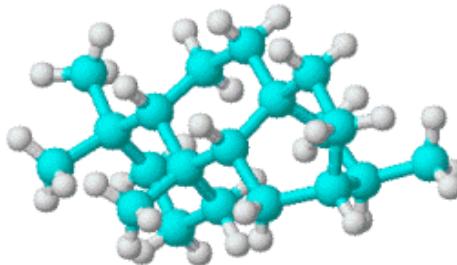


# **COCO: A Platform for Comparing Continuous Optimizers Effortlessly**

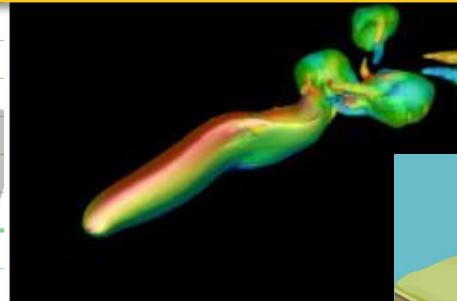
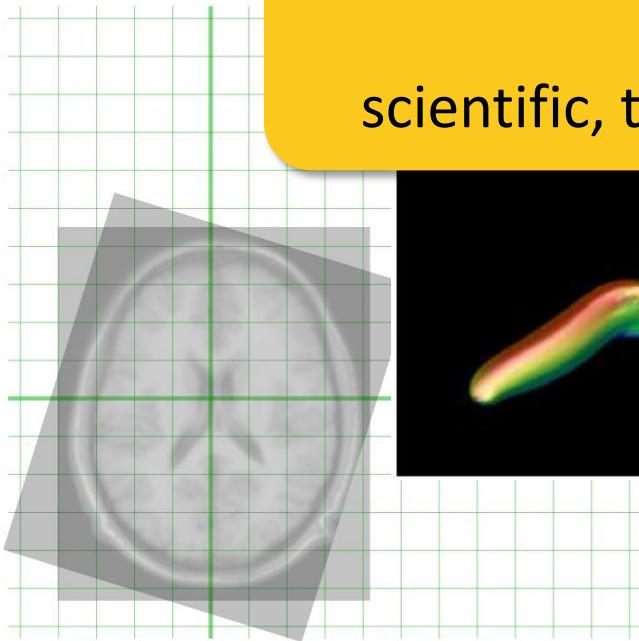
**Dimo Brockhoff**  
[dimo.brockhoff@inria.fr](mailto:dimo.brockhoff@inria.fr)



**optimization problems are ubiquitous**

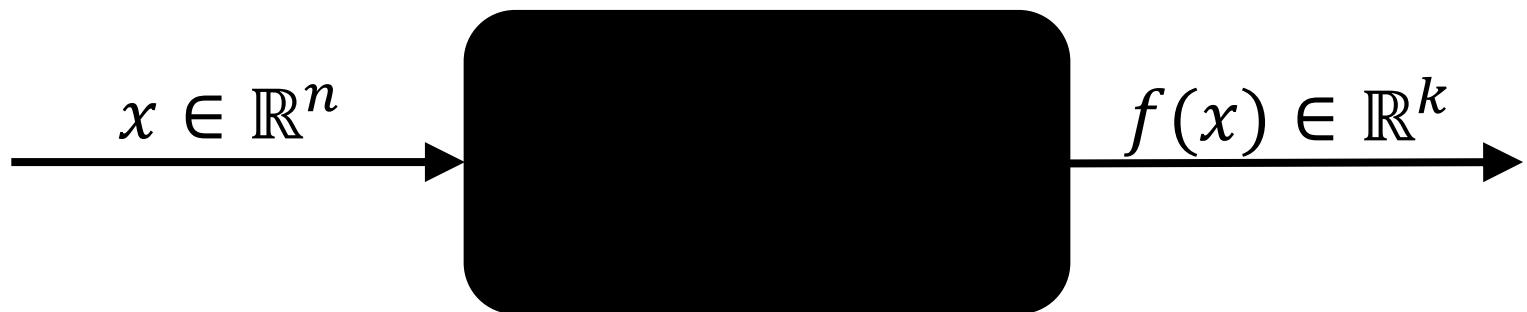


challenging optimization problems  
appear in many  
scientific, technological and industrial domains



# Numerical Blackbox Optimization

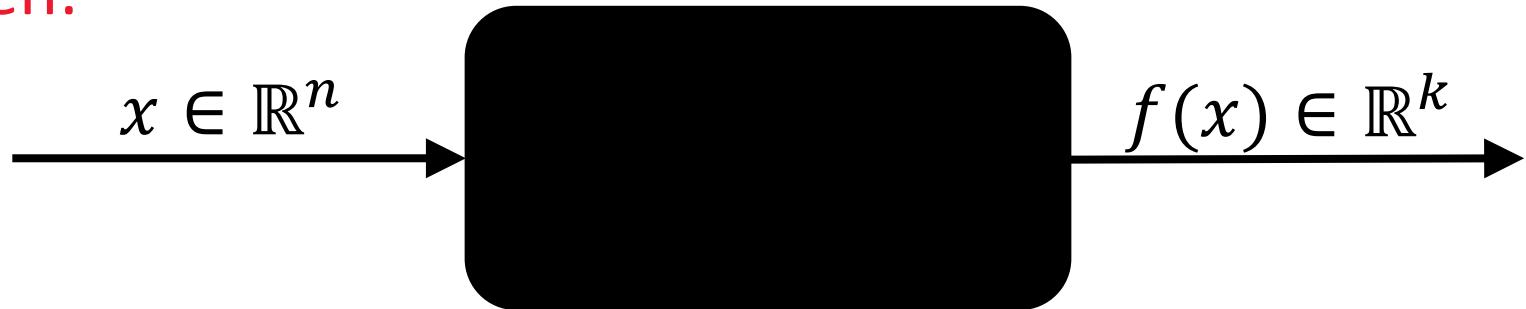
Optimize  $f: \Omega \subset \mathbb{R}^n \mapsto \mathbb{R}^k$



*derivatives not available or not useful*

# Practical Blackbox Optimization

Given:



Not clear:

which of the many algorithms should I use on my problem?

# Numerical Blackbox Optimizers

## Deterministic algorithms

Quasi-Newton with estimation of gradient (BFGS) [Broyden et al. 1970]

Simplex downhill [Nelder & Mead 1965]

Pattern search [Hooke and Jeeves 1961]

Trust-region methods (NEWUOA, BOBYQA) [Powell 2006, 2009]

## Stochastic (randomized) search methods

Evolutionary Algorithms (continuous domain)

- Differential Evolution [Storn & Price 1997]
- Particle Swarm Optimization [Kennedy & Eberhart 1995]
- **Evolution Strategies, CMA-ES** [Rechenberg 1965, Hansen & Ostermeier 2001]
- Estimation of Distribution Algorithms (EDAs) [Larrañaga, Lozano, 2002]
- Cross Entropy Method (same as EDA) [Rubinstein, Kroese, 2004]
- **Genetic Algorithms** [Holland 1975, Goldberg 1989]

Simulated annealing [Kirkpatrick et al. 1983]

Simultaneous perturbation stochastic approx. (SPSA) [Spall 2000]

# Numerical Blackbox Optimizers

## Deterministic algorithms

Quasi-Newton with estimation of gradient (BFGS) [Broyden et al. 1970]

Simplex downhill [Nelder & Mead 1965]

Pattern search [Hooke and Jeeves 1961]

Trust-region methods (NEWUOA, BOBYQA) [Powell 2006, 2009]

## Stochastic (randomized) search methods

Evolutionary Algorithms (continuous domain)

- Differential Evolution [Storn & Price 1997]
- Particle Swarm Optimization [Kennedy & Eberhart 1995]
- **Evolution Strategies, CMA-ES** [Rechenberg 1965, Hansen & Ostermeier 2001]
- Estimation of Distribution Algorithms (EDAs) [Larrañaga, Lozano, 2002]
- Cross Entropy Method (same as EDA) [Rubinstein, Kroese, 2004]
- **Genetic Algorithms** [Holland 1975, Goldberg 1989]

Simulated annealing [Kirkpatrick et al. 1983]

Simultaneous perturbation stochastic approx. (SPSA) [Spall 2000]

- choice typically not immediately clear
- although practitioners have knowledge about which difficulties their problem has (e.g. multi-modality, non-separability, ...)

# Need: Benchmarking

- understanding of algorithms
- algorithm selection
- putting algorithms to a standardized test
  - simplify judgement
  - simplify comparison
  - regression test under algorithm changes

Kind of everybody has to do it (and it is tedious):

- choosing (and implementing) problems, performance measures, visualization, stat. tests, ...
- running a set of algorithms

**that's where COCO comes into play**

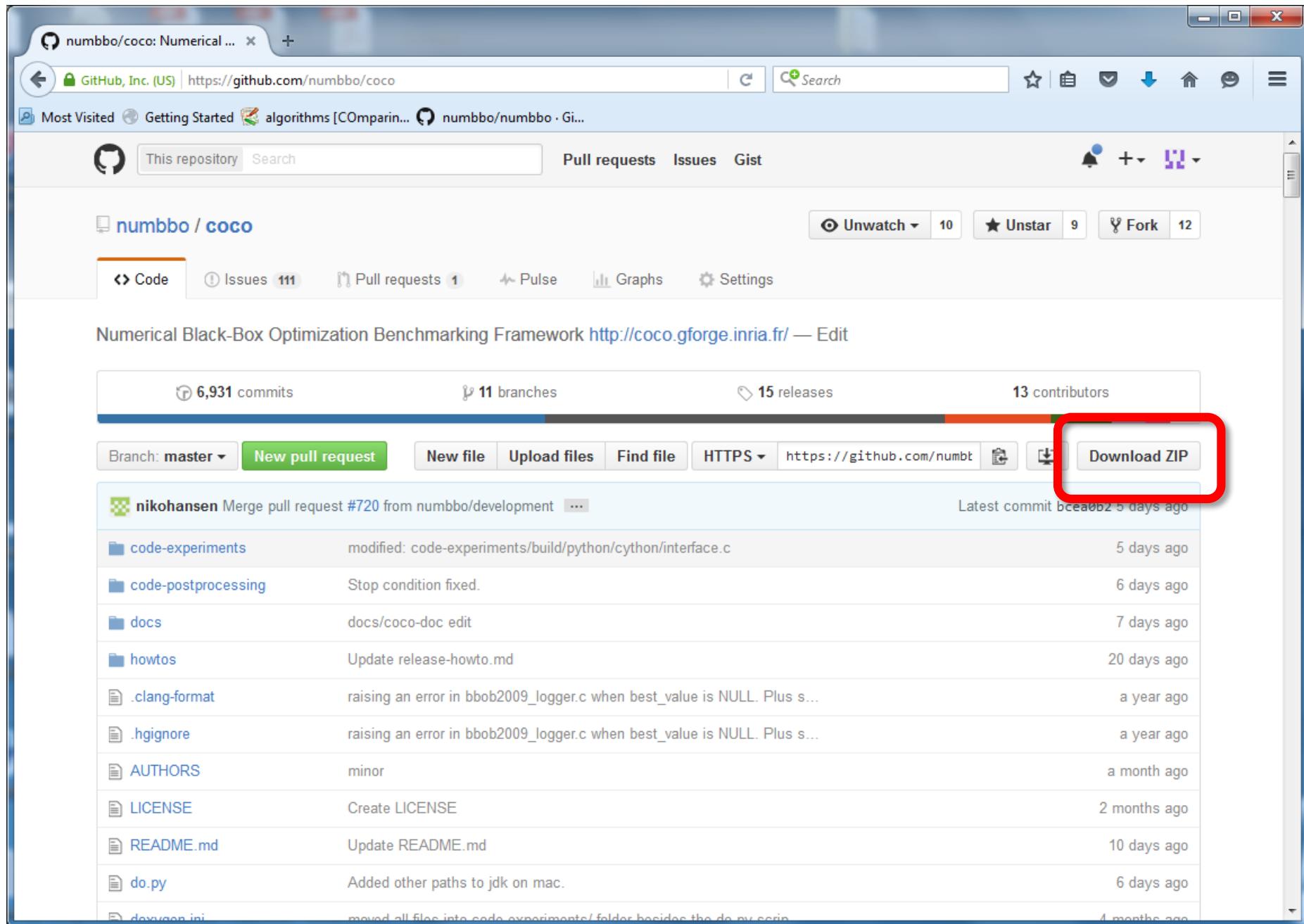
**automatized** benchmarking

# How to benchmark algorithms with COCO?

# https://github.com/numbbo/coco

numbbo/coco: Numerical ... + GitHub, Inc. (US) https://github.com/numbbo/coco Search Most Visited Getting Started algorithms [C] algorit... numbbo/numbbo · Gi... This repository Search Pull requests Issues Gist Unwatch 10 Unstar 9 Fork 12 Code Issues 111 Pull requests 1 Pulse Graphs Settings Numerical Black-Box Optimization Benchmarking Framework <http://coco.gforge.inria.fr/> — Edit 6,931 commits 11 branches 15 releases 13 contributors Branch: master New pull request New file Upload files Find file HTTPS https://github.com/numbbo/numbbo Download ZIP nikohansen Merge pull request #720 from numbbo/development ... Latest commit bcea0b2 5 days ago code-experiments modified: code-experiments/build/python/cython/interface.c 5 days ago code-postprocessing Stop condition fixed. 6 days ago docs docs/coco-doc edit 7 days ago howtos Update release-howto.md 20 days ago .clang-format raising an error in bbob2009\_logger.c when best\_value is NULL. Plus s... a year ago .hgignore raising an error in bbob2009\_logger.c when best\_value is NULL. Plus s... a year ago AUTHORS minor a month ago LICENSE Create LICENSE 2 months ago README.md Update README.md 10 days ago do.py Added other paths to jdk on mac. 6 days ago doxygen.ini moved all files into code-experiments/ folder besides the do.py scrip 4 months ago

<https://github.com/numbbo/coco>



Numerical Black-Box Optimization Benchmarking Framework <http://coco.gforge.inria.fr/> — Edit

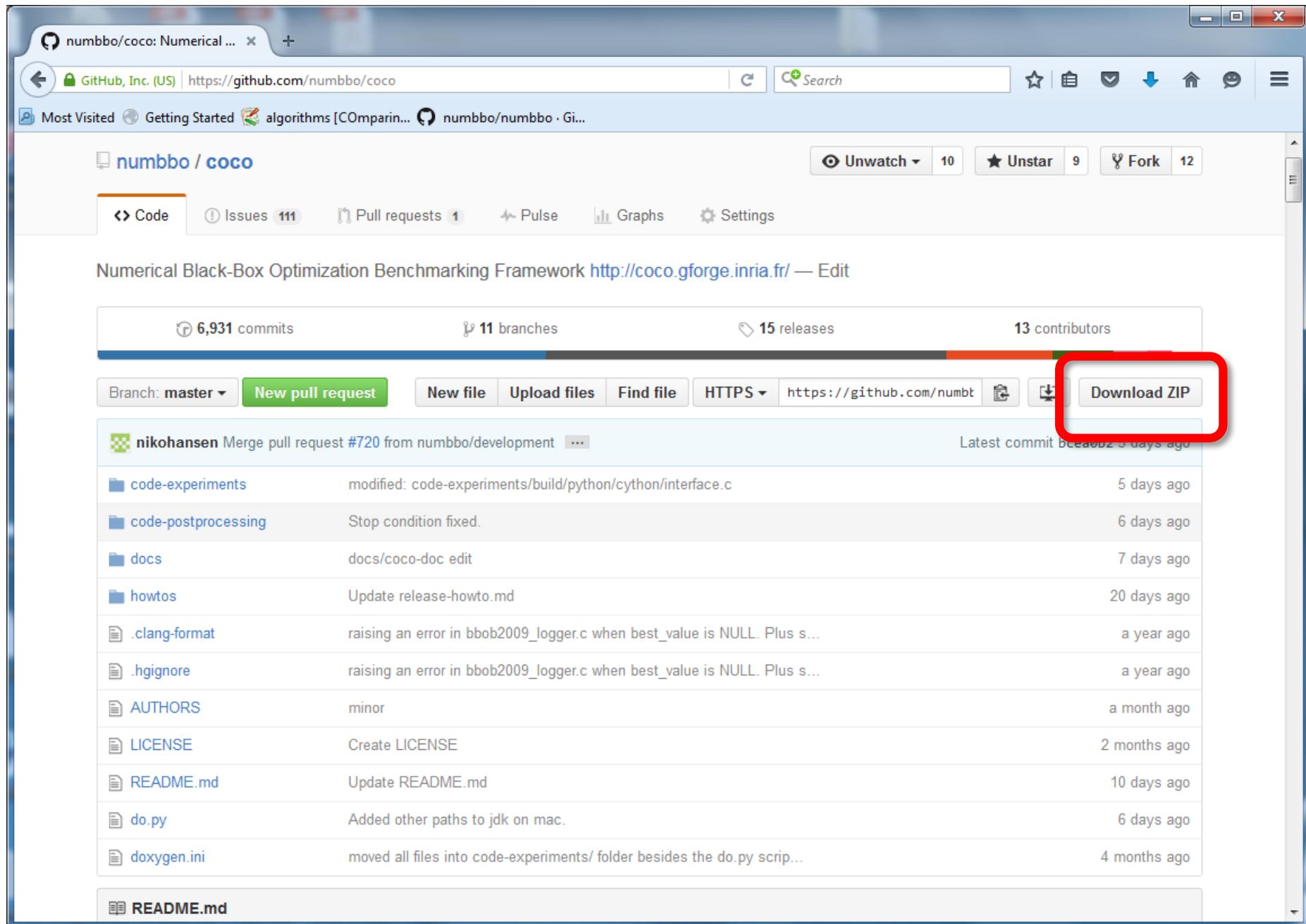
6,931 commits 11 branches 15 releases 13 contributors

Branch: **master** New pull request New file Upload files Find file HTTPS <https://github.com/numbbo/coco> Download ZIP

**nikohansen** Merge pull request #720 from numbbo/development ... Latest commit `bcea0b2` 5 days ago

File	Commit Message	Time
code-experiments	modified: code-experiments/build/python/cython/interface.c	5 days ago
code-postprocessing	Stop condition fixed.	6 days ago
docs	docs/coco-doc edit	7 days ago
howtos	Update release-howto.md	20 days ago
.clang-format	raising an error in bbob2009_logger.c when best_value is NULL. Plus s...	a year ago
.hgignore	raising an error in bbob2009_logger.c when best_value is NULL. Plus s...	a year ago
AUTHORS	minor	a month ago
LICENSE	Create LICENSE	2 months ago
README.md	Update README.md	10 days ago
do.py	Added other paths to jdk on mac.	6 days ago
doxygen.ini	removed all files into code-experiments/ folder besides the do.py scrip...	4 months ago

<https://github.com/numbbo/coco>



The screenshot shows a GitHub repository page for 'numbbo/coco'. The page includes a header with a search bar and navigation links. Below the header, there are buttons for 'Unwatch' (10), 'Unstar' (9), and 'Fork' (12). The main content area displays the repository's name, 'numbbo / coco', and a navigation bar with 'Code' (selected), 'Issues' (111), 'Pull requests' (1), 'Pulse', 'Graphs', and 'Settings'. A summary bar shows 6,931 commits, 11 branches, 15 releases, and 13 contributors. The commit history table lists various commits, including a merge pull request from 'nikohansen' and several updates to files like 'code-experiments', 'docs', and 'LICENSE'. A red box highlights the 'Download ZIP' button in the top right of the commit history area.

Numerical Black-Box Optimization Benchmarking Framework <http://coco.gforge.inria.fr/> — Edit

6,931 commits 11 branches 15 releases 13 contributors

Branch: master [New pull request](#) [New file](#) [Upload files](#) [Find file](#) [HTTPS](#) <https://github.com/numbbo/coco> [Download ZIP](#)

nikohansen	Merge pull request #720 from numbbo/development	...	Latest commit bcead02 3 days ago
code-experiments	modified: code-experiments/build/python/cython/interface.c		5 days ago
code-postprocessing	Stop condition fixed.		6 days ago
docs	docs/coco-doc edit		7 days ago
howtos	Update release-howto.md		20 days ago
.clang-format	raising an error in bbob2009_logger.c when best_value is NULL. Plus s...		a year ago
.hgignore	raising an error in bbob2009_logger.c when best_value is NULL. Plus s...		a year ago
AUTHORS	minor		a month ago
LICENSE	Create LICENSE		2 months ago
README.md	Update README.md		10 days ago
do.py	Added other paths to jdk on mac.		6 days ago
doxygen.ini	moved all files into code-experiments/ folder besides the do.py scrip...		4 months ago
README.md			

<https://github.com/numbbo/coco>

Numerical Black-Box Optimization Benchmarking Framework <http://coco.gforge.inria.fr/> — Edit

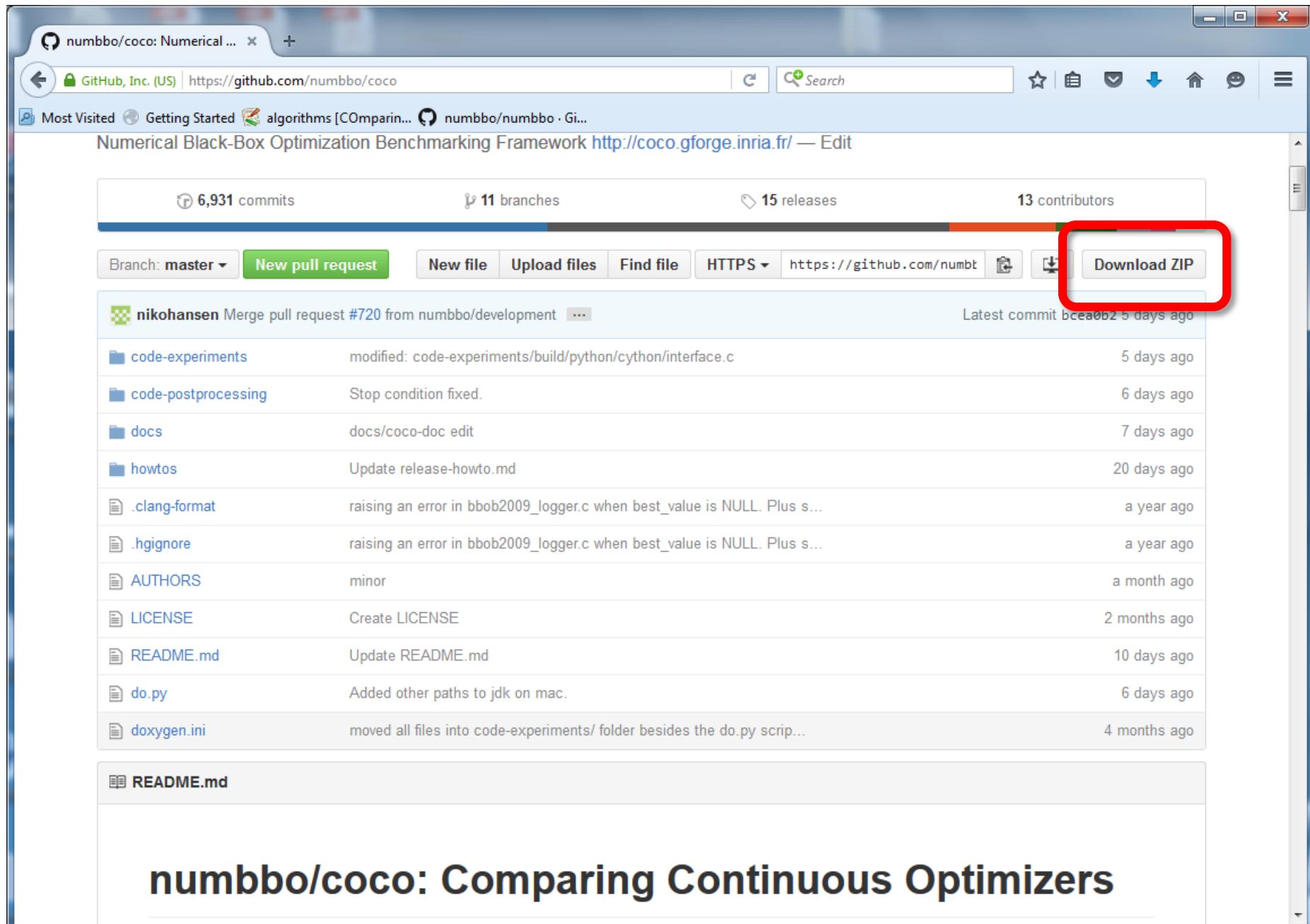
6,931 commits 11 branches 15 releases 13 contributors

Branch: master [New pull request](#) [New file](#) [Upload files](#) [Find file](#) [HTTPS](#) <https://github.com/numbbo/coco> [Download ZIP](#)

**nikohansen** Merge pull request #720 from numbbo/development ... Latest commit [bbob2009 3 days ago](#)

File	Commit Message	Time
code-experiments	modified: code-experiments/build/python/cython/interface.c	5 days ago
code-postprocessing	Stop condition fixed.	6 days ago
docs	docs/coco-doc edit	7 days ago
howtos	Update release-howto.md	20 days ago
.clang-format	raising an error in bbob2009_logger.c when best_value is NULL. Plus s...	a year ago
.hgignore	raising an error in bbob2009_logger.c when best_value is NULL. Plus s...	a year ago
AUTHORS	minor	a month ago
LICENSE	Create LICENSE	2 months ago
README.md	Update README.md	10 days ago
do.py	Added other paths to jdk on mac.	6 days ago
doxygen.ini	moved all files into code-experiments/ folder besides the do.py scrip...	4 months ago
README.md		

<https://github.com/numbbo/coco>



Numerical Black-Box Optimization Benchmarking Framework <http://coco.gforge.inria.fr/> — Edit

6,931 commits 11 branches 15 releases 13 contributors

Branch: master [New pull request](#) [New file](#) [Upload files](#) [Find file](#) [HTTPS](#) <https://github.com/numbbo/coco> [Download ZIP](#)

nikohansen	Merge pull request #720 from numbbo/development	...	Latest commit bcea0b2 5 days ago
code-experiments	modified: code-experiments/build/python/cython/interface.c		5 days ago
code-postprocessing	Stop condition fixed.		6 days ago
docs	docs/coco-doc edit		7 days ago
howtos	Update release-howto.md		20 days ago
.clang-format	raising an error in bbob2009_logger.c when best_value is NULL. Plus s...		a year ago
.hgignore	raising an error in bbob2009_logger.c when best_value is NULL. Plus s...		a year ago
AUTHORS	minor		a month ago
LICENSE	Create LICENSE		2 months ago
README.md	Update README.md		10 days ago
do.py	Added other paths to jdk on mac.		6 days ago
doxygen.ini	moved all files into code-experiments/ folder besides the do.py scrip...		4 months ago
README.md			

**numbbo/coco: Comparing Continuous Optimizers**

<https://github.com/numbbo/coco>

The screenshot shows a web browser window with the GitHub repository for `numbbo/coco`. The repository page includes a list of recent commits, a `README.md` file, and a brief description of the project.

**Recent Commits:**

Author	Commit Message	Time Ago	
nikohansen	Merge pull request #720 from numbbo/development	Latest commit bcea0b2 5 days ago	
	modified: code-experiments/build/python/cython/interface.c	5 days ago	
	code-experiments	Stop condition fixed.	6 days ago
	code-postprocessing	docs/coco-doc edit	7 days ago
	docs	Update release-howto.md	20 days ago
	howtos	raising an error in bbob2009_logger.c when best_value is NULL. Plus s...	a year ago
	.clang-format	raising an error in bbob2009_logger.c when best_value is NULL. Plus s...	a year ago
	.hgignore	minor	a month ago
	AUTHORS	Create LICENSE	2 months ago
	LICENSE	Update README.md	10 days ago
	README.md	Added other paths to jdk on mac.	6 days ago
	do.py	moved all files into code-experiments/ folder besides the do.py scrip...	4 months ago
	doxygen.ini		

**README.md:**

## numbbo/coco: Comparing Continuous Optimizers

This code reimplements the original Comparing Continuous Optimizer platform, now rewritten fully in `ANSI C` with other languages calling the `C` code. As the name suggests, the code provides a platform to benchmark and compare continuous optimizers, AKA non-linear solvers for numerical optimization. Languages currently available are

<https://github.com/numbbo/coco>

The screenshot shows a web browser window with the GitHub repository for `numbbo/coco`. The repository page includes a list of recent commits, a `README.md` file, and a detailed description of the project.

**Commits:**

File	Commit Message	Time Ago
<code>howtos</code>	Update release-howto.md	20 days ago
<code>.clang-format</code>	raising an error in <code>bbob2009_logger.c</code> when <code>best_value</code> is <code>NULL</code> . Plus s...	a year ago
<code>.hgignore</code>	raising an error in <code>bbob2009_logger.c</code> when <code>best_value</code> is <code>NULL</code> . Plus s...	a year ago
<code>AUTHORS</code>	minor	a month ago
<code>LICENSE</code>	Create LICENSE	2 months ago
<code>README.md</code>	Update README.md	10 days ago
<code>do.py</code>	Added other paths to jdk on mac.	6 days ago
<code>doxygen.ini</code>	moved all files into <code>code-experiments/</code> folder besides the <code>do.py</code> scrip...	4 months ago

**README.md:**

## numbbo/coco: Comparing Continuous Optimizers

This code reimplements the original Comparing Continuous Optimizer platform, now rewritten fully in `ANSI C` with other languages calling the `C` code. As the name suggests, the code provides a platform to benchmark and compare continuous optimizers, AKA non-linear solvers for numerical optimization. Languages currently available are

- `C/C++`
- `Java`
- `MATLAB/Octave`
- `Python`

Contributions to link further languages (including a better example in `C++`) are more than welcome.

For more information,

numbbo/coco: Numerical ... + GitHub, Inc. (US) https://github.com/numbbo/coco Search Most Visited Getting Started algorithms [C] numbbo/numbbo · Gi... doxygen.ini moved all files into code-experiments/ folder besides the do.py scri... 4 months ago README.md

# numbbo/coco: Comparing Continuous Optimizers

This code reimplements the original Comparing Continuous Optimizer platform, now rewritten fully in `ANSI C` with other languages calling the `C` code. As the name suggests, the code provides a platform to benchmark and compare continuous optimizers, NLP solvers and linear solvers for numerical optimization. Languages currently available are

- `C/C++`
- `Java`
- `MATLAB/Octave`
- `Python`

Contributions to link further languages (including a better example in `C++`) are more than welcome.

For more information,

- consult the [BBOB workshops series](#),
- consider to [register here](#) for news,
- see the [previous COCO home page here](#) and
- see the [links below](#) to learn more about the ideas behind CoCO.

## Requirements

1. For a machine running experiments

numbbo/coco: Numerical ... GitHub, Inc. (US) https://github.com/numbbo/coco Search Most Visited Getting Started algorithms [COMparin... numbbo/numbbo · Gi...

## Getting Started

1. Check out the [Requirements](#) above.
2. **Download** the COCO framework code from [github](#),
  - either by clicking [here](#) and unzip the `zip` file,
  - or (preferred) by typing `git clone https://github.com/numbbo/coco.git`. This way allows to remain up-to-date easily (but needs `git` to be installed). After cloning, `git pull` keeps the code up-to-date with the latest release.
3. In a system shell, `cd` into the `coco` or `coco-<version>` folder (framework root), where the file `do.py` can be found. Type, i.e. **execute**, one of the following commands once

```
python do.py run-c
python do.py run-java
python do.py run-matlab
python do.py run-octave
python do.py run-python
```

depending on which language shall be used to run the experiments. `run-*` will build the respective code and run the example experiment once. The build result and the example experiment code can be found under `code-experiments/build/<language>` (`<language>=matlab` for Octave). `python do.py` lists all available commands.
4. On the computer where experiment data shall be post-processed, run

```
python do.py install-postprocessing
```

numbbo/coco: Numerical ... + GitHub, Inc. (US) https://github.com/numbbo/coco Search Most Visited Getting Started algorithms [COMparin... numbbo/numbbo · Gi...

## Getting Started

1. Check out the [Requirements](#) above.
2. **Download** the COCO framework code from [github](#),
  - o either by clicking [here](#) and unzip the `zip` file,
  - o or (preferred) by typing `git clone https://github.com/numbbo/coco.git`. This way allows to remain up-to-date easily (but needs `git` to be installed). After cloning, `git pull` keeps the code up-to-date with the latest release.
3. In a system shell, `cd` into the `coco` or `coco-<version>` folder (framework root), where the file `do.py` can be found. Type, i.e. **execute**, one of the following commands once

```
python do.py run-c
python do.py run-java
python do.py run-matlab
python do.py run-octave
python do.py run-python
```

depending on which language shall be used to run the experiments. `run-*` will build the respective code and run the example experiment once. The build result and the example experiment code can be found under `code-experiments/build/<language>` (`<language>=matlab` for Octave). `python do.py` lists all available commands.

4. On the computer where experiment data shall be post-processed, run

```
python do.py install-postprocessing
```

numbbo/coco: Numerical ... + GitHub, Inc. (US) https://github.com/numbbo/coco Search Most Visited Getting Started algorithms [COMparin... numbbo/numbbo · Gi...

4. On the computer where experiment data shall be post-processed, run

```
python do.py install-postprocessing
```

to (user-locally) install the post-processing. From here on, `do.py` has done its job and is only needed again for updating the builds to a new release.

5. **Copy** the folder `code-experiments/build/YOUR-FAVORITE-LANGUAGE` and its content to another location. In Python it is sufficient to copy the file `example_experiment.py`. Run the example experiment (it already is compiled, in case). As the details vary, see the respective read-me's and/or example experiment files:

- o C [read me and example experiment](#)
- o Java [read me and example experiment](#)
- o Matlab/Octave [read me and example experiment](#)
- o Python [read me and example experiment](#)

If the example experiment runs, **connect** your favorite algorithm to Coco: replace the call to the random search optimizer in the example experiment file by a call to your algorithm (see above). **Update** the output `result_folder`, the `algorithm_name` and `algorithm_info` of the observer options in the example experiment file.

Another entry point for your own experiments can be the `code-experiments/examples` folder.

6. Now you can **run** your favorite algorithm on the `bbob-biobj` (for multi-objective algorithms) or on the `bbob` suite (for single-objective algorithms). Output is automatically generated in the specified data `result_folder`.

7. **Postprocess** the data from the results folder by typing

```
python -m bbo_bbob_pproc [-o OUTPUT_FOLDERNAME] YOURDATAFOLDER [MORE_DATAFOLDERS]
```

numbbo/coco: Numerical ... + GitHub, Inc. (US) https://github.com/numbbo/coco Search Most Visited Getting Started algorithms [COmparing... numbbo/numbbo · Gi...

4. On the computer where experiment data shall be post-processed, run

```
python do.py install-postprocessing
```

to (user-locally) install the post-processing. From here on, `do.py` has done its job and is only needed again for updating the builds to a new release.

5. **Copy** the folder `code-experiments/build/YOUR-FAVORITE-LANGUAGE` and its content to another location. In Python it is sufficient to copy the file `example_experiment.py`. Run the example experiment (it already is compiled, in case). As the details vary, see the respective read-me's and/or example experiment files:

- o C [read me and example experiment](#)
- o Java [read me and example experiment](#)
- o Matlab/Octave [read me and example experiment](#)
- o Python [read me and example experiment](#)

If the example experiment runs, **connect** your favorite algorithm to Coco: replace the call to the random search optimizer in the example experiment file by a call to your algorithm (see above). **Update** the output `result_folder`, the `algorithm_name` and `algorithm_info` of the observer options in the example experiment file.

Another entry point for your own experiments can be the `code-experiments/examples` folder.

6. Now you can **run** your favorite algorithm on the `bbob-biobj` (for multi-objective algorithms) or on the `bbob` suite (for single-objective algorithms). Output is automatically generated in the specified data `result_folder`.

7. **Postprocess** the data from the results folder by typing

```
python -m bbo_bbob_pproc [-o OUTPUT_FOLDERNAME] YOURDATAFOLDER [MORE_DATAFOLDERS]
```

# example\_experiment.c

```
/* Iterate over all problems in the suite */
while ((PROBLEM = coco_suite_get_next_problem(suite, observer)) != NULL)
{
    size_t dimension = coco_problem_get_dimension(PROBLEM);

    /* Run the algorithm at least once */
    for (run = 1; run <= 1 + INDEPENDENT_RESTARTS; run++) {

        size_t evaluations_done = coco_problem_get_evaluations(PROBLEM);
        long evaluations_remaining =
            (long) (dimension * BUDGET_MULTIPLIER) - (long)evaluations_done;

        if (... || (evaluations_remaining <= 0))
            break;

        my_random_search(evaluate_function, dimension,
                         coco_problem_get_number_of_objectives(PROBLEM),
                         coco_problem_get_smallest_values_of_interest(PROBLEM),
                         coco_problem_get_largest_values_of_interest(PROBLEM),
                         (size_t) evaluations_remaining,
                         random_generator);
    }
}
```

# example\_experiment.c

```
/* Iterate over all problems in the suite */
while ((PROBLEM = coco_suite_get_next_problem(suite, observer)) != NULL)
{
    size_t dimension = coco_problem_get_dimension(PROBLEM);

    /* Run the algorithm at least once */
    for (run = 1; run <= 1 + INDEPENDENT_RESTARTS; run++) {

        size_t evaluations_done = coco_problem_get_evaluations(PROBLEM);
        long evaluations_remaining =
            (long) (dimension * BUDGET_MULTIPLIER) - (long)evaluations_done;

        if (... || (evaluations_remaining <= 0))
            break;

        my_random_search(evaluate_function, dimension,
                         coco_problem_get_number_of_objectives(PROBLEM),
                         coco_problem_get_smallest_values_of_interest(PROBLEM),
                         coco_problem_get_largest_values_of_interest(PROBLEM),
                         (size_t) evaluations_remaining,
                         random_generator);
    }
}
```

numbbo/coco: Numerical ... GitHub, Inc. (US) https://github.com/numbbo/coco Search

Most Visited Getting Started algorithms [C] numbbo/numbbo · Gi...

Another entry point for your own experiments can be the `code-experiments/examples` folder.

6. Now you can **run** your favorite algorithm on the `bbob-biobj` (for multi-objective algorithms) or on the `bbob` suite (for single-objective algorithms). Output is automatically generated in the specified data `result_folder`.

7. **Postprocess** the data from the results folder by typing

```
python -m bbo_pproc [-o OUTPUT_FOLDERNAME] YOURDATAFOLDER [MORE_DATAFOLDERS]
```

The name `bbob_pproc` will become `cocopp` in future. Any subfolder in the folder arguments will be searched for logged data. That is, experiments from different batches can be in different folders collected under a single "root" `YOURDATAFOLDER` folder. We can also compare more than one algorithm by specifying several data result folders generated by different algorithms.

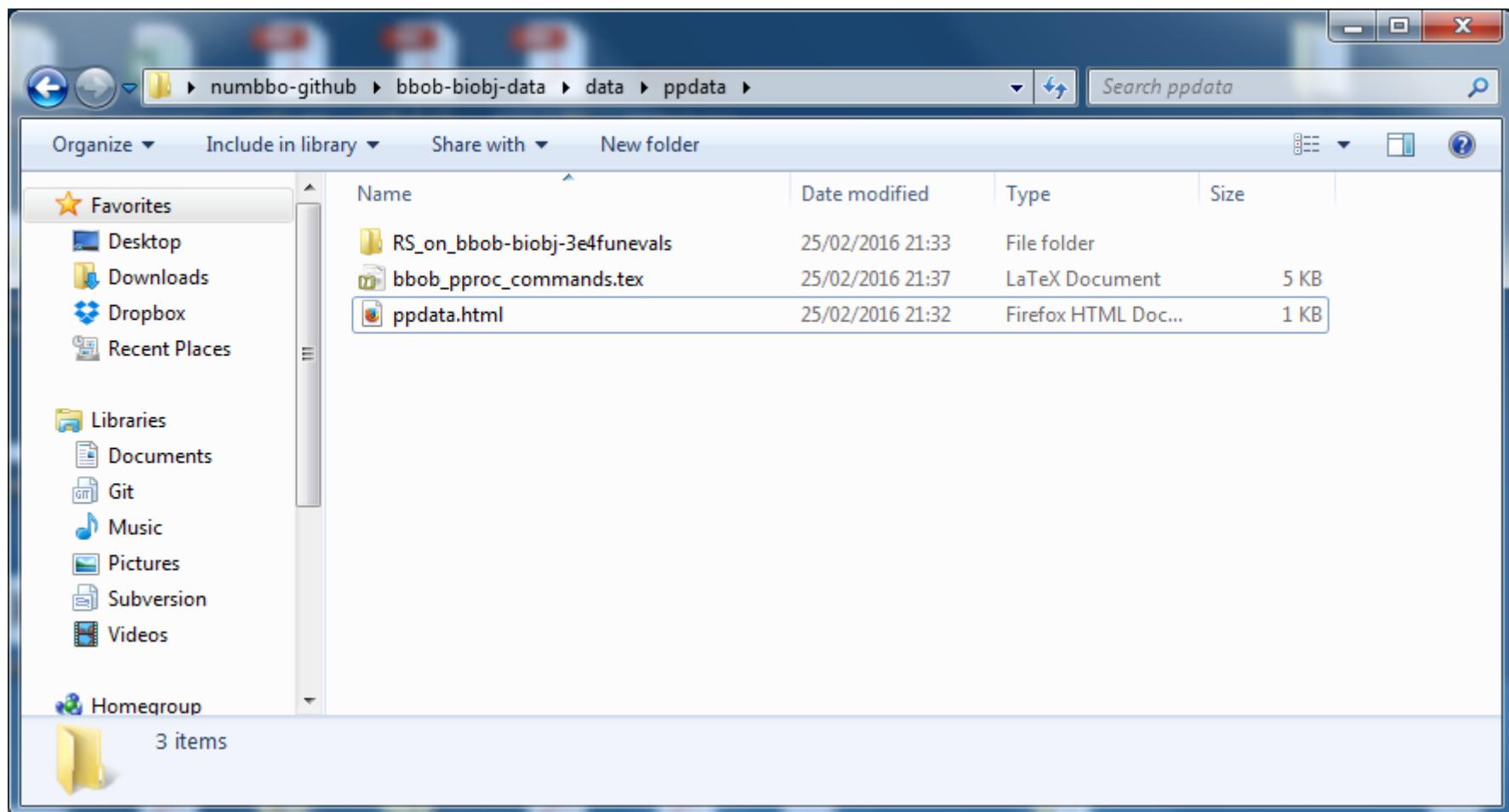
A folder, `ppdata` by default, will be generated, which contains all output from the post-processing, including a `ppdata.html` file, useful as main entry point to explore the result with a browser. Data might be overwritten, it is therefore useful to change the output folder name with the `-o OUTPUT_FOLDERNAME` option.

For the single-objective `bbob` suite, a summary pdf can be produced via LaTeX. The corresponding templates in ACM format can be found in the `code-postprocessing/latex-templates` folder. LaTeX templates for the multi-objective `bbob-biobj` suite will follow in a later release. A basic html output is also available in the result folder of the postprocessing (file `templateBBOBarticle.html`).

8. Once your algorithm runs well, **increase the budget** in your experiment script, if necessary implement randomized independent restarts, and follow the above steps successively until you are happy.

If you detect bugs or other issues, please let us know by opening an issue in our issue tracker at <https://github.com/numbbo/coco/issues>.

# result folder

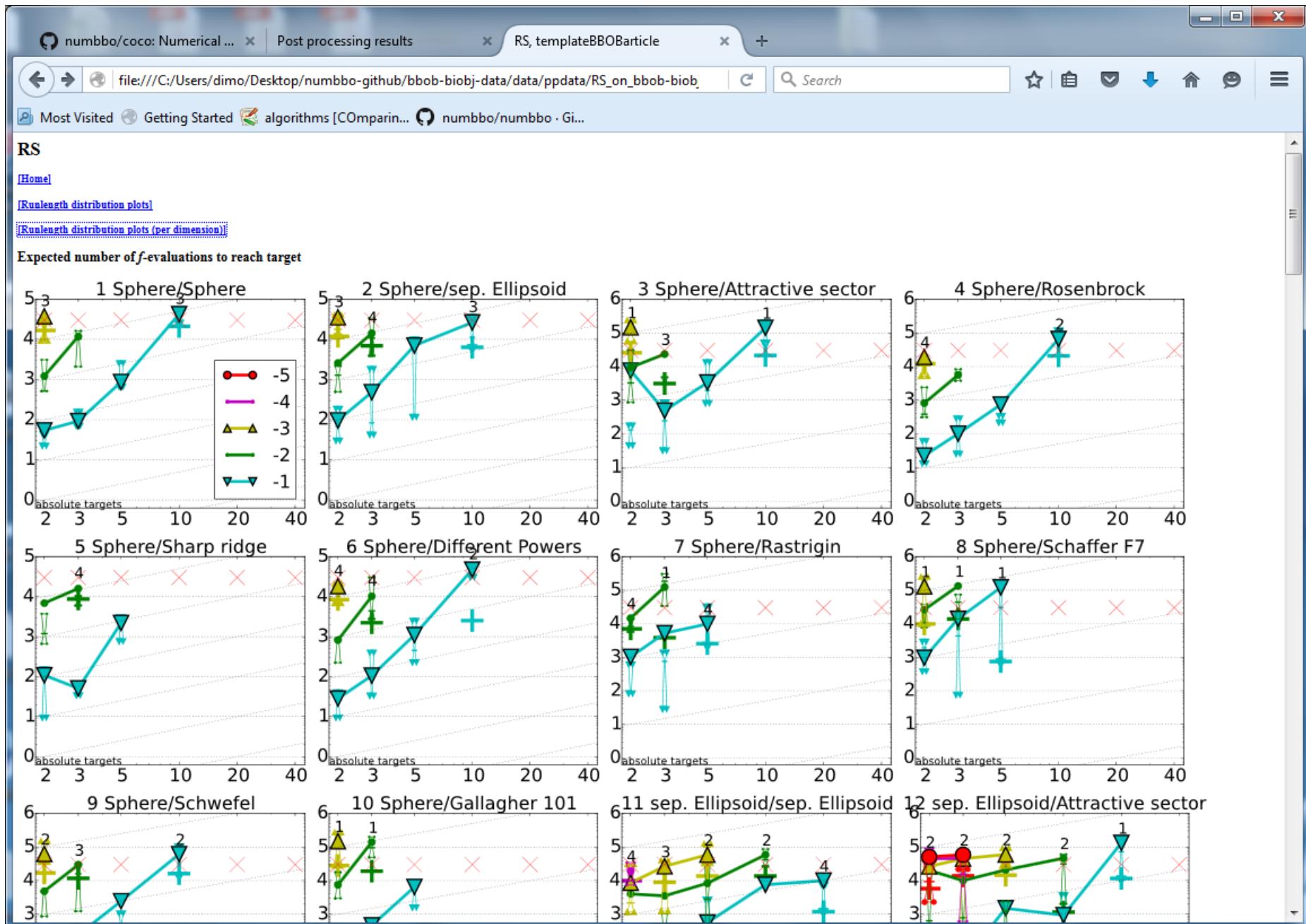


# automatically generated results

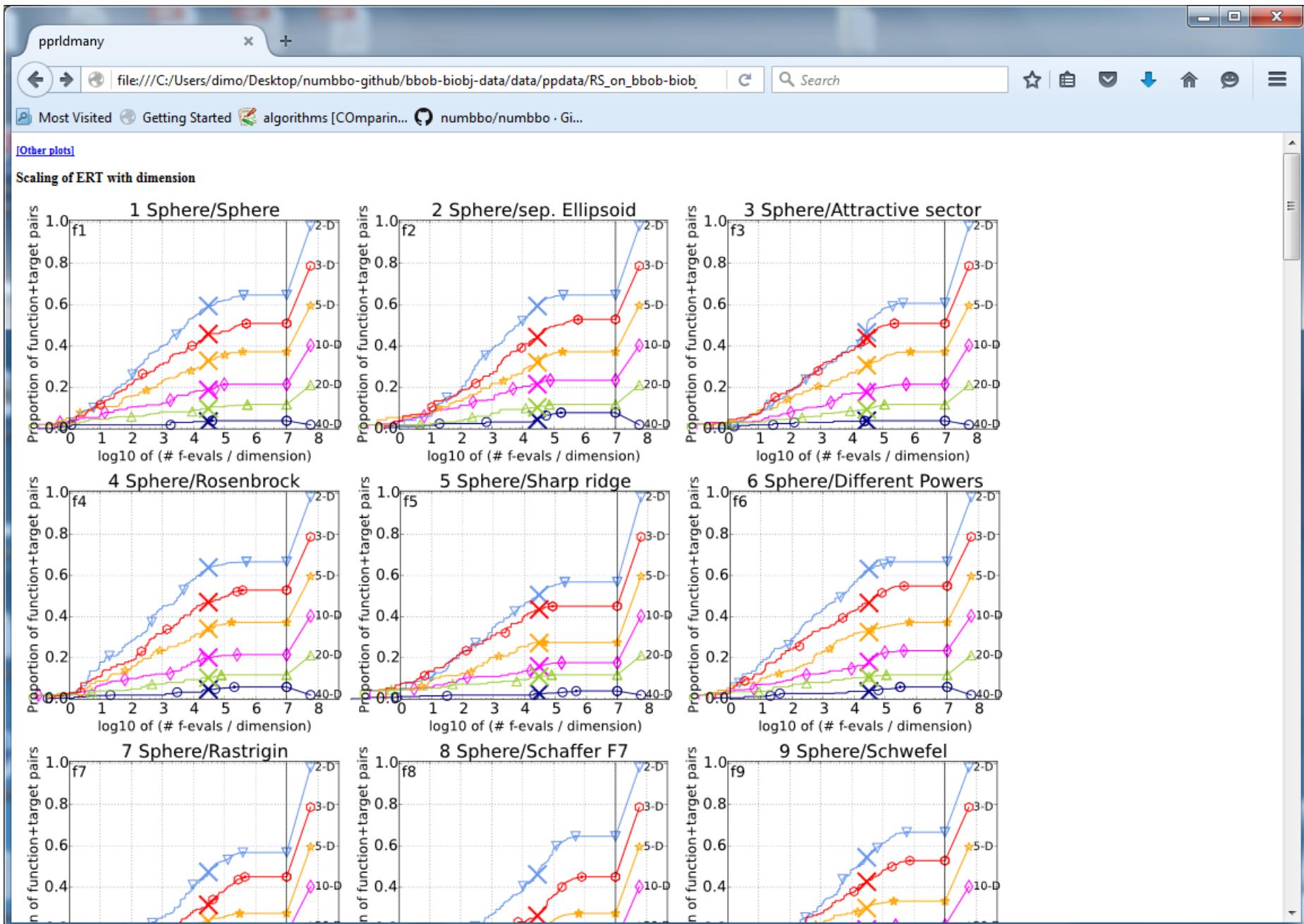
The screenshot shows a web browser window with the following details:

- Tab Bar:** The active tab is "Post processing results". Other tabs include "numbbo/coco: Numerical ..." and a "+" button for new tabs.
- Address Bar:** The URL is "file:///C:/Users/dimo/Desktop/numbbo-github/bbob-biobj-data/data/ppdata/ppdata.html".
- Toolbar:** Standard browser icons for back, forward, search, and other functions.
- Page Content:** The main content area displays the heading "Post processing results" and the sub-section "Single algorithm data". Below this, a blue underlined link "RS on bbo-biobj-3e4funevals" is visible.

## automatically generated results



# automatically generated results



**doesn't look too complicated, does it?**

[the devil is in the details ☺]

**so far:**

data for about 150 algorithm variants

118 workshop papers

by 79 authors from 25 countries

# Measuring Performance

On

- real world problems
  - expensive
  - comparison typically limited to certain domains
  - experts have limited interest to publish
- "artificial" benchmark functions
  - cheap
  - controlled
  - data acquisition is comparatively easy
  - problem of representativeness

# Test Functions

- define the "scientific question"  
the relevance can hardly be overestimated
- should represent "reality"
- are often too simple?  
remind separability
- a number of testbeds are around

# Available Test Suites in COCO

• bbob	24 noiseless fcts	140+ algo data sets
• bbob-noisy	30 noisy fcts	40+ algo data sets
• bbob-biobj	55 bi-objective fcts	 in 2016

## Under development:

- large-scale versions
- linearly constrained test suite

## Long-term goals:

- combining difficulties
- almost real-world problems
- real-world problems

# How Do We Measure Performance?

- Account for **invariance properties**  
prediction of performance is based on “similarity”,  
ideally equivalence classes of functions
- Meaningful **quantitative measure**
  - **quantitative** on the ratio scale (highest possible)  
"algo A is two *times* better than algo B" is a **meaningful statement**
  - assume a wide range of values
  - **meaningful (interpretable)** with regard to the real world  
possible to transfer from benchmarking to real world

runtime or **first hitting time** is the prime candidate  
(we don't have many choices anyway)

# How Do We Measure Performance?

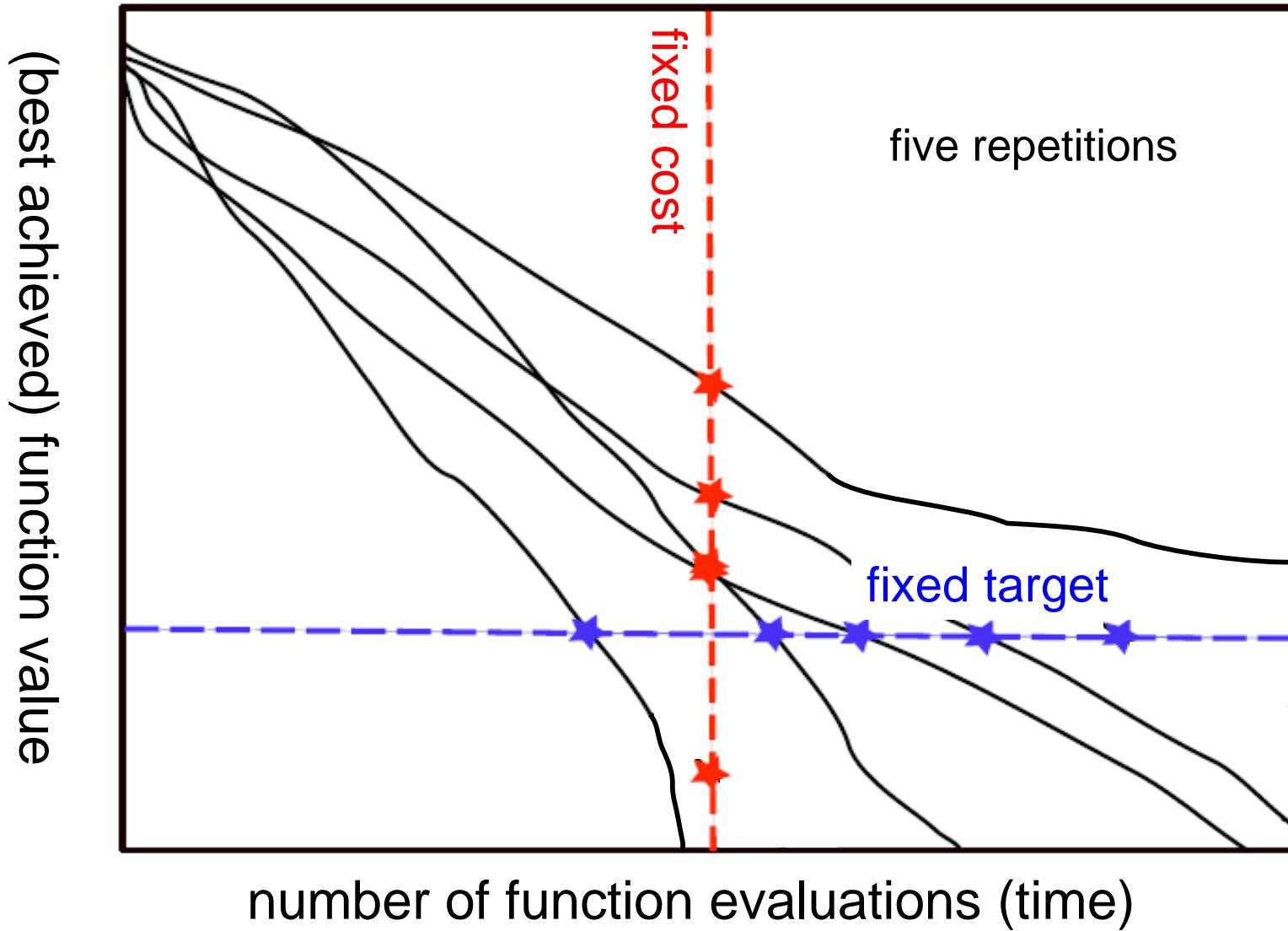
Two objectives:

- Find solution with small(est possible) **function value**
- With the least possible **search costs** (number of function evaluations)

For measuring performance: fix one and measure the other

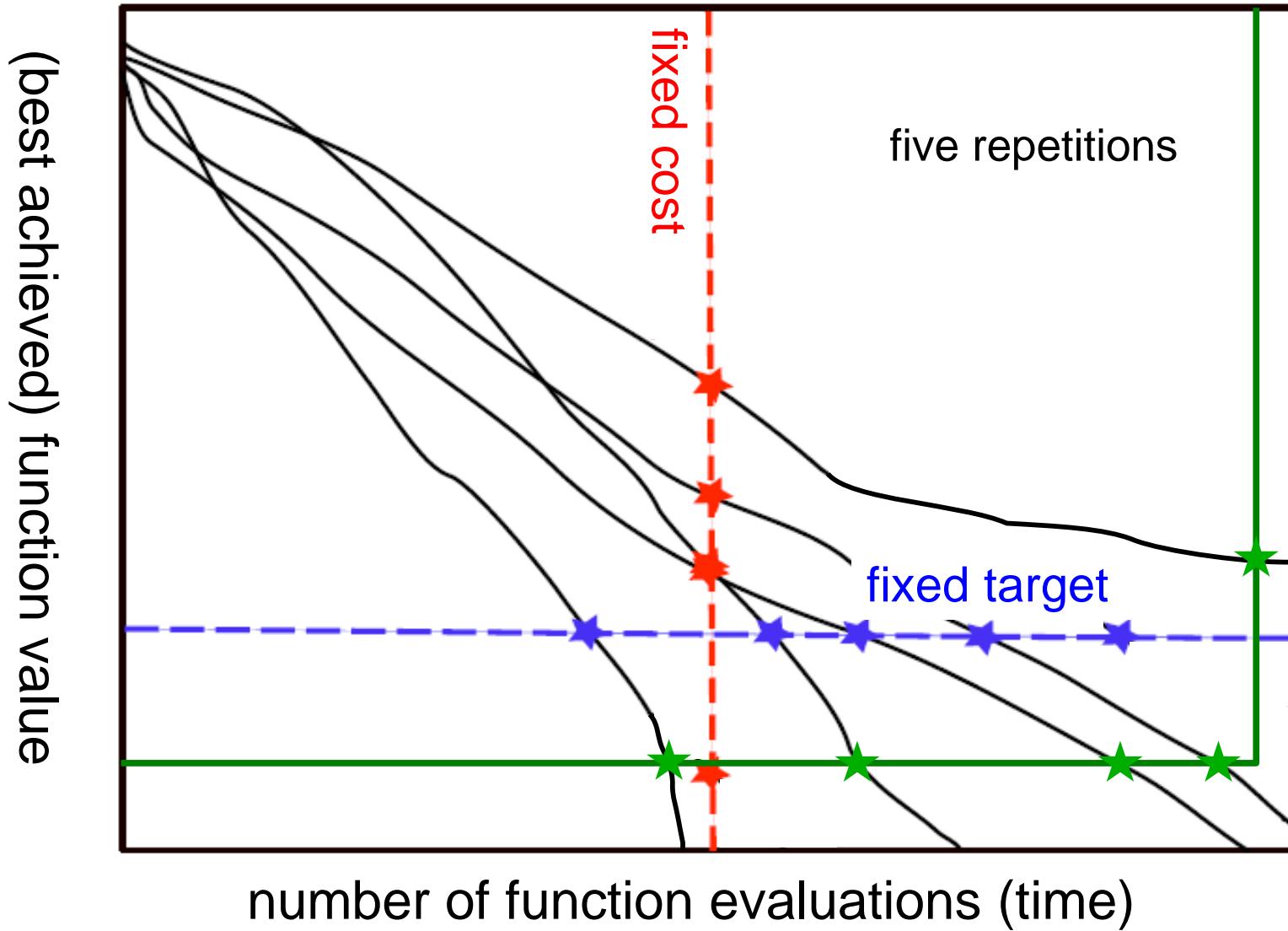
# Measuring Performance Empirically

convergence graphs is all we have to start with...



# Measuring Performance Empirically

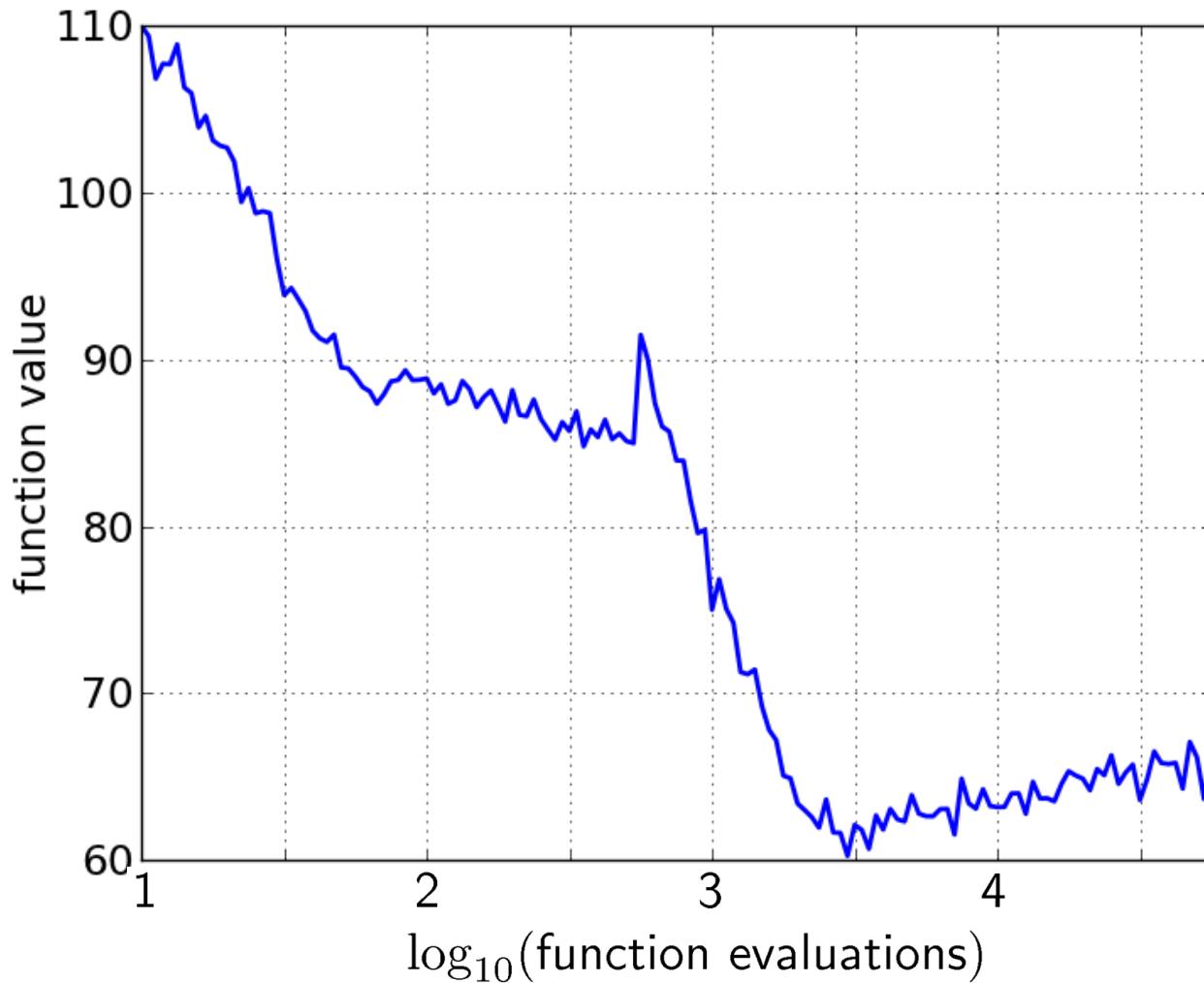
convergence graphs is all we have to start with...



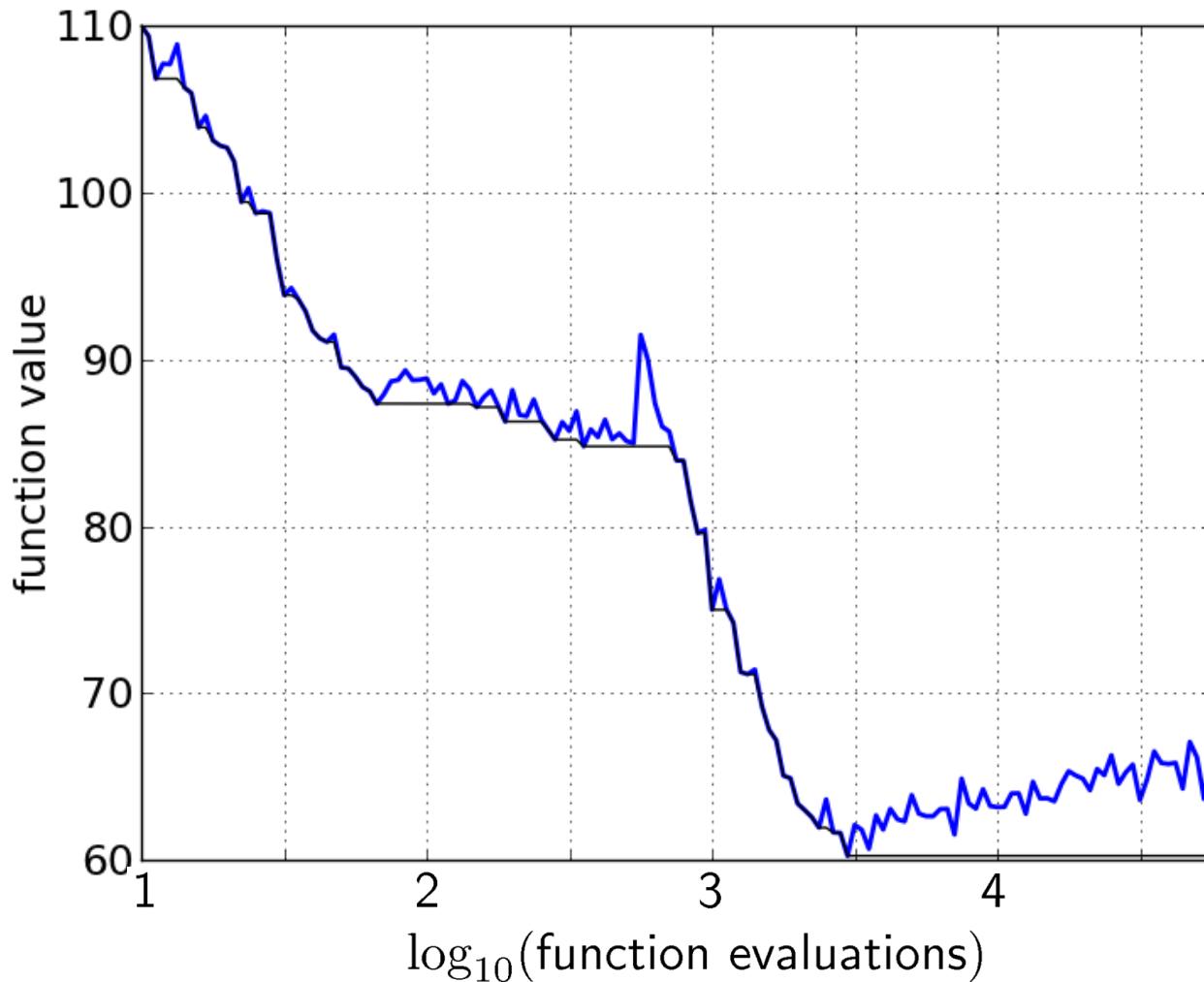
**ECDF:**

Empirical Cumulative Distribution Function of the Runtime  
[aka data profile]

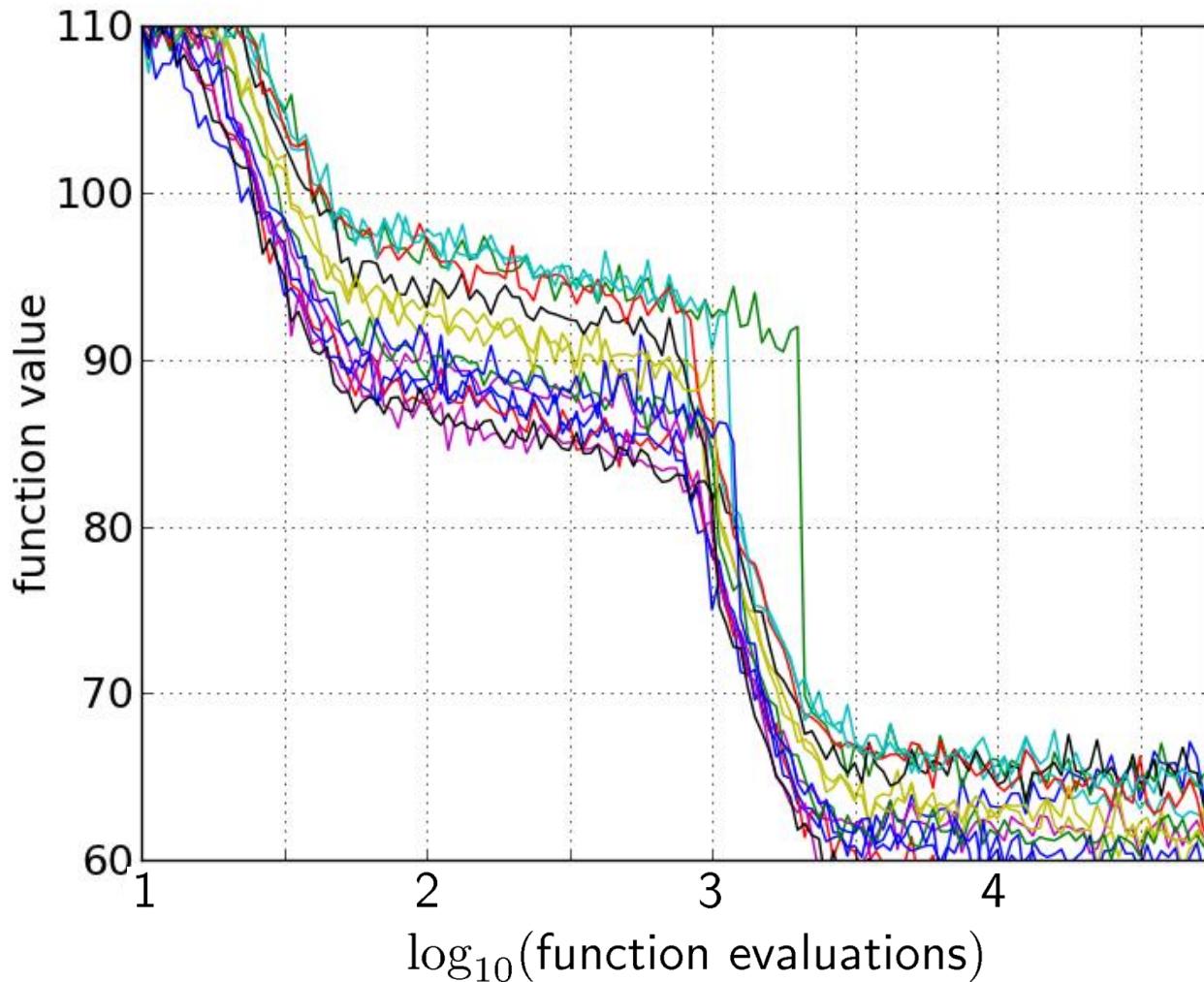
# A Convergence Graph



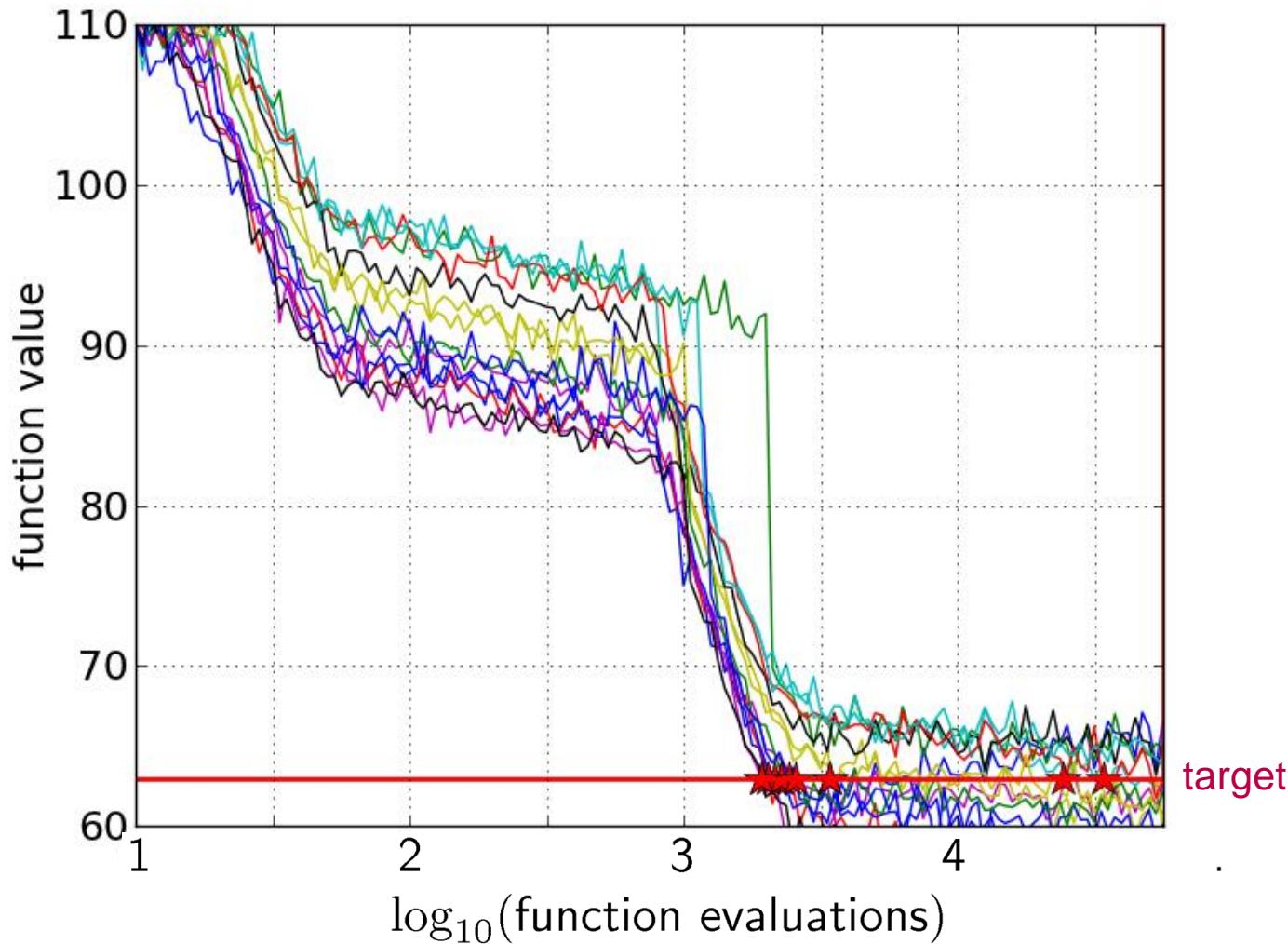
# First Hitting Time is Monotonous



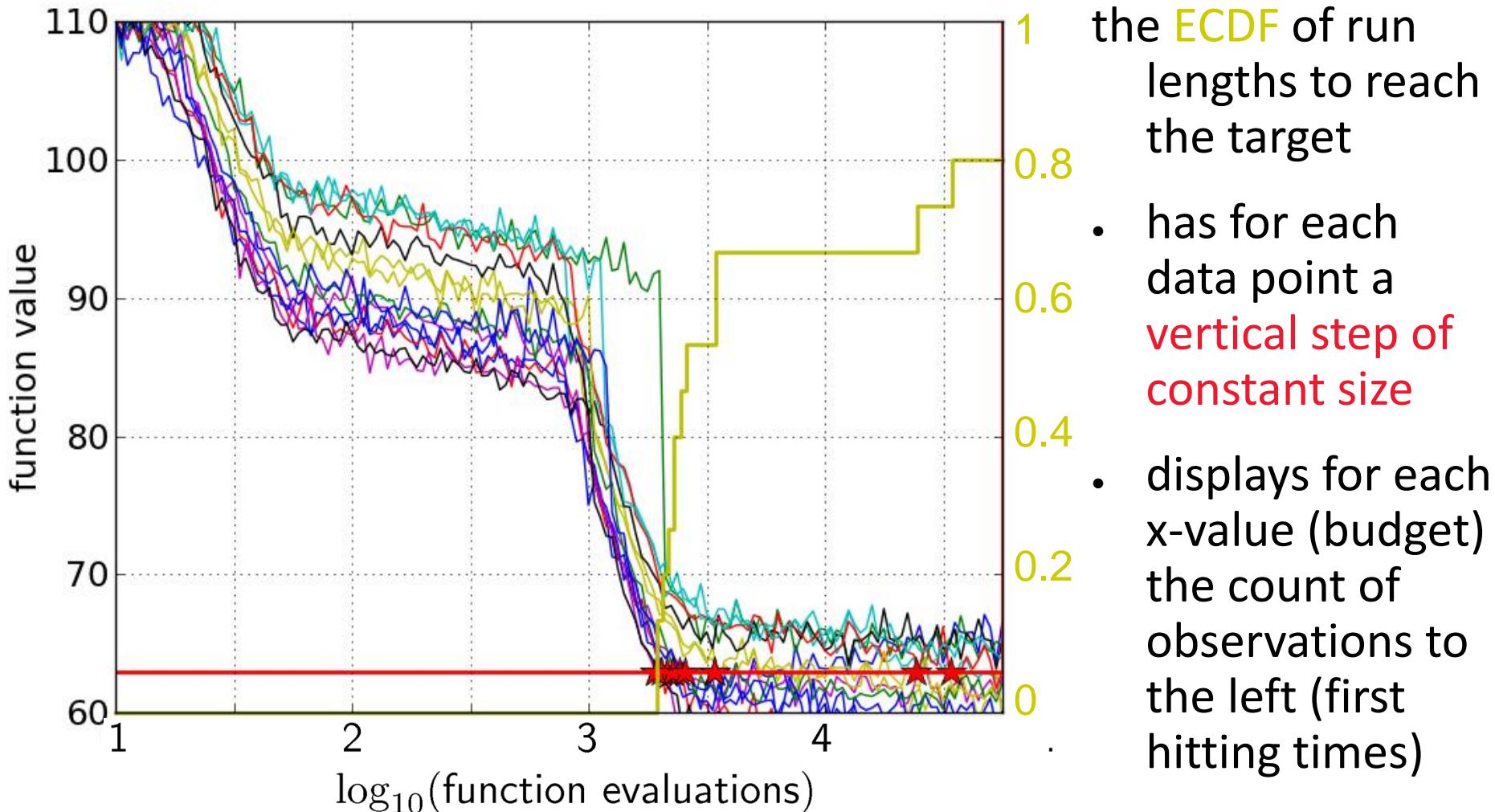
# 15 Runs



# 15 Runs $\leq$ 15 Runtime Data Points

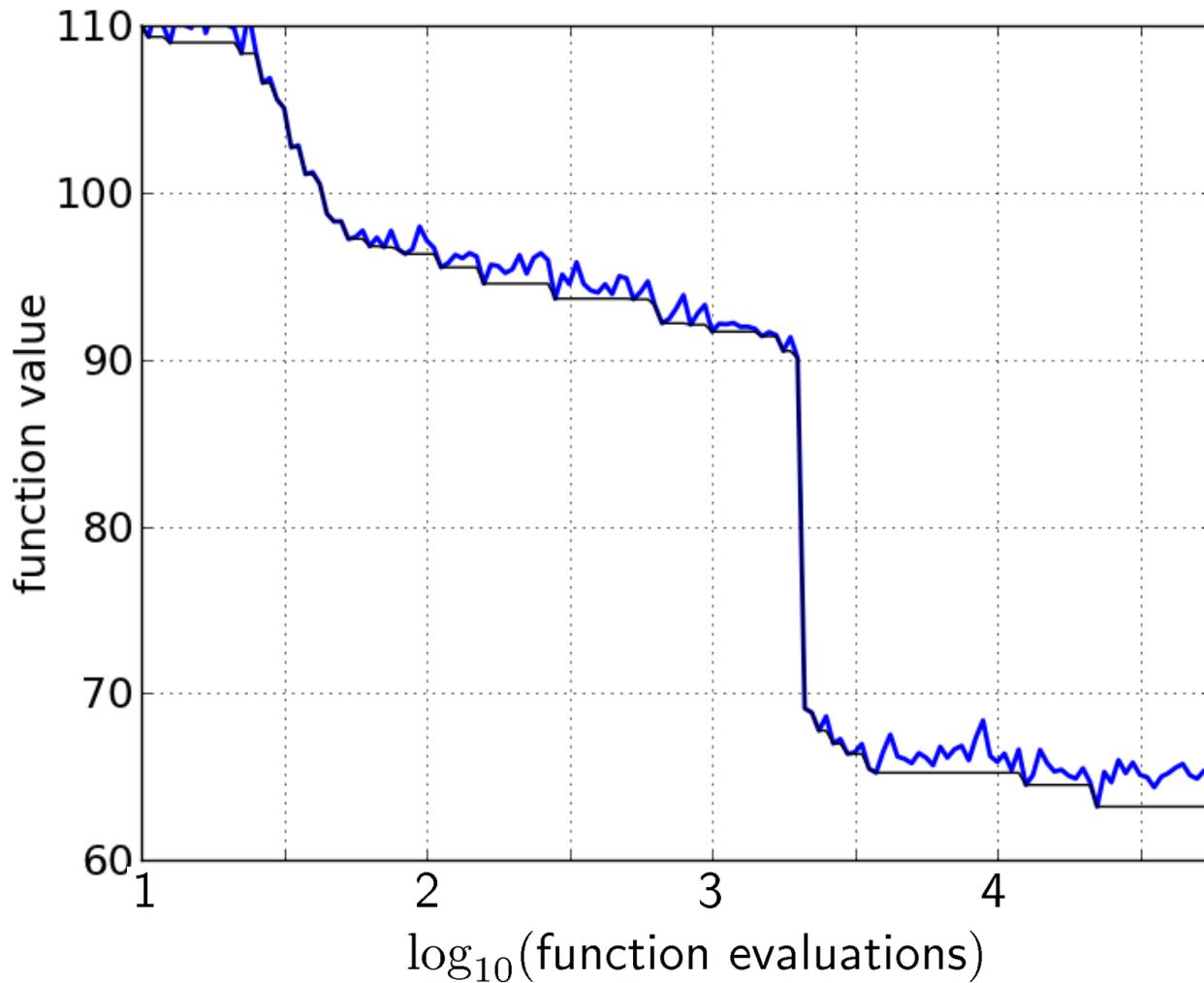


# Empirical Cumulative Distribution

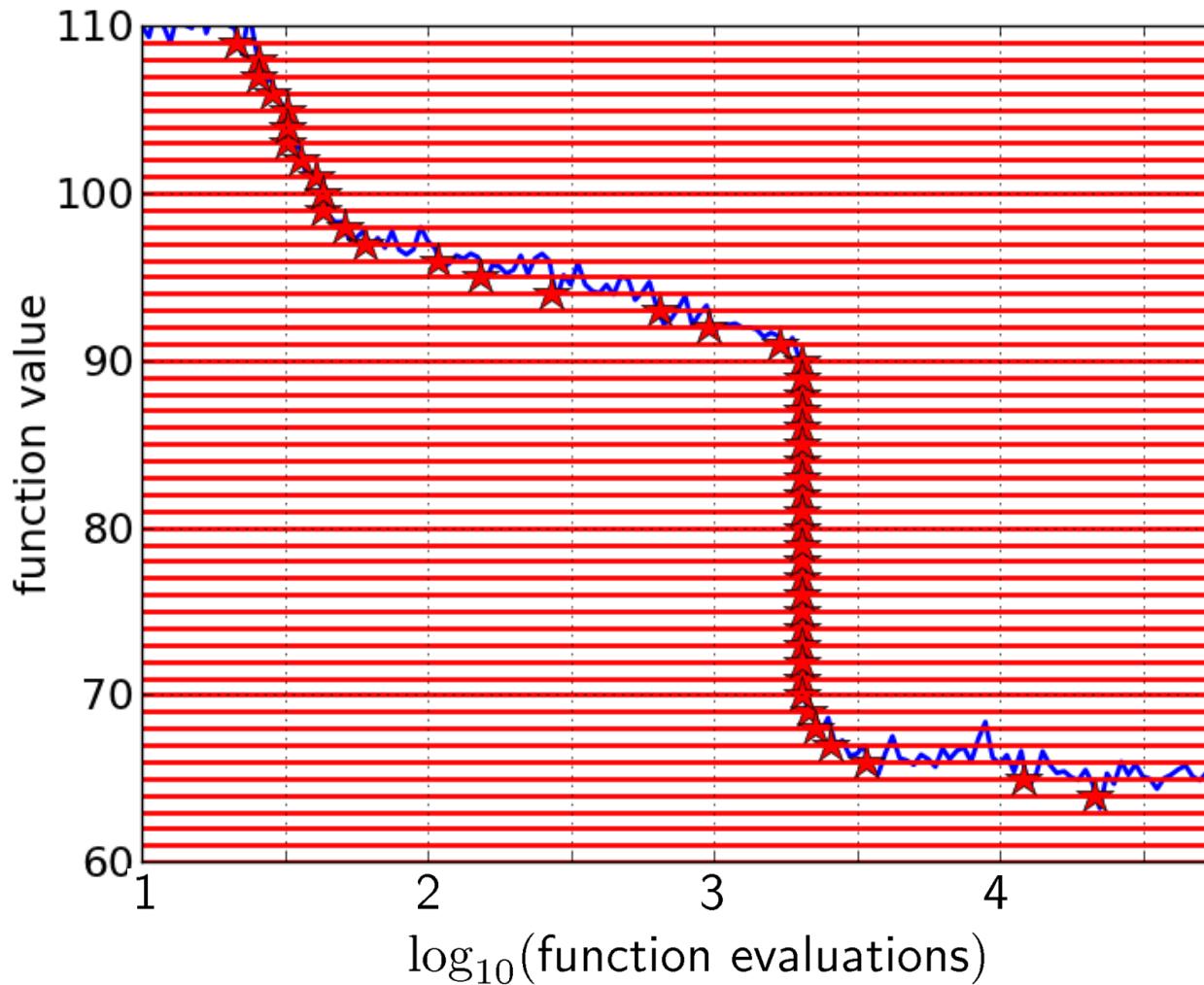


e.g. 60% of the runs need between 2000 and 4000 evaluations  
80% of the runs reached the target

# Reconstructing A Single Run

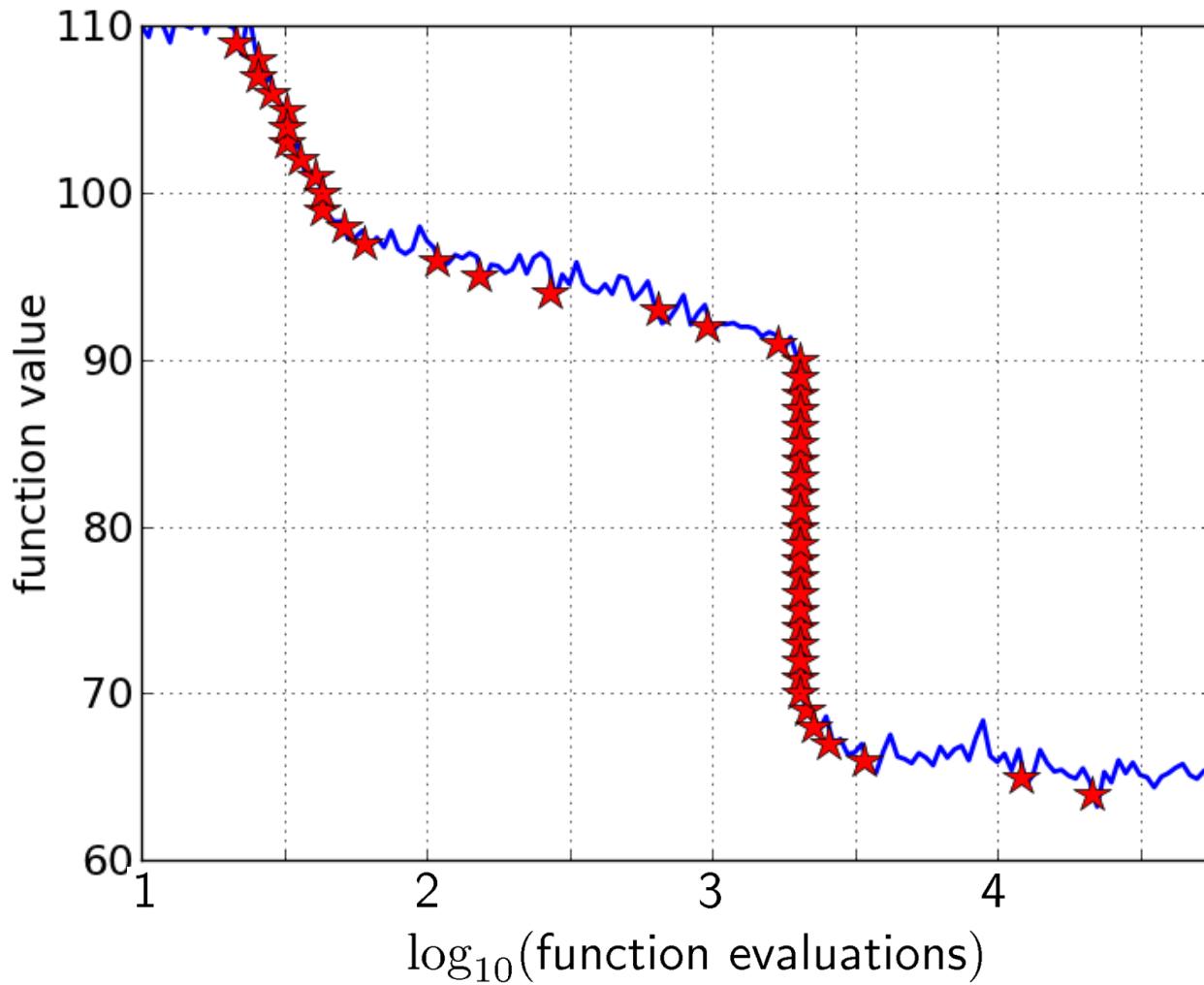


# Reconstructing A Single Run

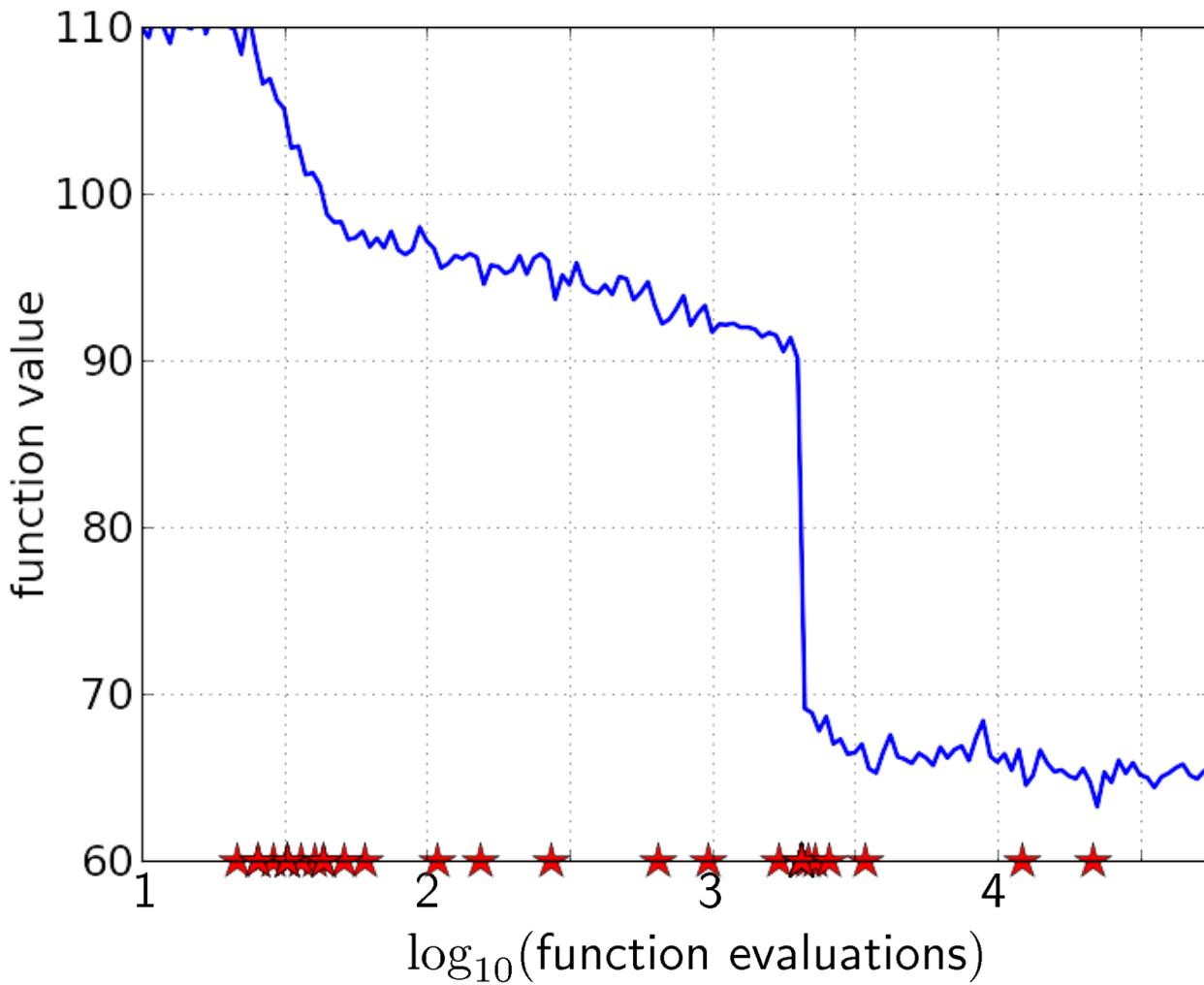


50 equally  
spaced targets

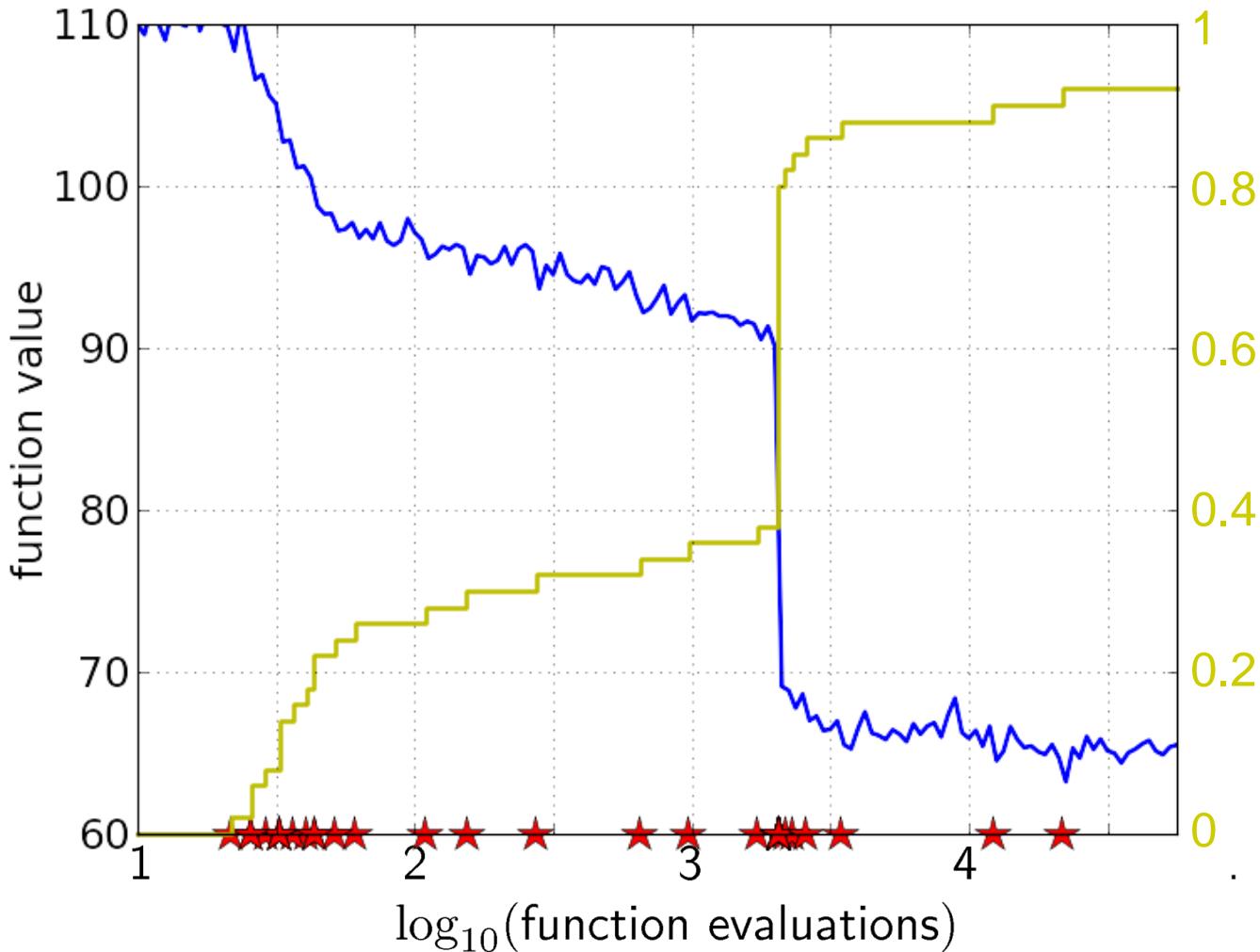
# Reconstructing A Single Run



# Reconstructing A Single Run

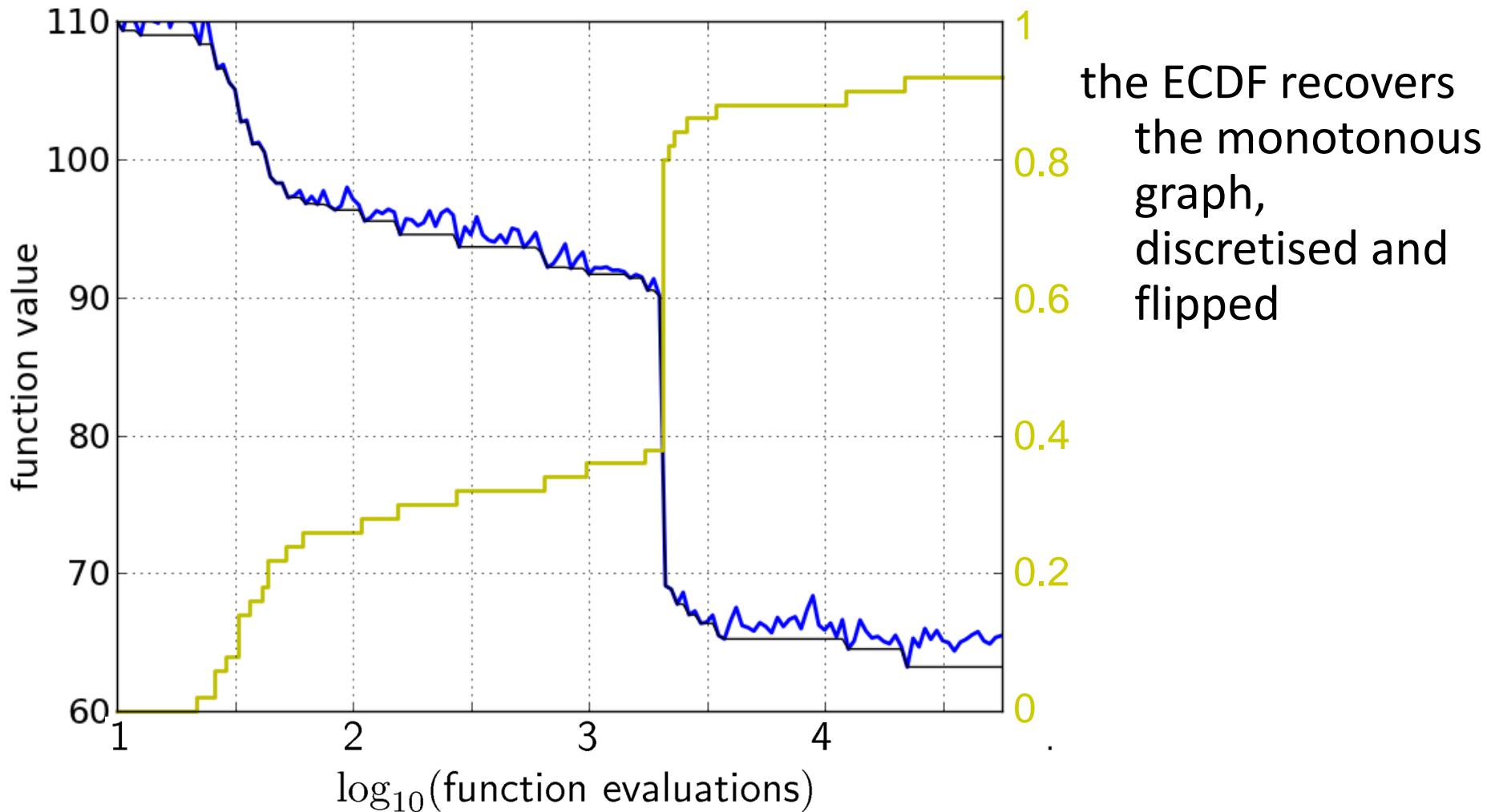


# Reconstructing A Single Run

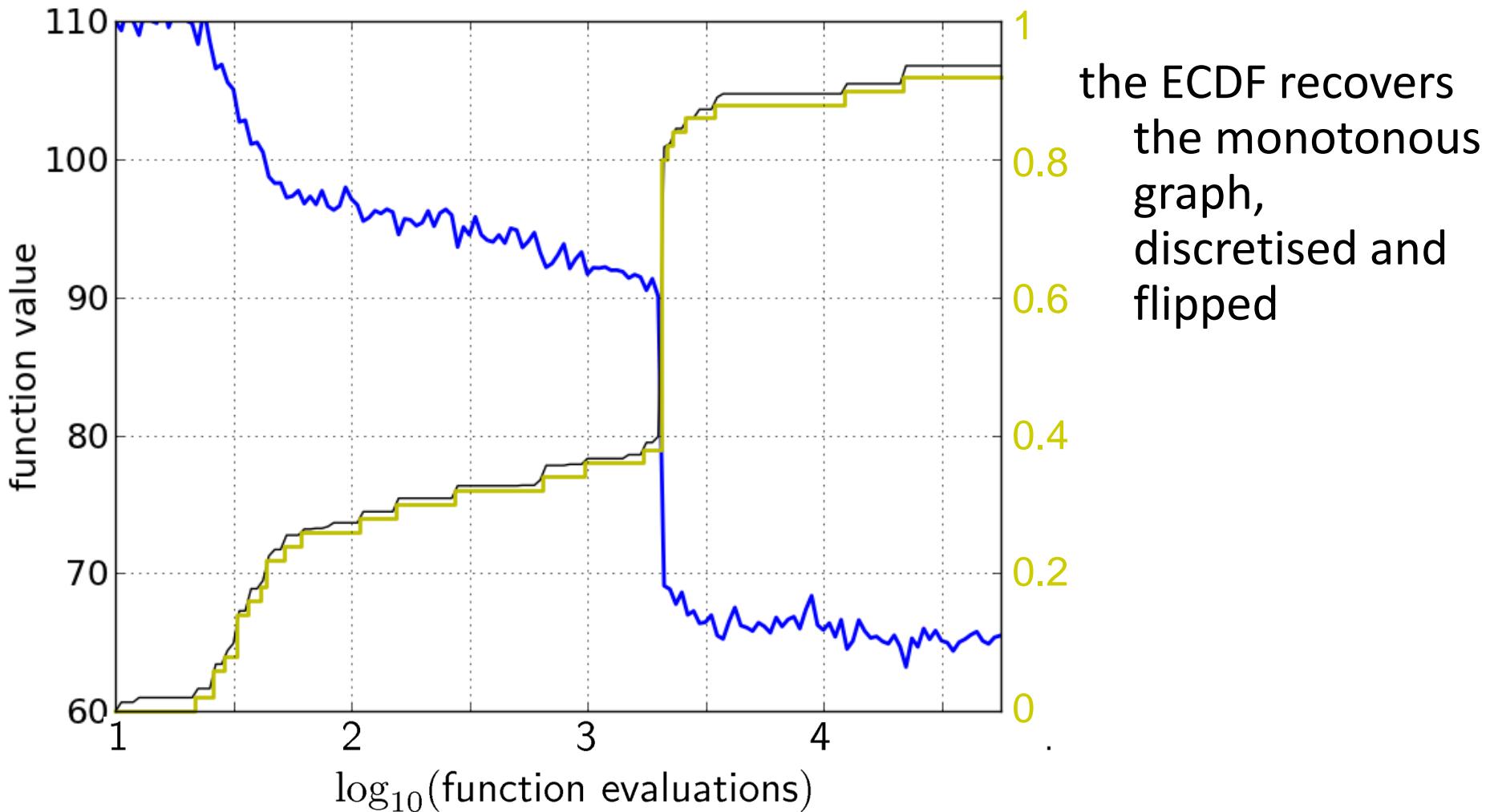


the empirical CDF makes a step for each star, is monotonous and displays for each budget the fraction of targets achieved within the budget

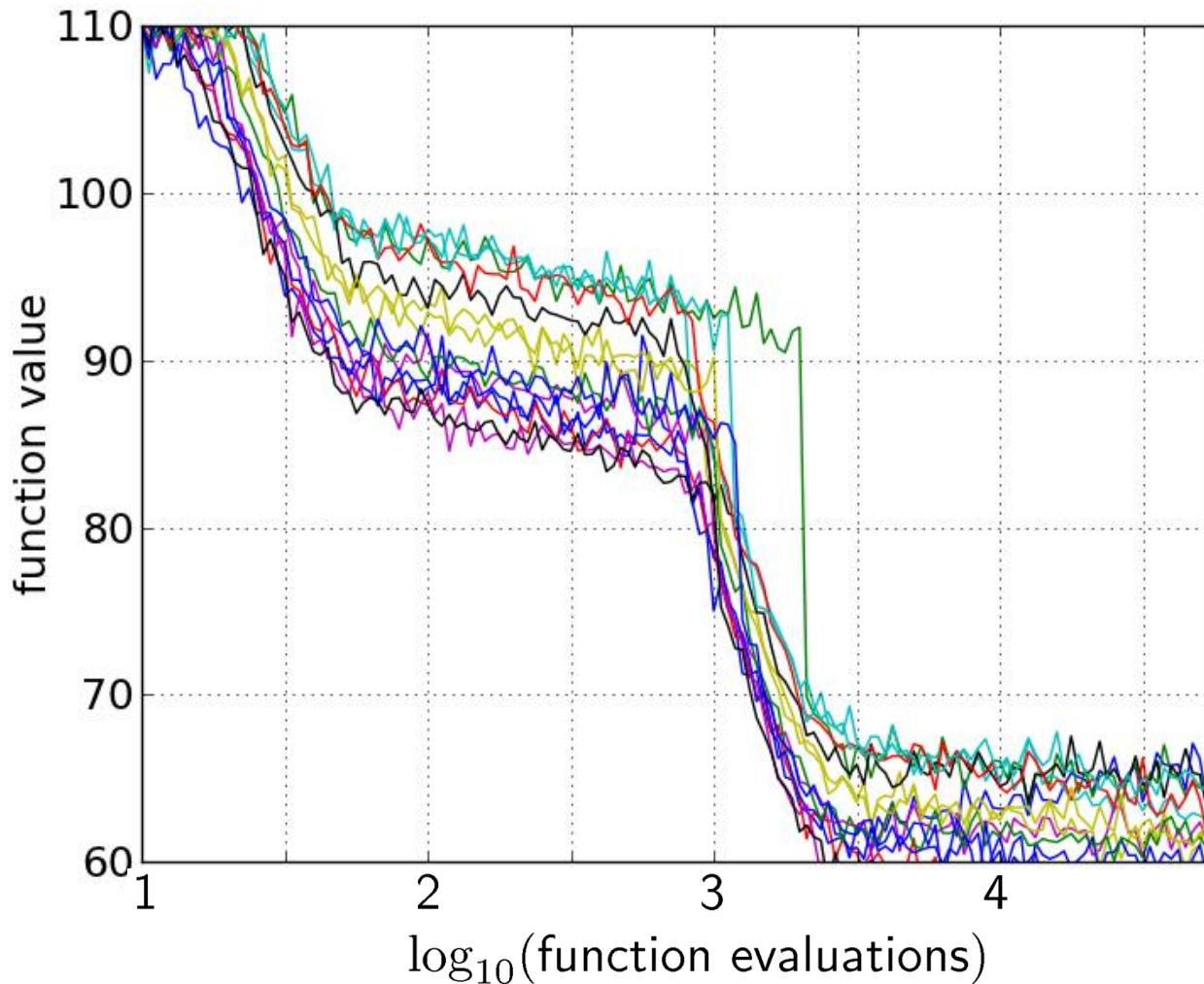
# Reconstructing A Single Run



# Reconstructing A Single Run

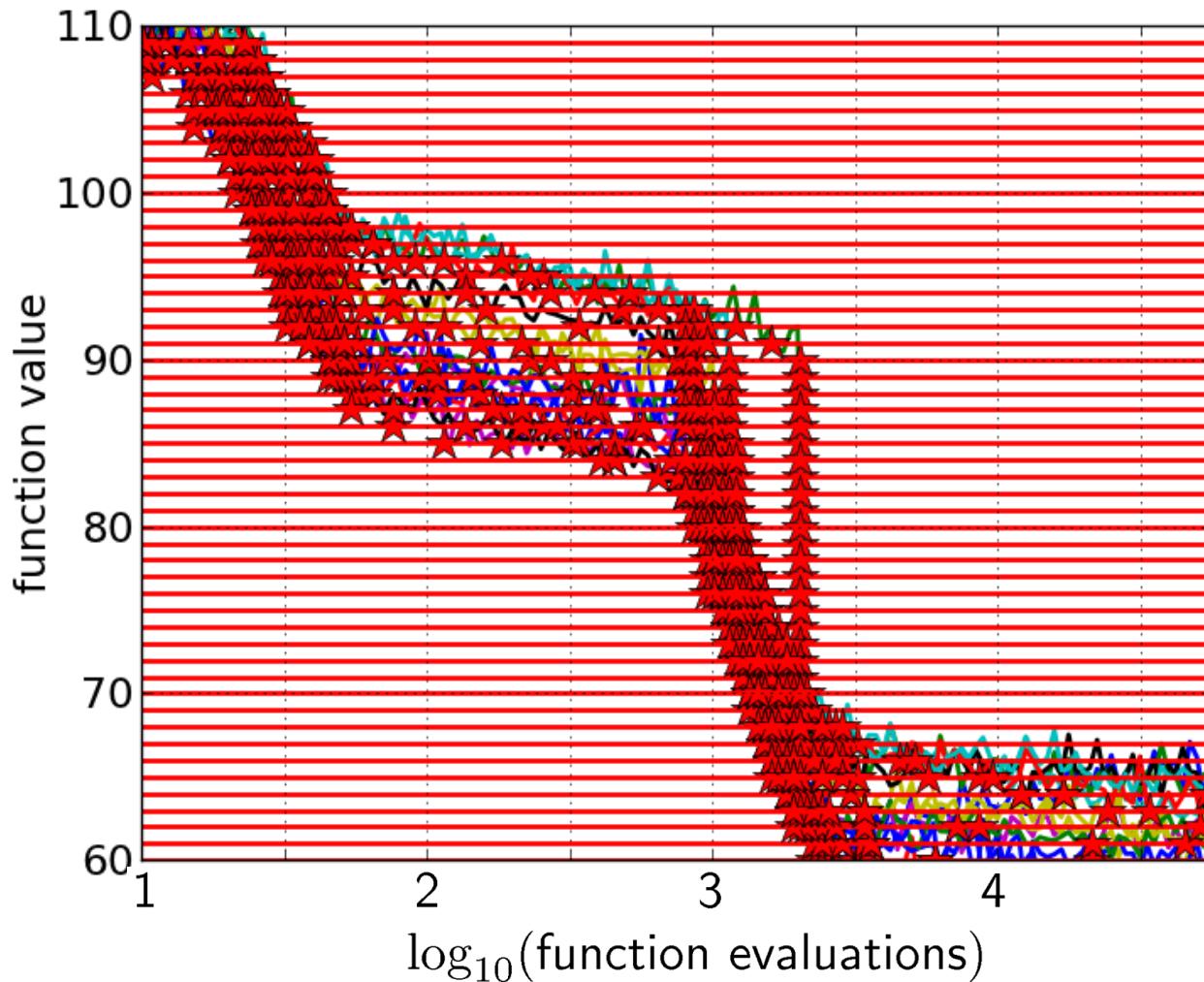


# Aggregation



15 runs

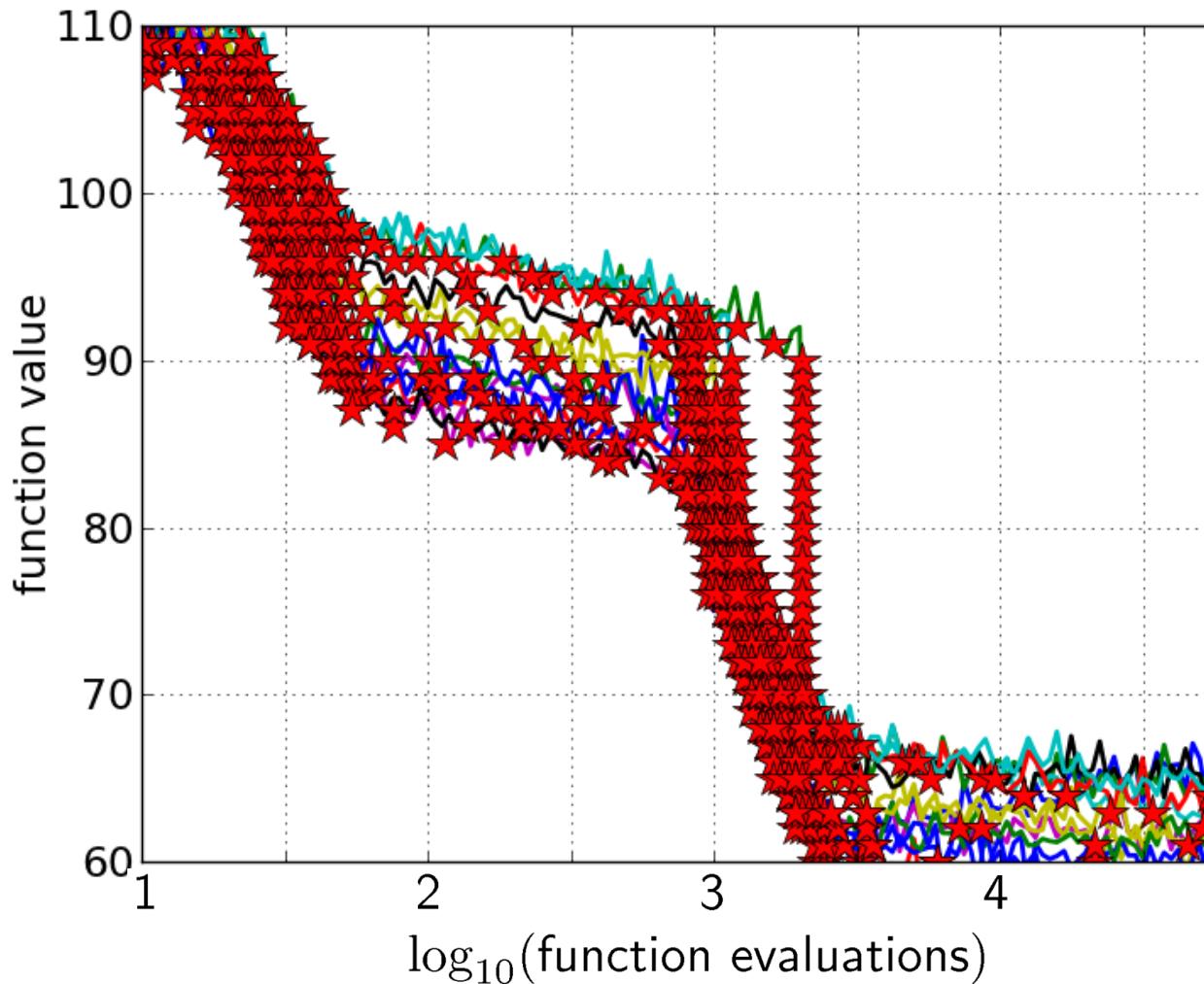
# Aggregation



15 runs

50 targets

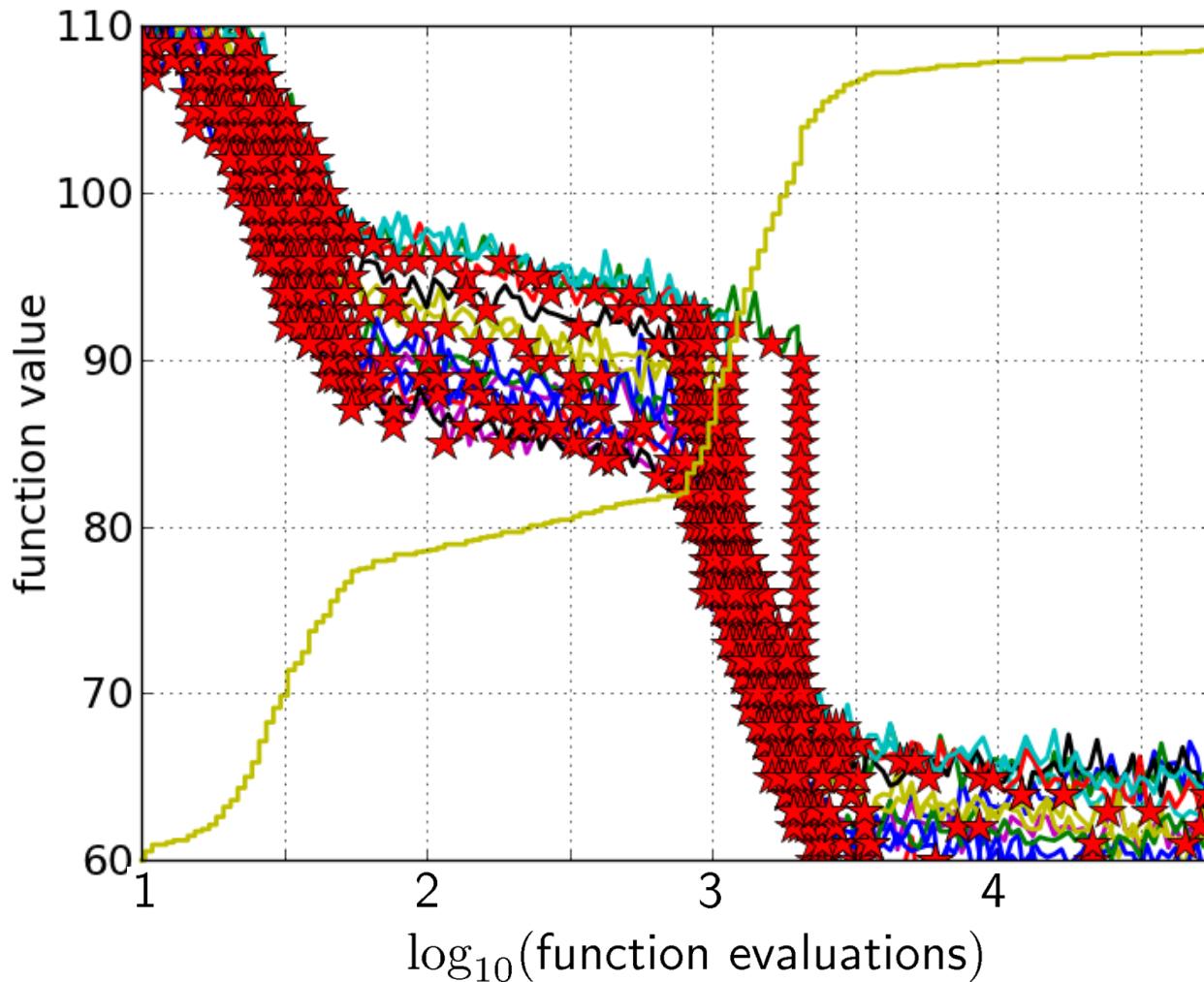
# Aggregation



15 runs

50 targets

# Aggregation

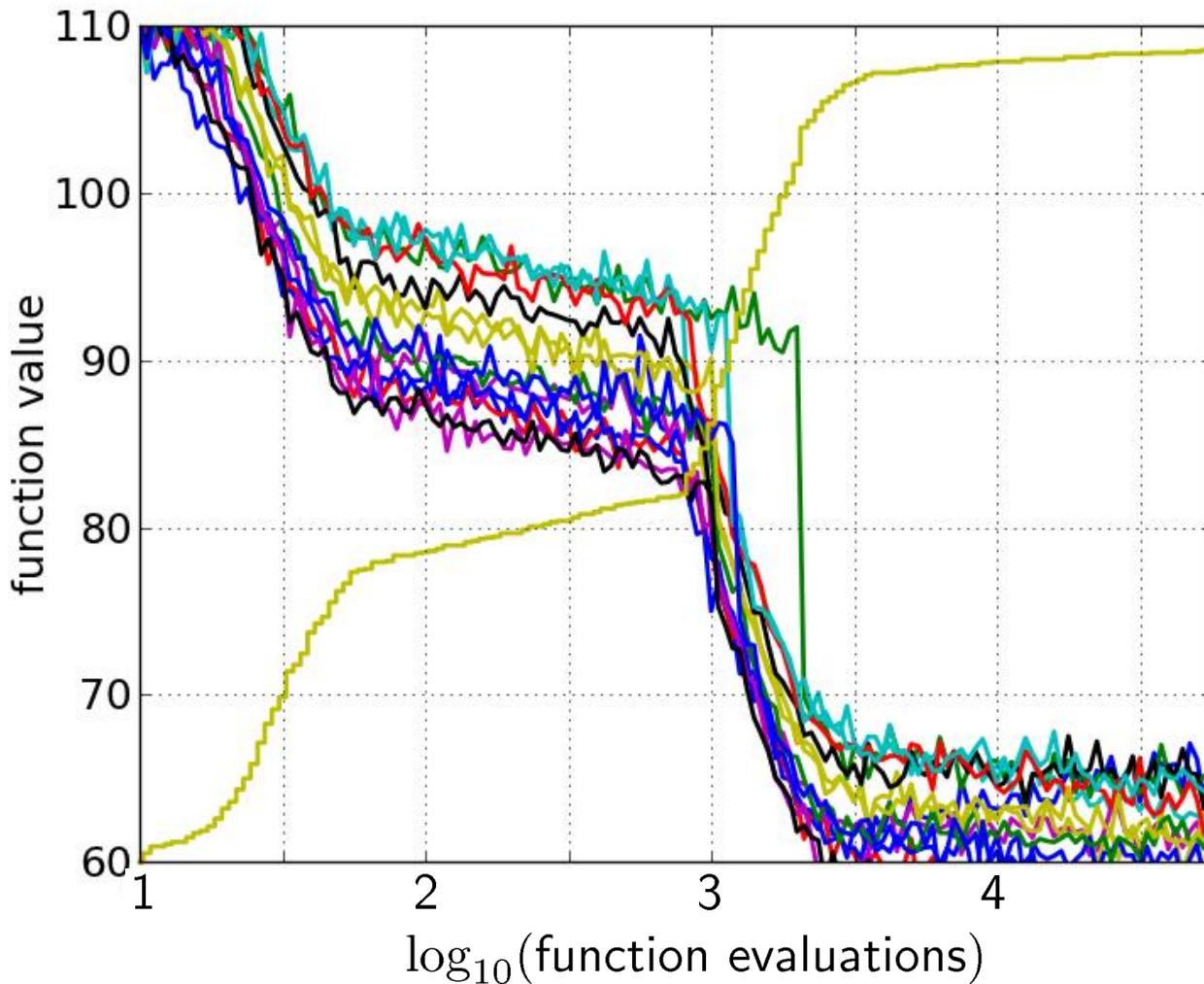


15 runs

50 targets

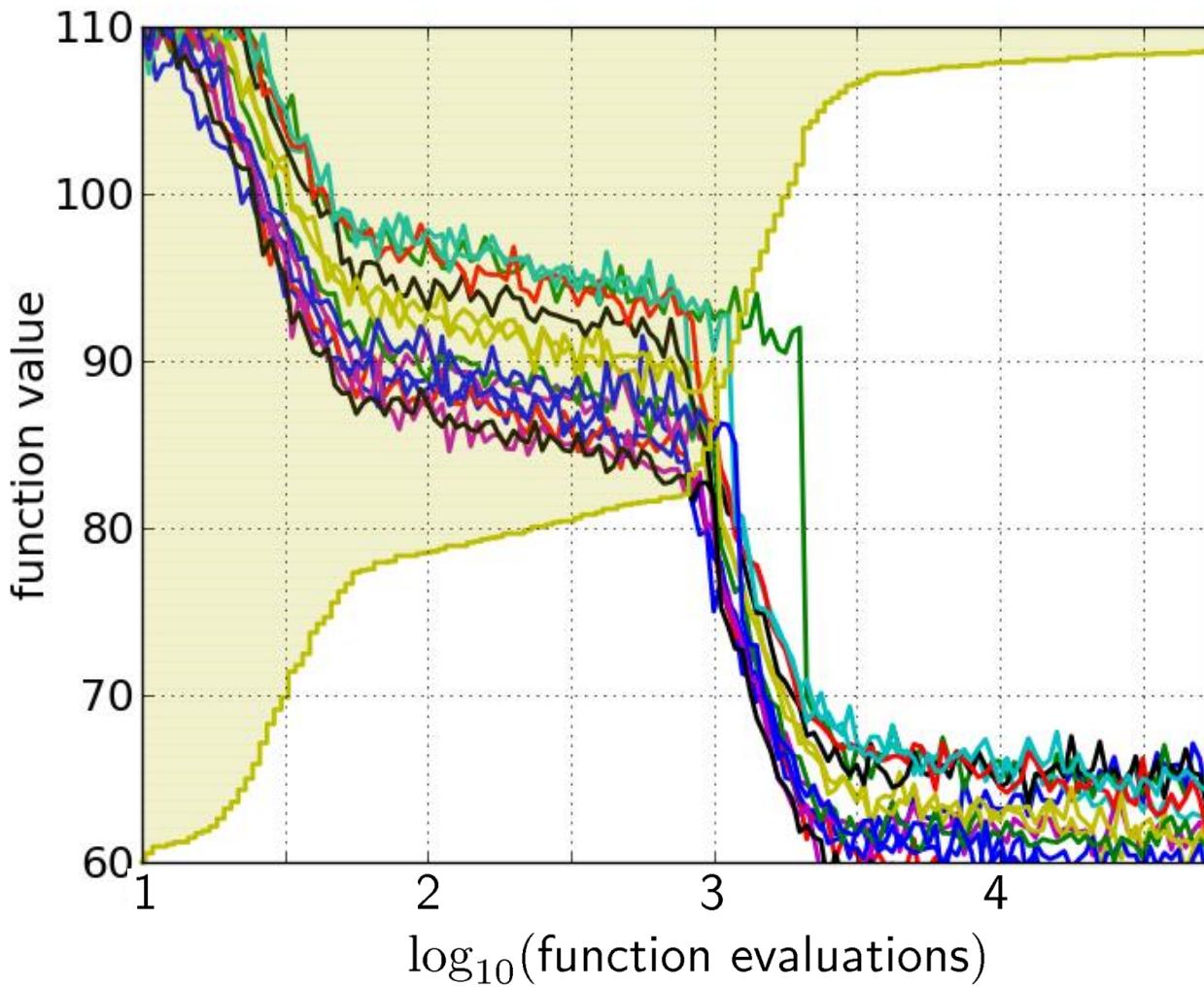
ECDF with 750  
steps

# Aggregation



50 targets from  
15 runs  
...integrated in a  
single graph

# Interpretation



50 targets from  
15 runs  
integrated in a  
single graph

area over the ECDF  
curve  
=

average log runtime  
(or geometric avg.  
runtime) over all  
targets (difficult and  
easy) and all runs

phew ;-)

start [COmparing Continu... [+>](#)

coco.gforge.inria.fr [Search](#) [Star](#) [Email](#) [Download](#) [Home](#) [Help](#) [Feedback](#) [☰](#)

Most Visited Getting Started [algorithms \[COmparin...](#) [numbbo/numbbo - Gi...](#)

## [[start]]

### COMPARING CONTINUOUS OPTIMISERS: COCO

Show pagesource Old revisions Recent changes Sitemap Login

Navigation

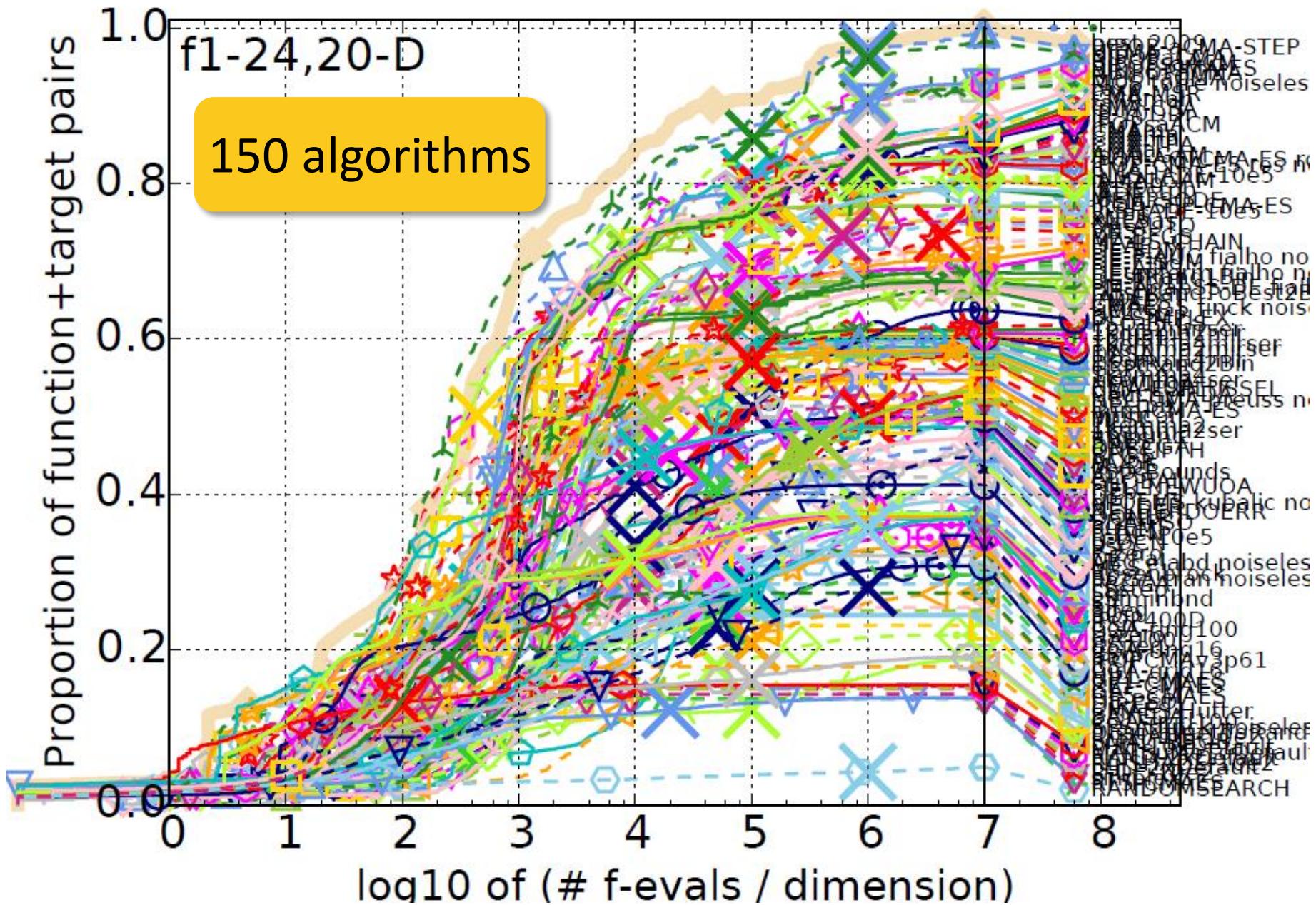
- Home
- Documentation
- download latest old code
- new code homepage
- download new code directly
- BBOB 2016
- BBOB 2015 @ GECCO
  - Algorithms
  - Results
  - Schedule
  - Downloads
- BBOB 2015 @ CEC
  - Algorithms
  - Results
  - Downloads
- BBOB 2013
  - Algorithms
  - Results
  - Schedule
  - Downloads
- BBOB 2012
  - Algorithms
  - Results
  - Downloads
- BBOB 2010
  - Algorithms
  - Results
  - Downloads

Proportion of functions

Running length / dimension

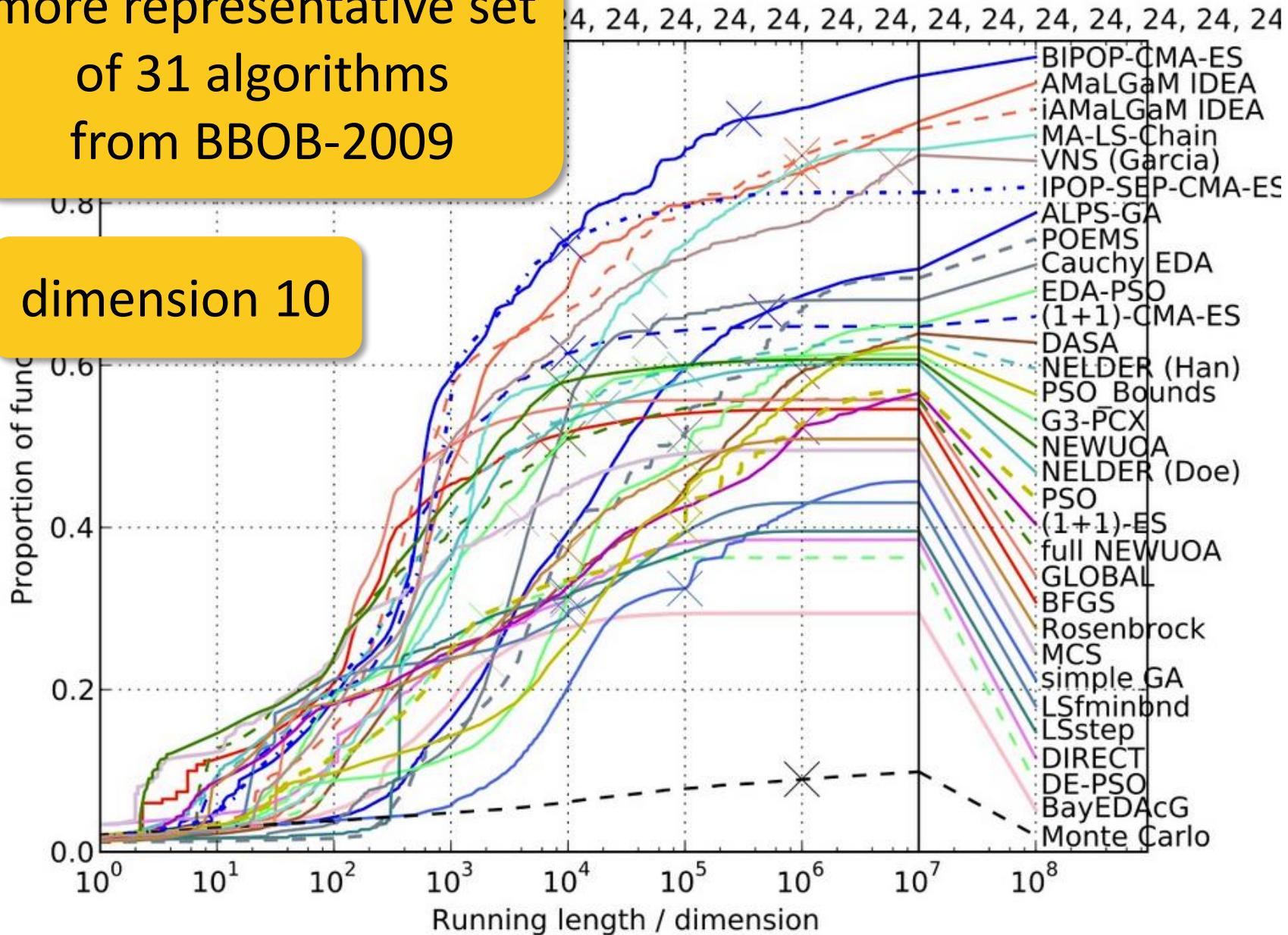
To subscribe to (or unsubscribe from) the bbbob discussion mailing list follow this link <http://lists.iri.fr/cgi-bin/mailman/listinfo/bbbob-discuss>.

To receive announcements related to the BBOB workshops simply send an email to BBOB team



more representative set  
of 31 algorithms  
from BBOB-2009

dimension 10



# Recommendations

## Results of BBOB 2009-2015 (noiseless bbob test suite)

- low dimension (2-D, 3-D), small budget: **Nelder Mead** (implementation by B. Doerr et al.)
- very small budget (<50D funevals): **BBOB-SMAC** (an EGO variant)
- larger dimension, small budget (<500D funevals): **NEWUOA, MCS**
- larger dimension, large budget (>500D funevals): **CMA-ES** variants
- portfolio algorithms can combine the best of all worlds, e.g. **HCMA**

# Further Observations

## Invariance

- many algorithms are not invariant under affine transformations of the search space (for example PSO), others like CMA-ES variants are
- as a consequence, other types of algorithms are good for separable and non-separable problems

## Robustness of Algorithms

- CMA-ES was tested in several variants, in several programming languages and with several, slightly changing settings
- nevertheless, it showed throughout its variants consistently good behavior and can be considered as a robust algorithm if the budget is not too low

# The Future of COCO

- bi-objective data coming up (BBOB workshop@GECCO'16)
- towards more realistic problems
  - constraints
  - "almost real-world" problems
- online visualization of data

**Thank you.**

GECCO workshop on

**Black Box Optimization Benchmarking (BBOB'2016)**

**with a focus on bi-objective problems**

organizers:

Anne Auger, Dimo Brockhoff, Nikolaus Hansen,  
Tea Tušar, Dejan Tušar, Tobias Wagner

submission deadline: **April 17, 2016**

<http://numbbo.github.io/workshops/BBOB-2016/>

<https://github.com/numbbo/coco>