alpha_k d$.

  else

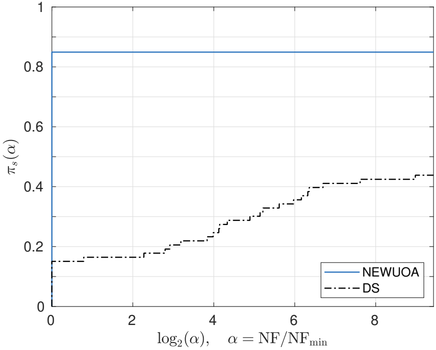
    Set $\alpha_{k+1} = \theta \alpha_k$ and $x_{k+1} = x_k$.

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 direction 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 direction 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: $x_0 \in \mathbb{R}^n$, $0 < \theta < 1 \le \gamma$, $\alpha_0^1, \ldots, \alpha_0^m \in (0, \infty)$, and searching set $\mathcal{D}^1, \ldots, \mathcal{D}^m \subset \mathbb{R}^n$.

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 is complicated?

There are so many choices needed to select carefully.

Why it is complicated?

There are so many choices needed to select carefully.

What is the best order to visit the blocks?

Why it is complicated?

There are so many choices needed to select carefully.

What is the best order to visit the blocks?

- Gauss-Seidel

Why it is complicated?

There are so many choices needed to select carefully.

What is the best order to visit the blocks?

- Gauss-Seidel

What is the best searching set for each block?

Why it is complicated?

There are so many choices needed to select carefully.

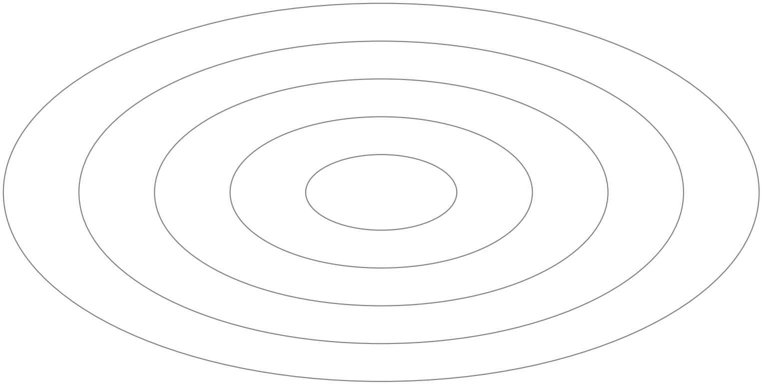
What is the best order to visit the blocks?

- Gauss-Seidel

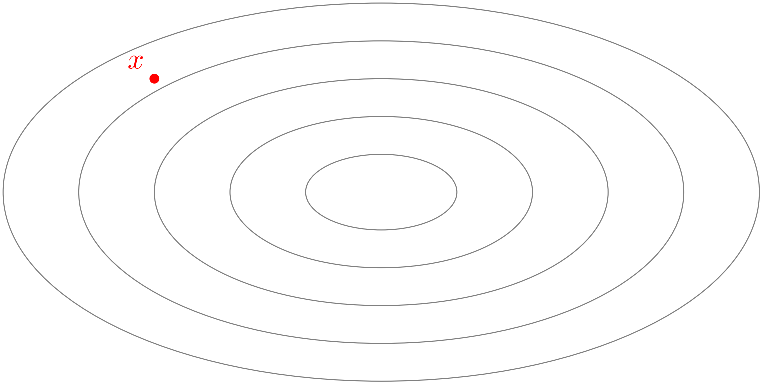
What is the best searching set for each block?

- Coordinate directions

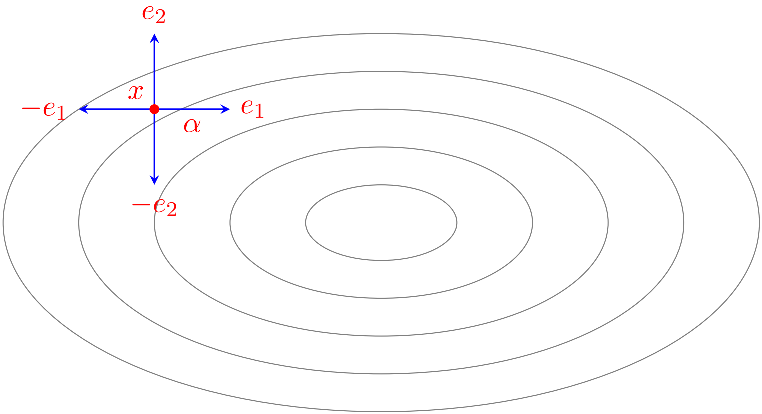
A simple example of the blockwise direct-search method



A simple example of the blockwise direct-search method

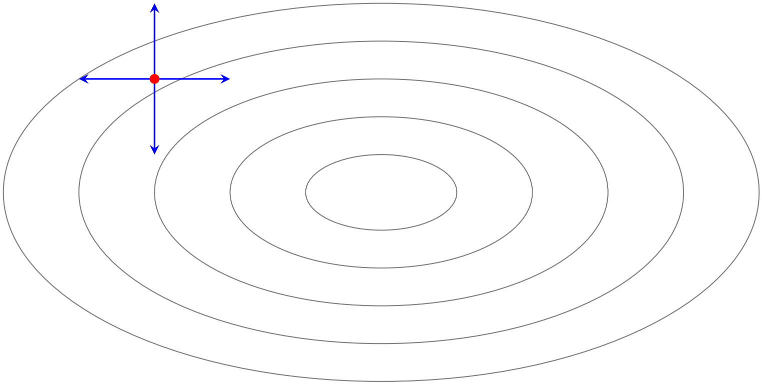


A simple example of the blockwise direct-search method

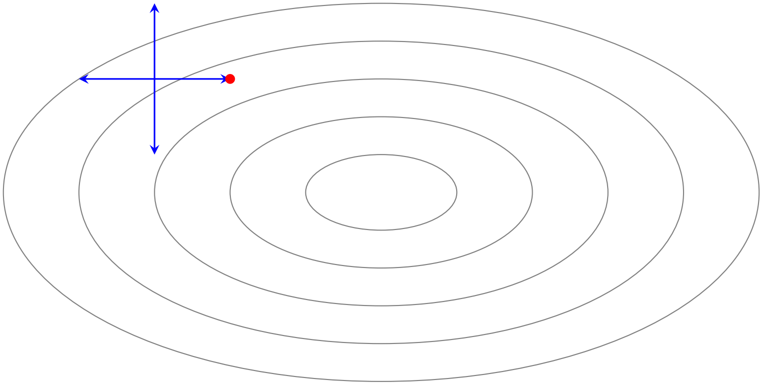


$\mathcal{D}_1 = \{e_1, -e_1\}$ and $\mathcal{D}_2 = \{e_2, -e_2\}$

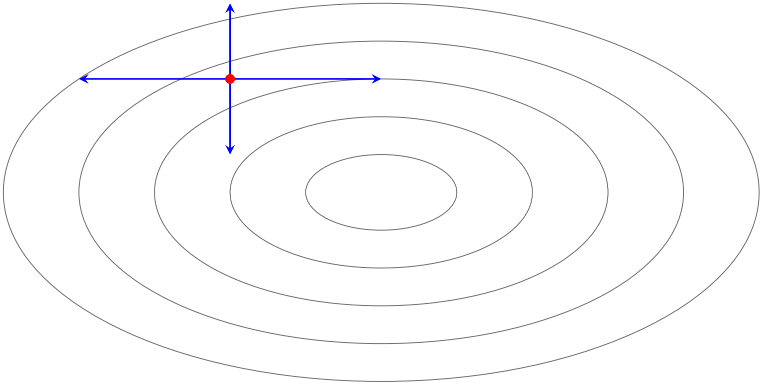
A simple example of the blockwise direct-search method



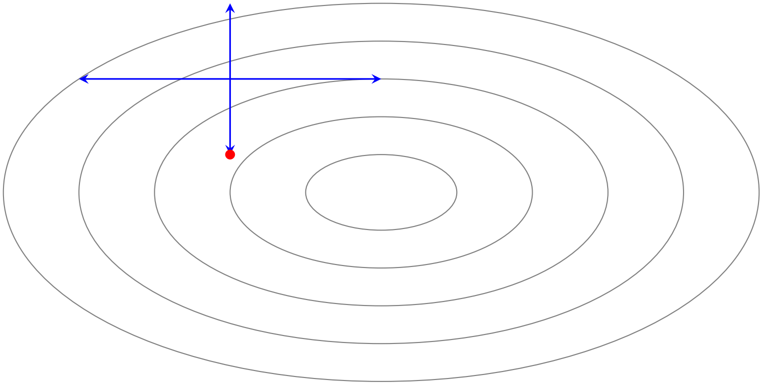
A simple example of the blockwise direct-search method



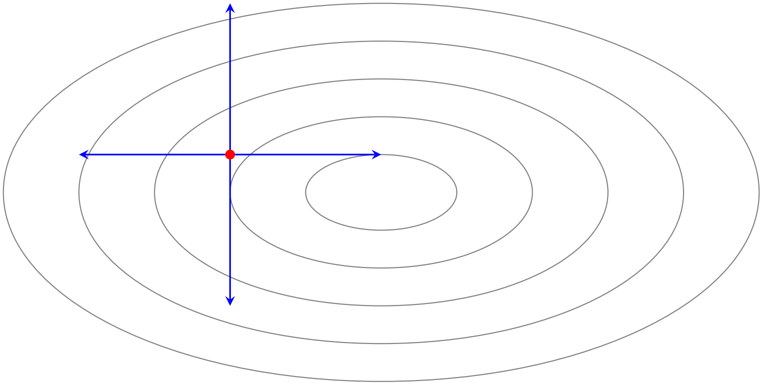
A simple example of the blockwise direct-search method



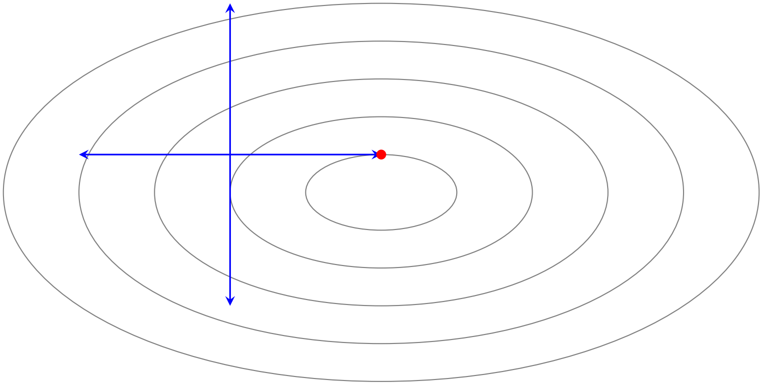
A simple example of the blockwise direct-search method



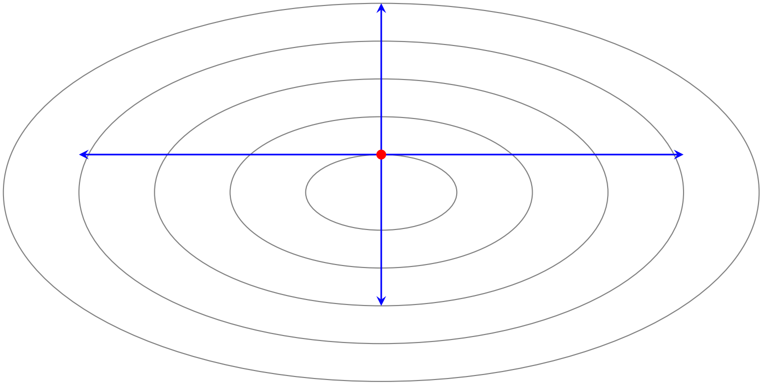
A simple example of the blockwise direct-search method



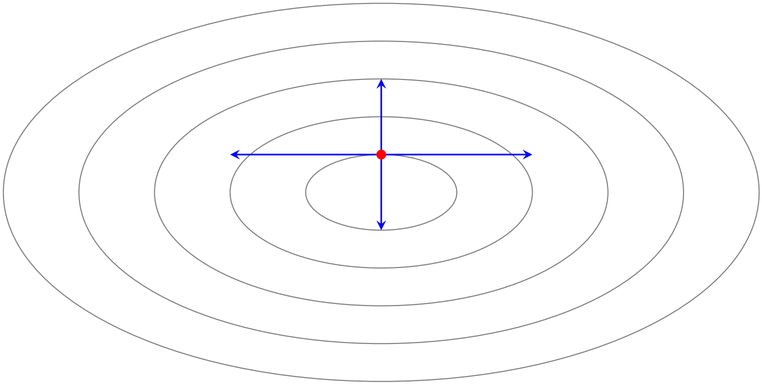
A simple example of the blockwise direct-search method



A simple example of the blockwise direct-search method



A simple example of the blockwise direct-search method



Outline

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

Comparison between BDS and existing DFO solvers

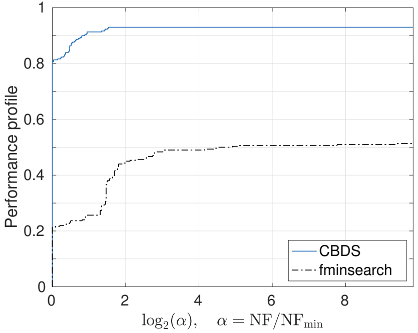
Observed value:

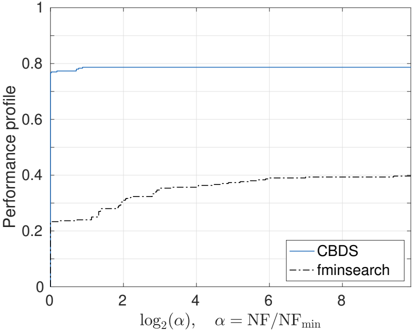
$$\widetilde{f}(x) = \begin{cases} f(x), & \text{there is no noise,} \\ f(x)[1+\epsilon(x)], & \text{there is noise,} \end{cases}$$

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

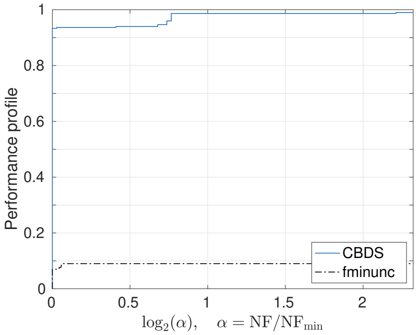
- problem set: unconstrained problems from CUTEst
- dimensions: $6 \leq n \leq 100$
- noise level: $\sigma = 10^{-3}$
- budget: $1000n$ function evaluations
- number of random experiments: 10

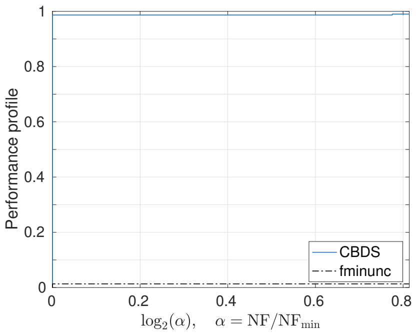
Comparison between BDS and existing DFO solvers



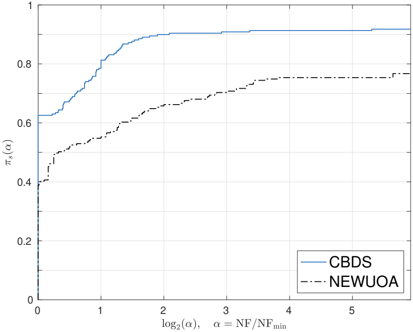


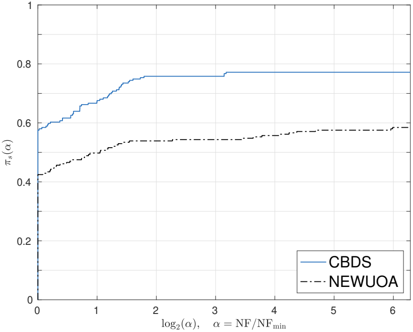
Comparison between BDS and existing DFO solvers



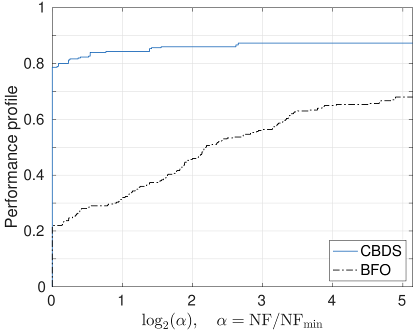


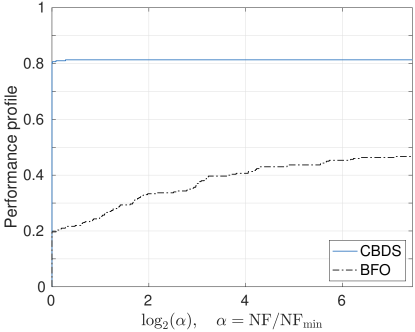
Comparison between BDS and existing DFO solvers





Comparison between BDS and existing DFO solvers





Battle-test!

README.md

Test of BDS.

The tests are **automated** by GitHub Actions.

- Check Spelling passing
- Unit test passing
- Stress test passing
- Parallel test passing
- Recursive test passing
- Test StepTolerance for bds passing
- Test forcing function for bds passing
- Test the technique of initial step size using initial point passing
- Plot performance profiles for BDS, small, testing level of reduction factor passing
- Plot performance profiles for BDS, big, testing level of reduction factor passing
- Plot performance profiles for bds and pbds, big passing
- Plot performance profiles for bds and pbds, small passing
- Plot performance profiles for bds and rbds, small passing
- Plot performance profiles for bds and dspd, big passing
- Plot performance profiles for bds and dspd, small passing
- Plot performance profiles for bds and bfgs, big passing
- Plot performance profiles for bds and bfgs, small passing
- Plot performance profiles for bds and prima, small passing
- Plot performance profiles for bds and simplex, big passing
- Plot performance profiles for bds and simplex, small passing
- Plot performance profiles for bds and nlopt, big passing
- Plot performance profiles for bds and nlopt, small passing
- Plot performance profiles for bds and bfo, small passing

Battle-test, is necessary!

Outline

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

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?

Conclusions

What we have achieved:

- Our project is open-source and easy to use
- Our method is efficient and adaptive to noise automatically

Future work:

- Convergence and worst-case complexity (of an adapted framework?)
- Finite difference or interpolation using existing iterates
- Apply our algorithm on constrained problems (like bound constraints)
- Apply our algorithm on other programming languages (like Python)

BDS homepage: `github.com/blockwise-direct-search`

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.