

8th GECCO Workshop on Blackbox Optimization Benchmarking (BBOB): Welcome and Introduction to COCO/BBOB

The BBOBies

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

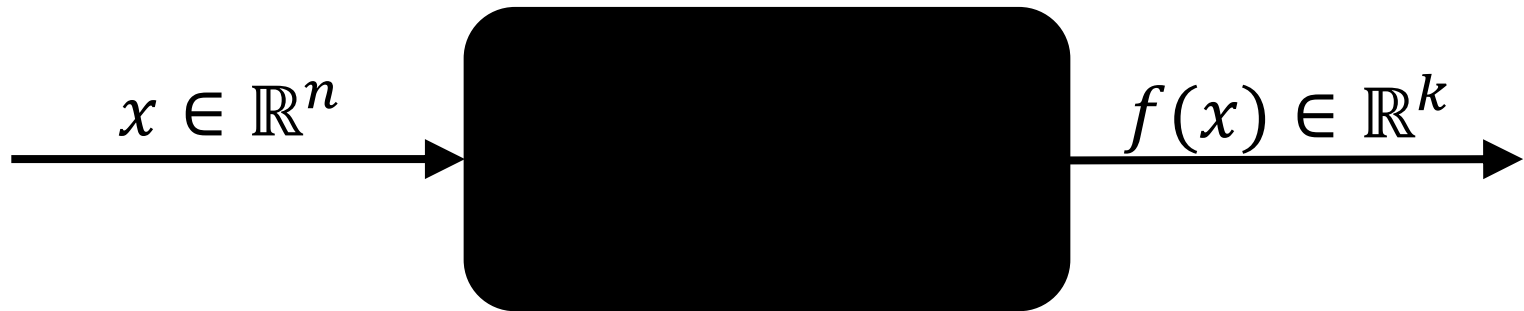


slides based on previous ones by A. Auger, N. Hansen, and D. Brockhoff

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

Numerical Blackbox Optimization

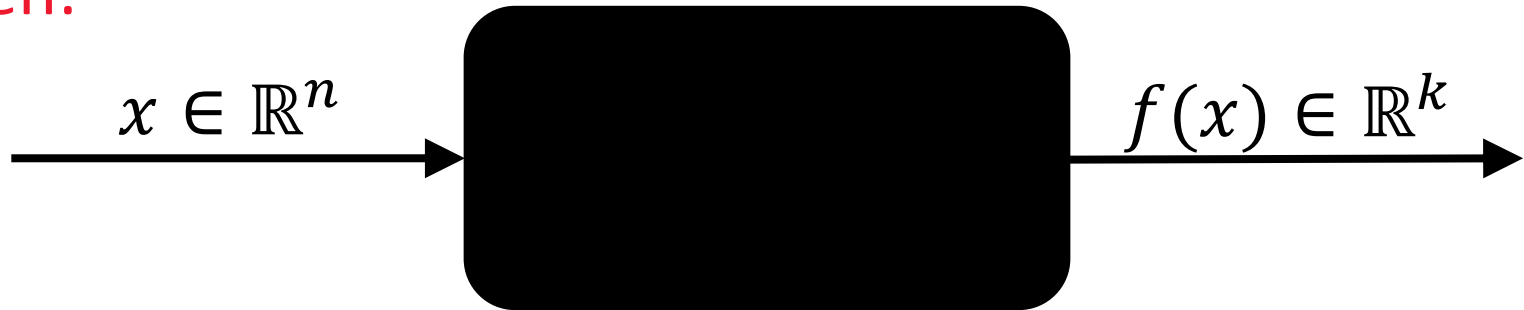
Optimize $f: \Omega \subset \mathbb{R}^n \mapsto \mathbb{R}^k$



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]
- [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]
- [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, ...)

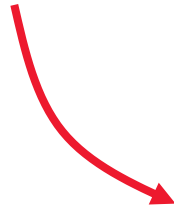
Need: Benchmarking

- 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** and **BBOB** come into play



Comparing Continuous Optimizers Platform

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

automatized benchmarking

How to benchmark algorithms with COCO?

https://github.com/numbbo/coco

numbbo/coco: Numerical ... x +

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

Most Visited Getting Started COCO-Algorithms numbbo/numbbo · Gi... RandOpt CMAP Inria GitLab RER B from lab

This repository Search Pull requests Issues Marketplace Gist

numbbo / coco Unwatch 15 Unstar 38 Fork 24

Code Issues 133 Pull requests 1 Projects 9 Settings Insights

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

Add topics

16,007 commits 11 branches 31 releases 15 contributors

Branch: master New pull request Create new file Upload files Find files Clone or download

brockho committed on GitHub Merge pull request #1352 from numbbo/development Latest commit 4b1497a on 20 Apr

code-experiments	A little more verbose error message when suite regression test fails	a month ago
code-postprocessing	Hashes are back on the plots.	a month ago
code-preprocessing	Fixed preprocessing to work correctly with the extended biobjective s...	3 months ago
howtos	Update create-a-suite-howto.md	4 months ago
.clang-format	raising an error in bbob2009_logger.c when best_value is NULL. Plus s...	2 years ago
.hgignore	raising an error in bbob2009_logger.c when best_value is NULL. Plus s...	2 years ago
AUTHORS	small correction in AUTHORS	a year ago
LICENSE	Update LICENSE	11 months ago

https://github.com/numbbo/coco

numbbo/coco: Numerical ...

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

Most Visited Getting Started COCO-Algorithms numbbo/numbbo · Gi... RandOpt CMAP Inria GitLab RER B from lab

This repository Search Pull requests Issues Marketplace Gist

numbbo / coco Unwatch 15 Unstar 38 Fork 24

Code Issues 133 Pull requests 1 Projects 9 Settings Insights

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

Add topics

16,007 commits 11 branches 31 releases 15 contributors

Branch: master New pull request Create new file Upload files Find file Clone or download

brockho committed on GitHub Merge pull request #1352 from numbbo/development

code-experiments	A little more verbose error message when suite regression test fai	
code-postprocessing	Hashes are back on the plots.	
code-preprocessing	Fixed preprocessing to work correctly with the extended biobjectiv	
howtos	Update create-a-suite-howto.md	
.clang-format	raising an error in bbob2009_logger.c when best_value is NULL. Plus s...	2 years ago
.hgignore	raising an error in bbob2009_logger.c when best_value is NULL. Plus s...	2 years ago
AUTHORS	small correction in AUTHORS	a year ago
LICENSE	Update LICENSE	11 months ago

Clone with HTTPS Use SSH

Use Git or checkout with SVN using the web URL.

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

Open in Desktop Download ZIP 4 months ago

https://github.com/numbbo/coco

numbbo / coco

Unwatch 15 Unstar 38 Fork 24

Code Issues 133 Pull requests 1 Projects 9 Settings Insights

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

Add topics

16,007 commits 11 branches 31 releases 15 contributors

Branch: master New pull request Create new file Upload files Find file Clone or download

brockho committed on GitHub Merge pull request #1352 from numbbo/development

code-experiments	A little more verbose error message when suite regression test fai	
code-postprocessing	Hashes are back on the plots.	
code-preprocessing	Fixed preprocessing to work correctly with the extended biojectiv	
howtos	Update create-a-suite-howto.md	
.clang-format	raising an error in bbob2009_logger.c when best_value is NULL. Plus s...	2 years ago
.hgignore	raising an error in bbob2009_logger.c when best_value is NULL. Plus s...	2 years ago
AUTHORS	small correction in AUTHORS	a year ago
LICENSE	Update LICENSE	11 months ago
README.md	Added link to #1335 before closing.	a month ago

Clone with HTTPS Use SSH

Use Git or checkout with SVN using the web URL.

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

Open in Desktop Download ZIP 4 months ago

https://github.com/numbbo/coco

numbbo/coco: Numerical ...

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

Most Visited Getting Started COCO-Algorithms numbbo/numbbo · Gi... RandOpt CMAP Inria GitLab RER B from lab

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

Add topics

16,007 commits 11 branches 31 releases 15 contributors

Branch: master New pull request Create new file Upload files Find file Clone or download

brockho committed on GitHub Merge pull request #1352 from numbbo/development

code-experiments	A little more verbose error message when suite regression test fai	
code-postprocessing	Hashes are back on the plots.	
code-preprocessing	Fixed preprocessing to work correctly with the extended biobjectiv	
howtos	Update create-a-suite-howto.md	
.clang-format	raising an error in bbob2009_logger.c when best_value is NULL. Plus s...	2 years ago
.hgignore	raising an error in bbob2009_logger.c when best_value is NULL. Plus s...	2 years ago
AUTHORS	small correction in AUTHORS	a year ago
LICENSE	Update LICENSE	11 months ago
README.md	Added link to #1335 before closing.	a month ago
do.py	refactoring here and there in do.py to get closer to PEP8 specifications	2 months ago
doxygen.ini	moved all files into code-experiments/ folder besides the do.py scrip...	2 years ago

README.md

Clone with HTTPS Use SSH

Use Git or checkout with SVN using the web URL.

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

Open in Desktop Download ZIP 4 months ago

https://github.com/numbbo/coco

numbbo/coco: Numerical ...

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

Most Visited Getting Started COCO-Algorithms numbbo/numbbo · Gi... RandOpt CMAP Inria GitLab RER B from lab

Branch: master New pull request

Create new file Upload files Find file Clone or download

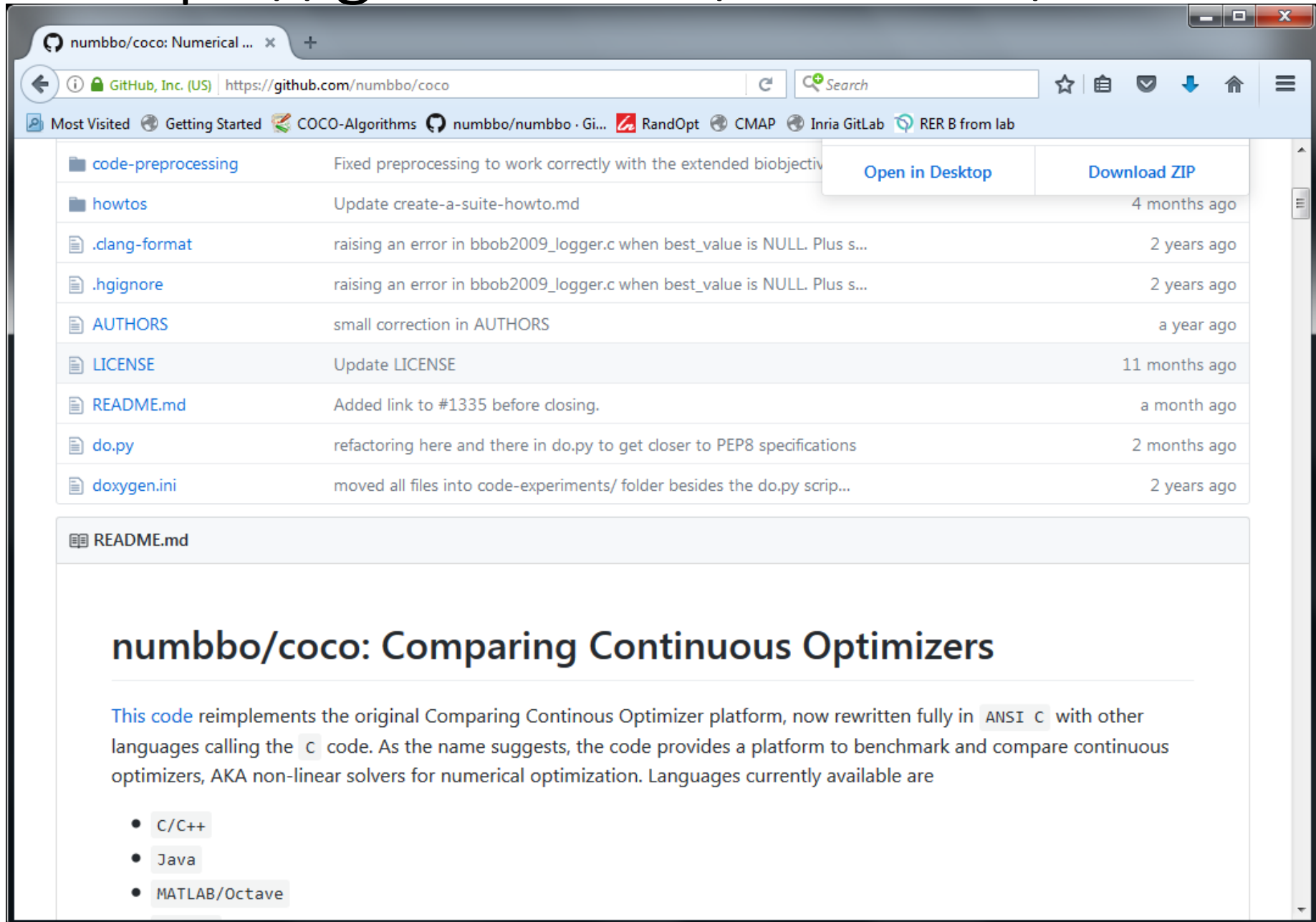
brockho committed on GitHub Merge pull request #1352 from numbbo/development

code-experiments	A little more verbose error message when suite regression test fai	
code-postprocessing	Hashes are back on the plots.	
code-preprocessing	Fixed preprocessing to work correctly with the extended bioobjectiv	
howtos	Update create-a-suite-howto.md	
.clang-format	raising an error in bbob2009_logger.c when best_value is NULL. Plus s...	2 years ago
.hgignore	raising an error in bbob2009_logger.c when best_value is NULL. Plus s...	2 years ago
AUTHORS	small correction in AUTHORS	a year ago
LICENSE	Update LICENSE	11 months ago
README.md	Added link to #1335 before closing.	a month ago
do.py	refactoring here and there in do.py to get closer to PEP8 specifications	2 months ago
doxygen.ini	moved all files into code-experiments/ folder besides the do.py scrip...	2 years ago

Open in Desktop **Download ZIP** 4 months ago

numbbo/coco: Comparing Continuous Optimizers

https://github.com/numbbo/coco



numbbo/coco: Numerical ... x +

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

Most Visited Getting Started COCO-Algorithms numbbo/numbbo · Gi... RandOpt CMAP Inria GitLab RER B from lab

code-preprocessing	Fixed preprocessing to work correctly with the extended biojectiv	Open in Desktop	Download ZIP
howtos	Update create-a-suite-howto.md		4 months ago
.clang-format	raising an error in bbob2009_logger.c when best_value is NULL. Plus s...		2 years ago
.hgignore	raising an error in bbob2009_logger.c when best_value is NULL. Plus s...		2 years ago
AUTHORS	small correction in AUTHORS		a year ago
LICENSE	Update LICENSE		11 months ago
README.md	Added link to #1335 before closing.		a month ago
do.py	refactoring here and there in do.py to get closer to PEP8 specifications		2 months ago
doxygen.ini	moved all files into code-experiments/ folder besides the do.py scrip...		2 years 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

https://github.com/numbbo/coco

The screenshot shows a web browser window with the URL `https://github.com/numbbo/coco`. The browser's address bar and tabs are visible at the top. Below the browser, the GitHub repository page is displayed. It features a list of recent commits on the left side, followed by the README content.

File	Commit Message	Time Ago
LICENSE	Update LICENSE	11 months ago
README.md	Added link to #1335 before closing.	a month ago
do.py	refactoring here and there in do.py to get closer to PEP8 specifications	2 months ago
doxygen.ini	moved all files into code-experiments/ folder besides the do.py scrip...	2 years ago

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

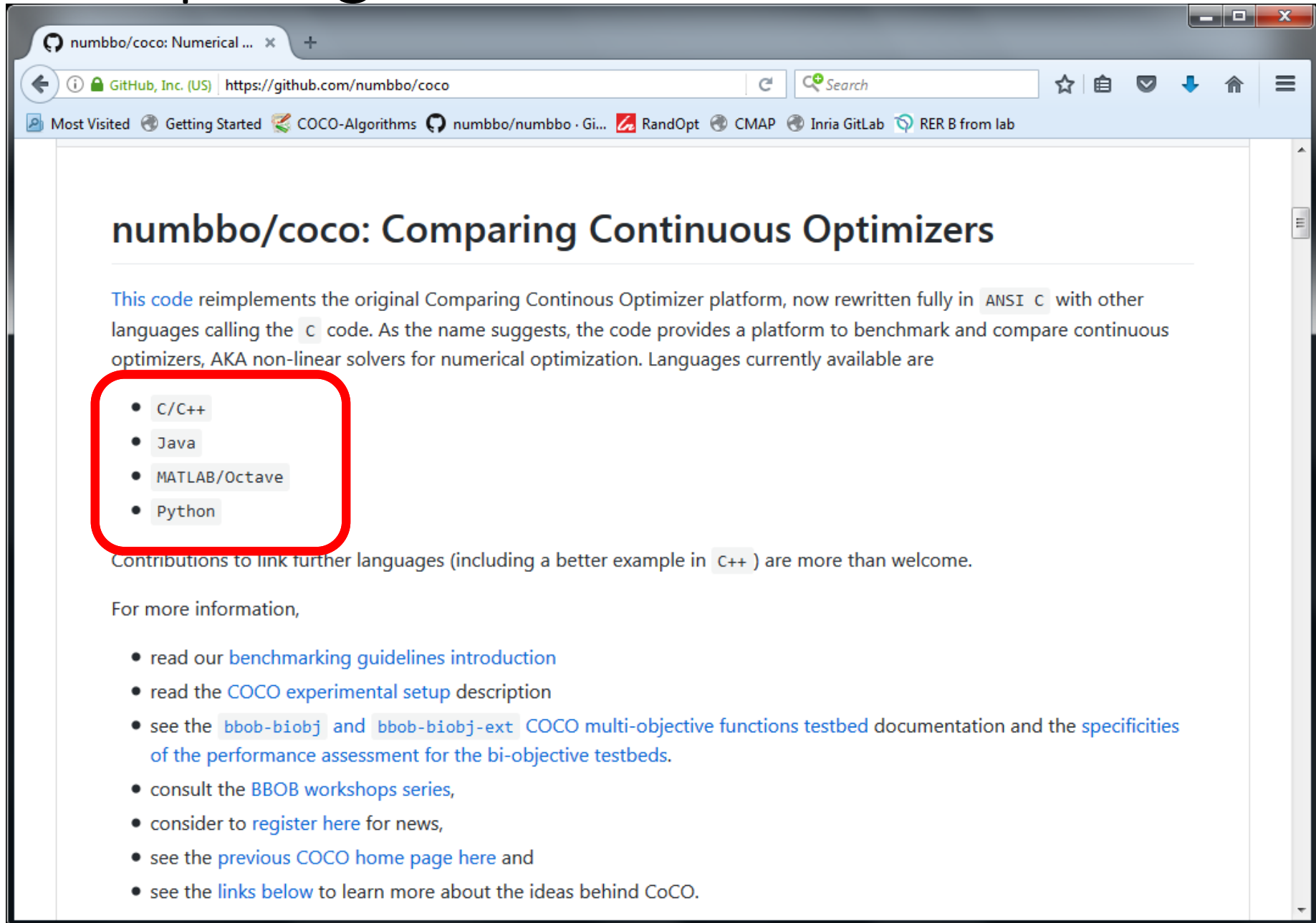
- C/C++
- Java
- MATLAB/Octave
- Python

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

For more information,

- read our [benchmarking guidelines introduction](#)
- read the [COCO experimental setup](#) description

https://github.com/numbbo/coco



numbbo/coco: Numerical ... x +

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

Most Visited Getting Started COCO-Algorithms numbbo/numbbo · Gi... RandOpt CMAP Inria GitLab RER B from lab

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,

- read our [benchmarking guidelines introduction](#)
- read the [COCO experimental setup description](#)
- see the [bbob-biobj](#) and [bbob-biobj-ext](#) [COCO multi-objective functions testbed](#) documentation and the [specificities of the performance assessment for the bi-objective testbeds](#).
- 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.

https://github.com/numbbo/coco

numbbo/coco: Numerical ... x +

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

Most Visited Getting Started COCO-Algorithms numbbo/numbbo · Gi... RandOpt CMAP Inria GitLab RER B from lab

Getting Started

0. Check out the [Requirements](#) above.

1. Download the COCO framework code from github,

- either by clicking the [Download ZIP button](#) and unzip the `zip` file,
- or 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.

The record of official releases can be found [here](#). The latest release corresponds to the [master branch](#) as linked above.

2. 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.

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

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

https://github.com/numbbo/coco

numbbo/coco: Numerical ... x +

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

Most Visited Getting Started COCO-Algorithms numbbo/numbbo · Gi... RandOpt CMAP Inria GitLab RER B from lab

Getting Started

0. Check out the [Requirements](#) above.
1. Download the COCO framework code from github,
 - either by clicking the [Download ZIP button](#) and unzip the `zip` file,
 - or 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.

The record of official releases can be found [here](#). The latest release corresponds to the [master branch](#) as linked above.

2. 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:

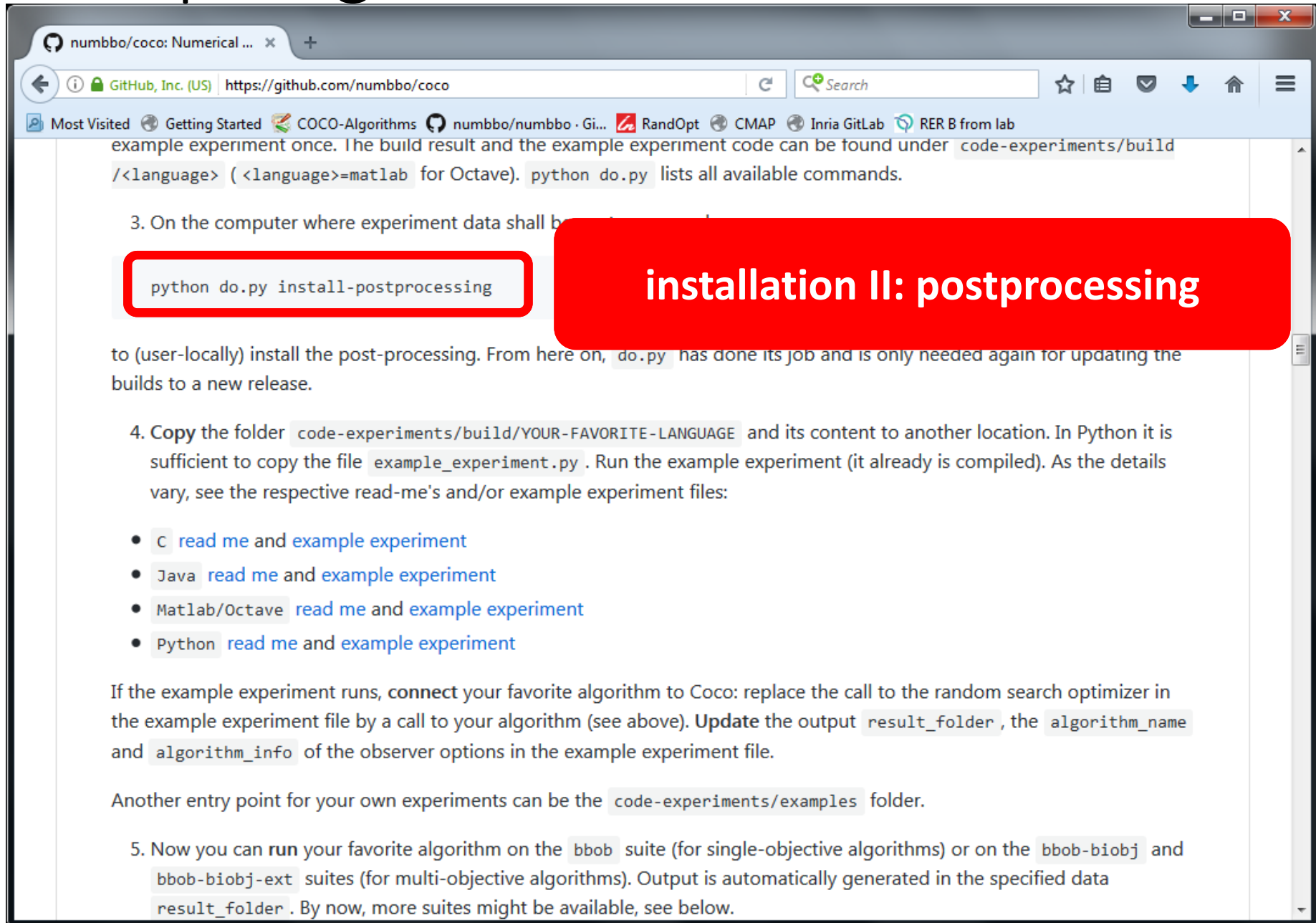
```
python do.py run-c
python do.py run-cuda
python do.py run-matlab
python do.py run-octave
python do.py run-python
```

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

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

installation I: experiments

https://github.com/numbbo/coco



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.

3. On the computer where experiment data shall be stored, run the following command:

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

(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.

4. 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). As the details vary, see the respective read-me's and/or example experiment files:

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

5. Now you can run your favorite algorithm on the `bbob` suite (for single-objective algorithms) or on the `bbob-biobj` and `bbob-biobj-ext` suites (for multi-objective algorithms). Output is automatically generated in the specified data `result_folder`. By now, more suites might be available, see below.

https://github.com/numbbo/coco



The image shows a browser window displaying the GitHub repository page for numbbbo/coco. The page content includes instructions for running experiments. A red rounded rectangle highlights a list of links for different languages: C, Java, Matlab/Octave, and Python. To the right of this list is a red rounded rectangle containing the text 'coupling algo + COCO'. The browser's address bar shows the URL https://github.com/numbbbo/coco. The page content includes a code block for 'python do.py install-postprocessing' and several numbered steps explaining how to use the tool.

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.

3. 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.

4. 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). As the details vary, see the respective read-me's and/or example experiment files:

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

5. Now you can run your favorite algorithm on the `bbob` suite (for single-objective algorithms) or on the `bbob-biobj` and `bbob-biobj-ext` suites (for multi-objective algorithms). Output is automatically generated in the specified data `result_folder`. By now, more suites might be available, see below.

coupling algo + COCO

example_experiment.c (slightly simplified)

```
/* 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 at develop... x +

GitHub, Inc. (US) | https://github.com/numbbo/coco/tree/development

Most Visited Getting Started COCO-Algorithms numbbo/numbbo · Gi... RandOpt CMAP Inria GitLab RER B from lab

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

5. Now you can run your favorite algorithm on the `bbob` suite (for single-objective algorithms) or on the `bbob-biobj` and `bbob-biobj-ext` suites (for multi-objective algorithms). Output is automatically generated in the specified data `result_folder`. By now, more suites might be available, see below.

6. Postprocess the data from the results folder by typing

```
python -m cocopp [-o OUTPUT_FOLDERNAME] YOURDATA
```

Any subfolder in the folder arguments will be searched for... on 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 an `index.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.

A summary pdf can be produced via LaTeX. The corresponding templates can be found in the `code-postprocessing/latex-templates` folder. Basic html output is also available in the result folder of the postprocessing (file `templateBBOBarticle.html`).

7. 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.

8. The experiments can be **parallelized** with any re-distribution of single problem instances to batches (see `example_experiment.py` for an example). Each batch must write in a different target folder (this should happen automatically). Results of each batch must be kept under their separate folder as is. These folders then must be

running the experiment

https://github.com/numbbo/coco

numbbo/coco at develop...

GitHub, Inc. (US) | https://github.com/numbbo/coco/tree/development

Most Visited Getting Started COCO-Algorithms numbbo/numbbo · Gi... RandOpt CMAP Inria GitLab RER B from lab

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

5. Now you can run your favorite algorithm on the `bbob` suite (for single-objective algorithms) or on the `bbob-biobj` and `bbob-biobj-ext` suites (for multi-objective algorithms). Output is automatically generated in the specified data `result_folder`. By now, more suites might be available, see below.

6. Postprocess the data from the results folder by typing

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

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` specifying several data result folders generated by different algorithms.

A folder, `ppdata` by default, will be generated, which contains a `ppdata` file, useful as main entry point to explore the result with a browser. Specify the output folder name with the `-o OUTPUT_FOLDERNAME` option.

A summary pdf can be produced via LaTeX. The corresponding templates can be found in the `code-postprocessing/latex-templates` folder. Basic html output is also available in the result folder of the postprocessing (file `templateBBOBarticle.html`).

7. 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.

8. The experiments can be **parallelized** with any re-distribution of single problem instances to batches (see `example_experiment.py` for an example). Each batch must write in a different target folder (this should happen automatically). Results of each batch must be kept under their separate folder as is. These folders then must be

postprocessing

https://github.com/numbbo/coco

numbbo/coco at develop...

GitHub, Inc. (US) | https://github.com/numbbo/coco/tree/development

Most Visited Getting Started COCO-Algorithms numbbo/numbbo · Gi... RandOpt CMAP Inria GitLab RER B from lab

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

5. Now you can run your favorite algorithm on the `bbob` suite (for single-objective algorithms) or on the `bbob-biobj` and `bbob-biobj-ext` suites (for multi-objective algorithms). Output is automatically generated in the specified data `result_folder`. By now, more suites might be available, see below.

6. Postprocess the data from the results folder by typing

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

Any subfolder in the folder arguments will be searched for logged data. That is, experiments from different batches can be in different folders. You can specify multiple folders by specifying

A folder, file, useful for the output

A summary templates template

7. Once independent

8. The experiments can be parallelized with any re-distribution of single problem instances to batches (see `example_experiment.py` for an example). Each batch must write in a different target folder (this should happen automatically). Results of each batch must be kept under their separate folder as is. These folders then must be

new feature: data archive

```
python -m cocopp BFGS! BIPOP! lmm*
```

Result Folder

The screenshot shows a Windows File Explorer window with the following details:

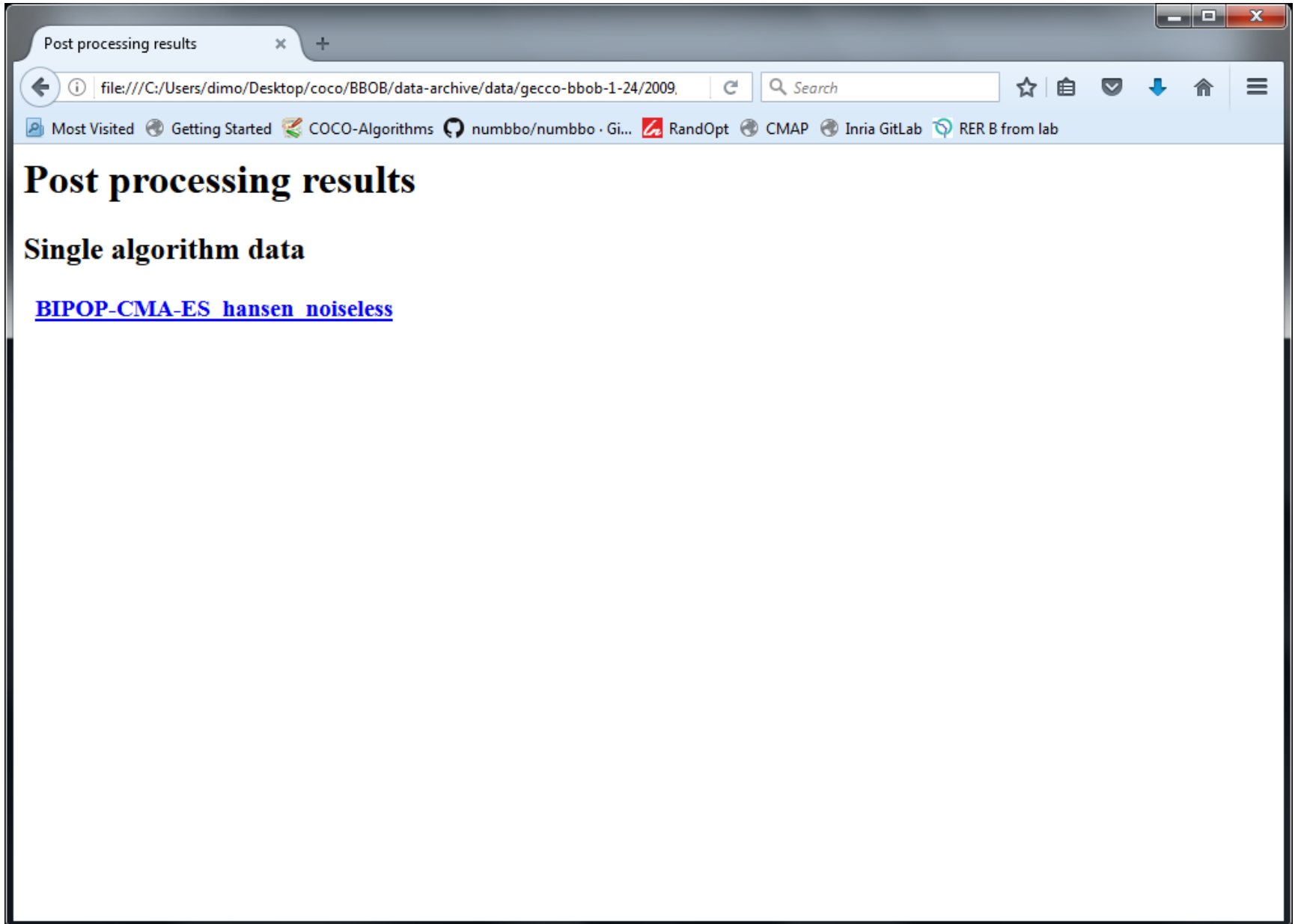
- Address Bar:** << data-archive > data > gecco-bbob-1-24 > 2009 > rawdata > ppdata > Search ppdata
- Navigation:** Organize, Include in library, Share with, New folder
- Left Pane (Navigation Pane):** Favorites (Downloads, Dropbox, Recent Places, Desktop, IntelGraphicsProfiles), Libraries (Documents, Git, Music, Pictures, Subversion, Videos), Homegroup, Computer (Windows (C:))
- Main Pane (Table):**

Name	Date modified	Type	Size
BIPOP-CMA-ES_hansen_noiseless	03/06/2017 11:33	File folder	
cocopp_commands.tex	03/06/2017 11:33	LaTeX Document	7 KB
index.html	03/06/2017 11:33	Firefox HTML Doc...	1 KB
ppdata.html	03/06/2017 11:33	Firefox HTML Doc...	1 KB

4 items State: Shared

Select a file to preview.

Automatically Generated Results



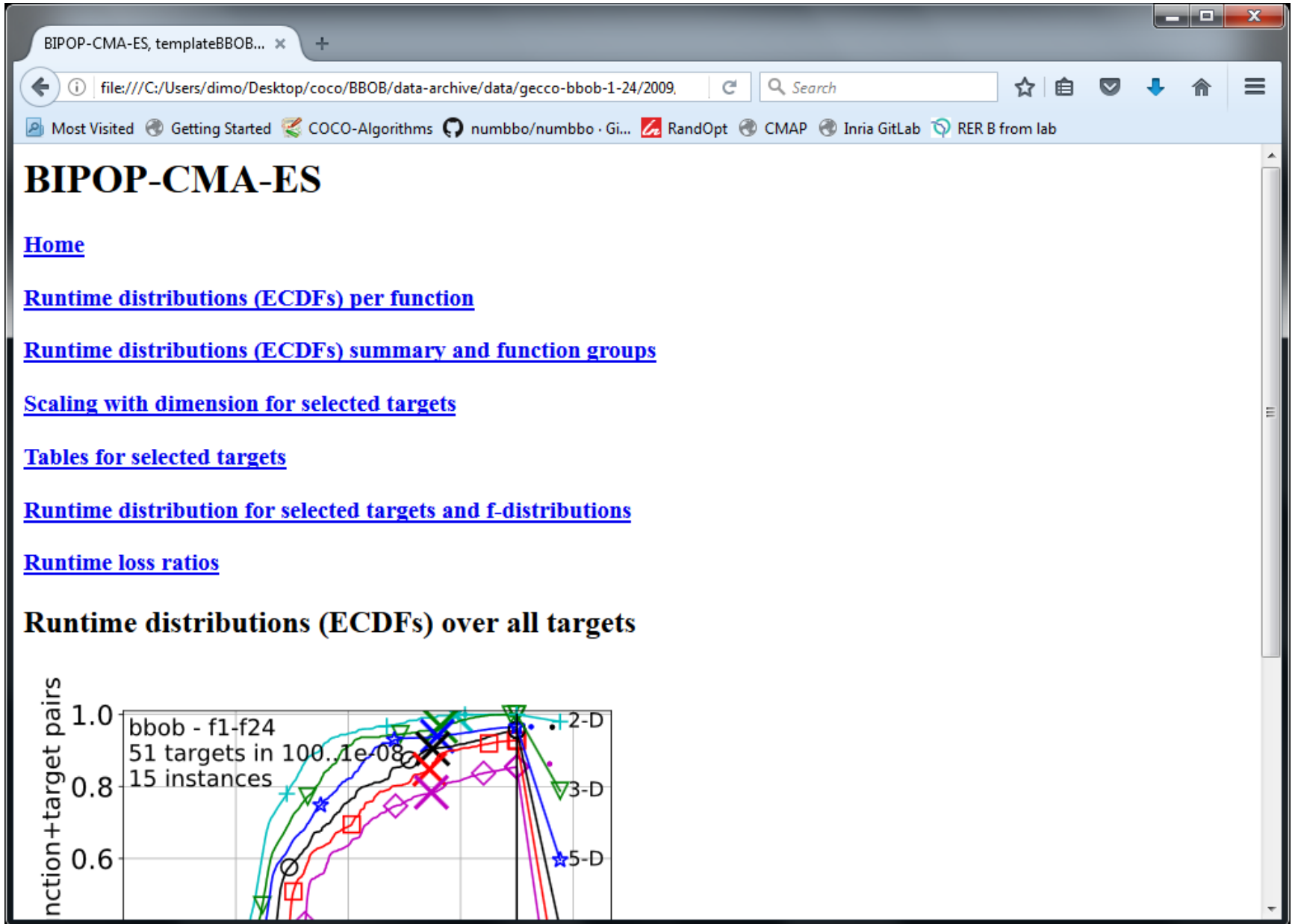
The image shows a web browser window with a single tab titled "Post processing results". The address bar contains the file path: `file:///C:/Users/dimo/Desktop/coco/BBOB/data-archive/data/gecco-bbob-1-24/2009`. The browser's toolbar includes a search bar and several navigation icons. Below the toolbar, a bookmarks bar lists several sites: "Most Visited", "Getting Started", "COCO-Algorithms", "numbbo/numbbo · Gi...", "RandOpt", "CMAP", "Inria GitLab", and "RER B from lab". The main content area of the browser displays the following text:

Post processing results

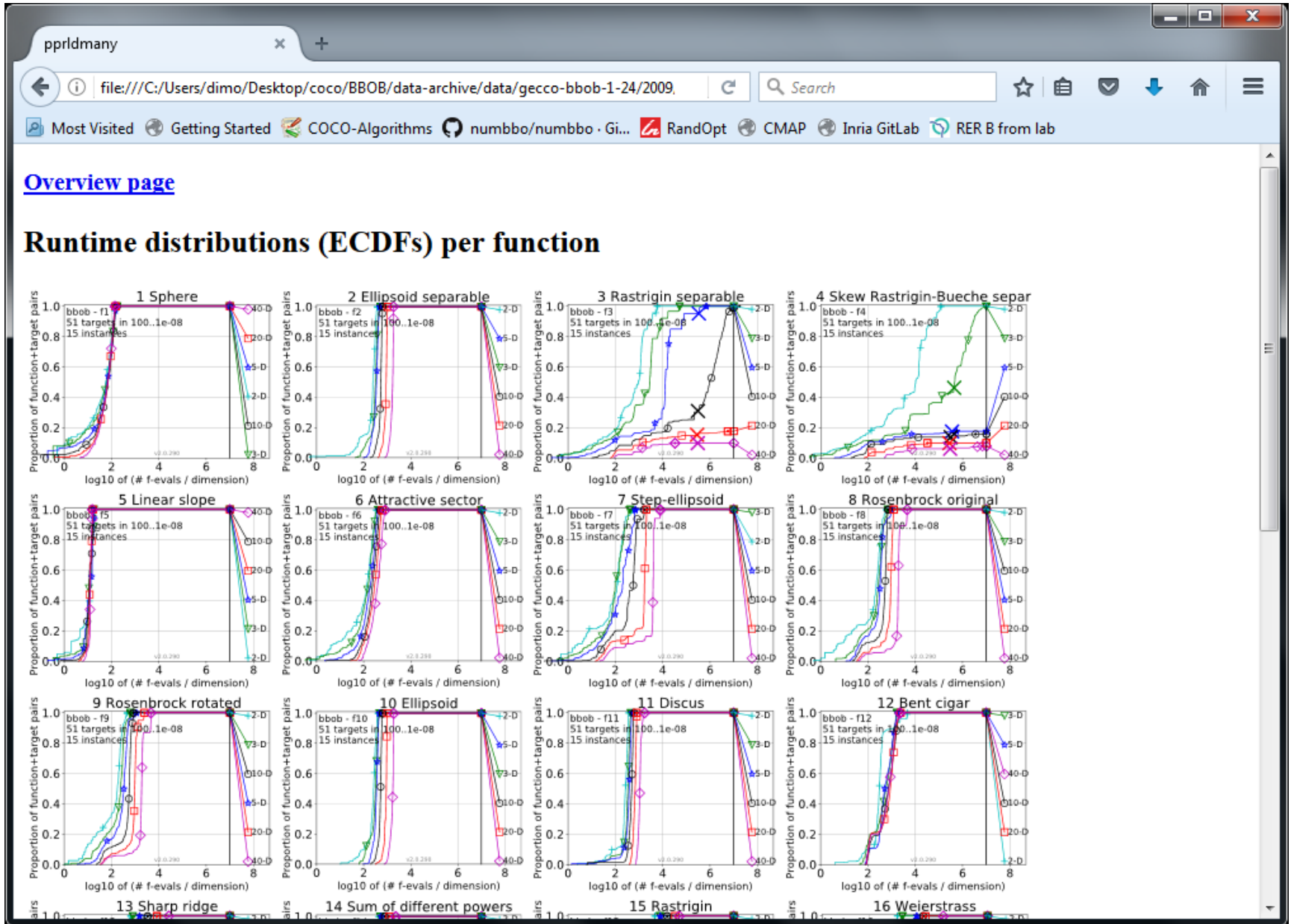
Single algorithm data

[BIPOP-CMA-ES hansen noiseless](#)

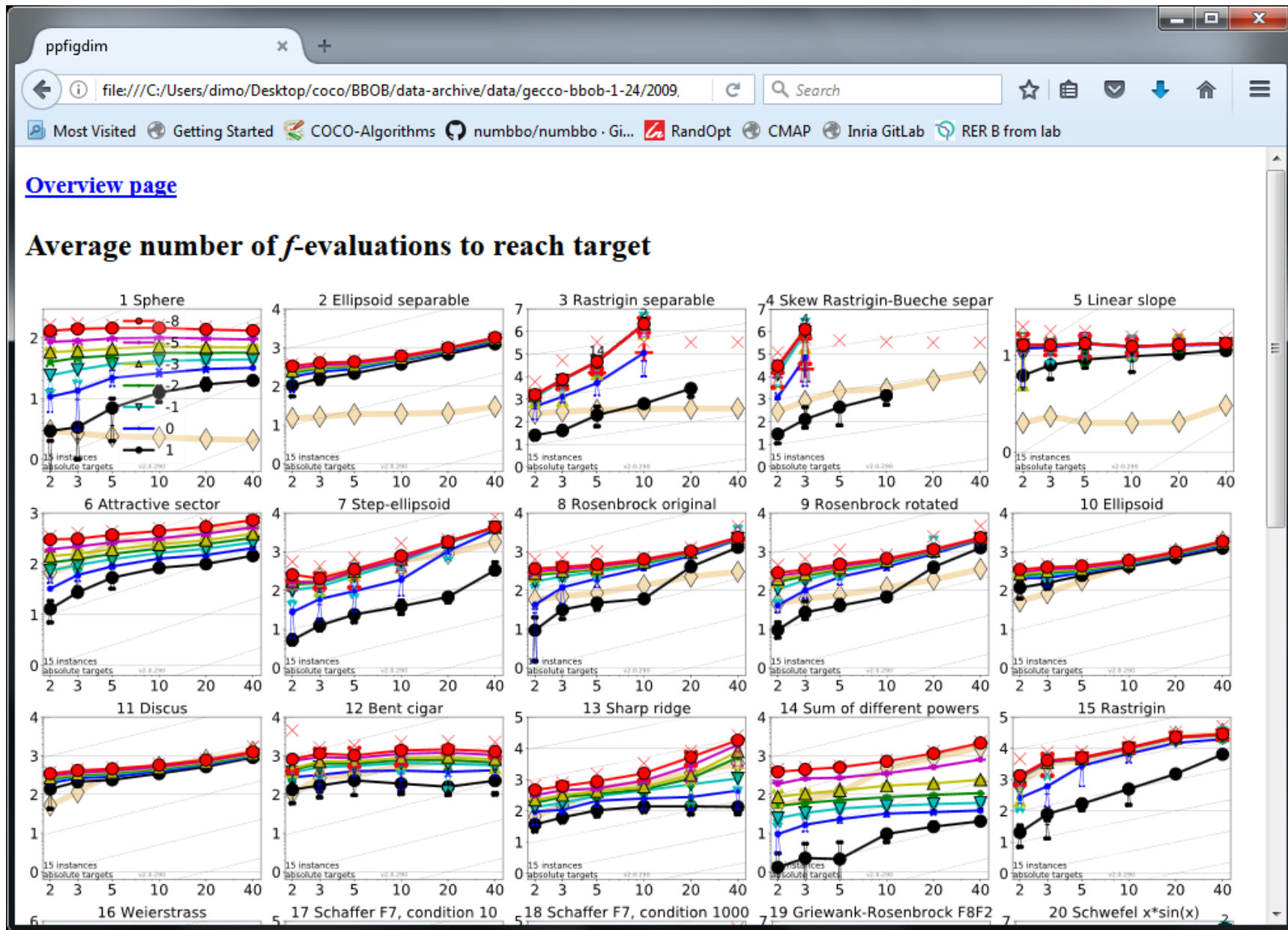
Automatically Generated Results



Automatically Generated Results



Automatically Generated Results



doesn't look too complicated, does it?

[the devil is in the details 😊]

so far (i.e. incl. BBOB-2018):

data for 200+ algorithm variants

(some of which on noisy or multiobjective test functions)

136 workshop papers

by 101 authors from 28 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

- account for **invariance properties**

prediction of performance is based on "similarity",
ideally equivalence classes of functions

Available Test Suites in COCO

- | | | |
|--------------|----------------------|---------------------|
| • bbob | 24 noiseless fcts | 180+ algo data sets |
| • bbob-noisy | 30 noisy fcts | 40+ algo data sets |
| • bbob-biobj | 55 bi-objective fcts | 16 algo data sets |

Almost finished:

- extended bi-objective suite (bbob-biobj-ext)
- large-scale version of bbob (bbob-largescale)

Under development/in planning phase:

- constrained test suite (bbob-constrained)
- mixed-integer suite
- real-world problems
(see also the game benchmarking workshop)

How Do We Measure Performance?

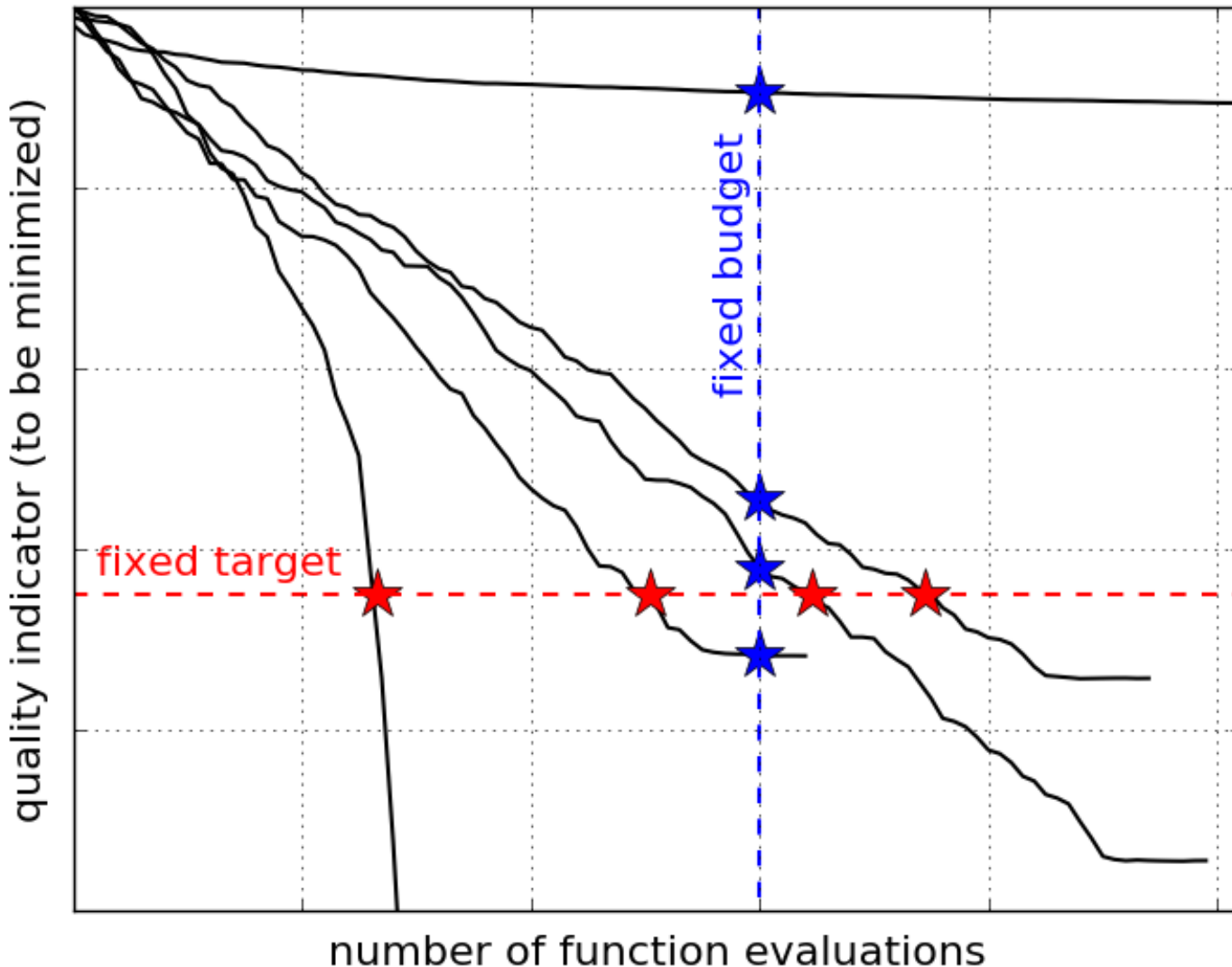
Meaningful quantitative measure

- **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)

Measuring Performance Empirically

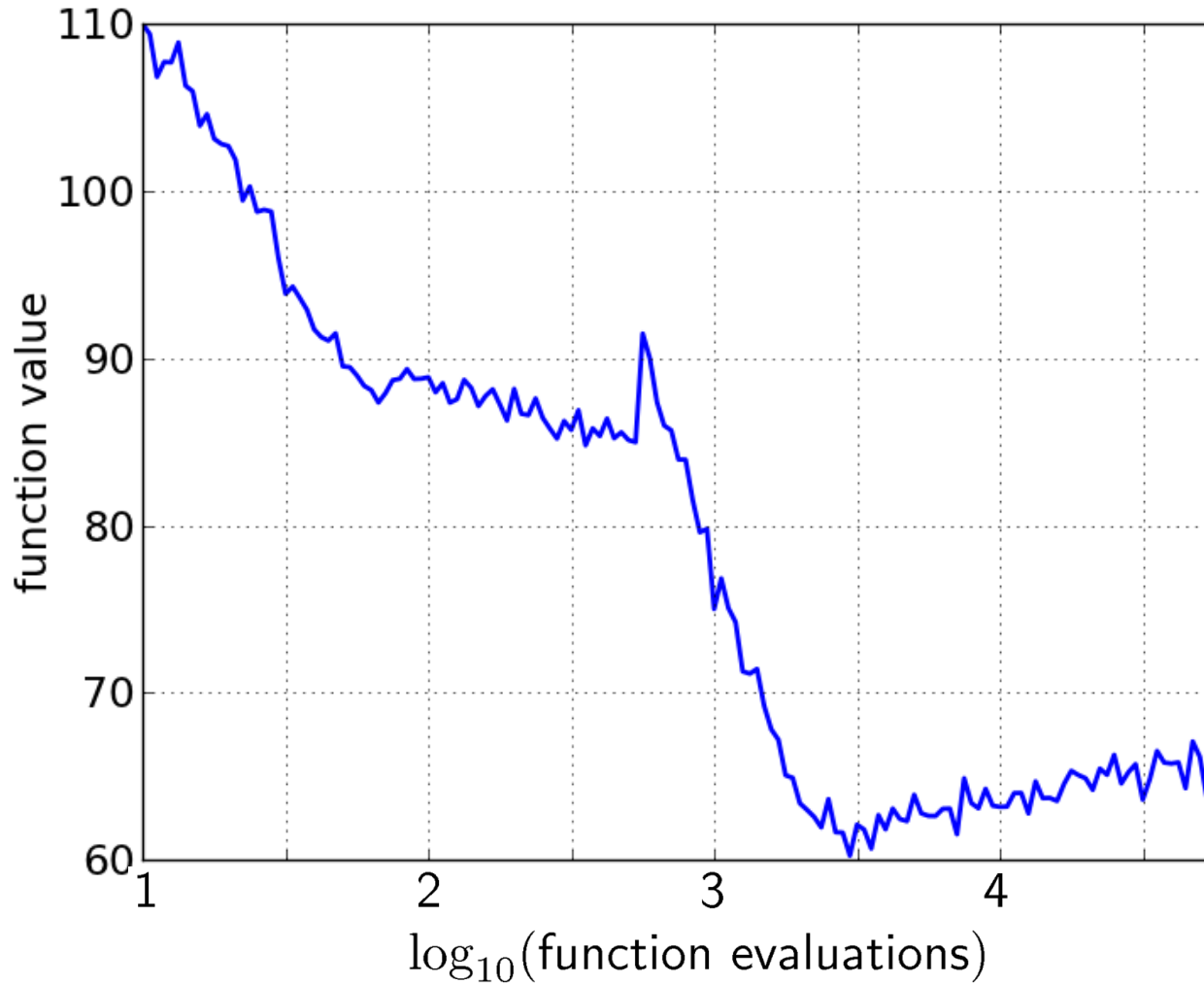
convergence graphs is all we have to start with...



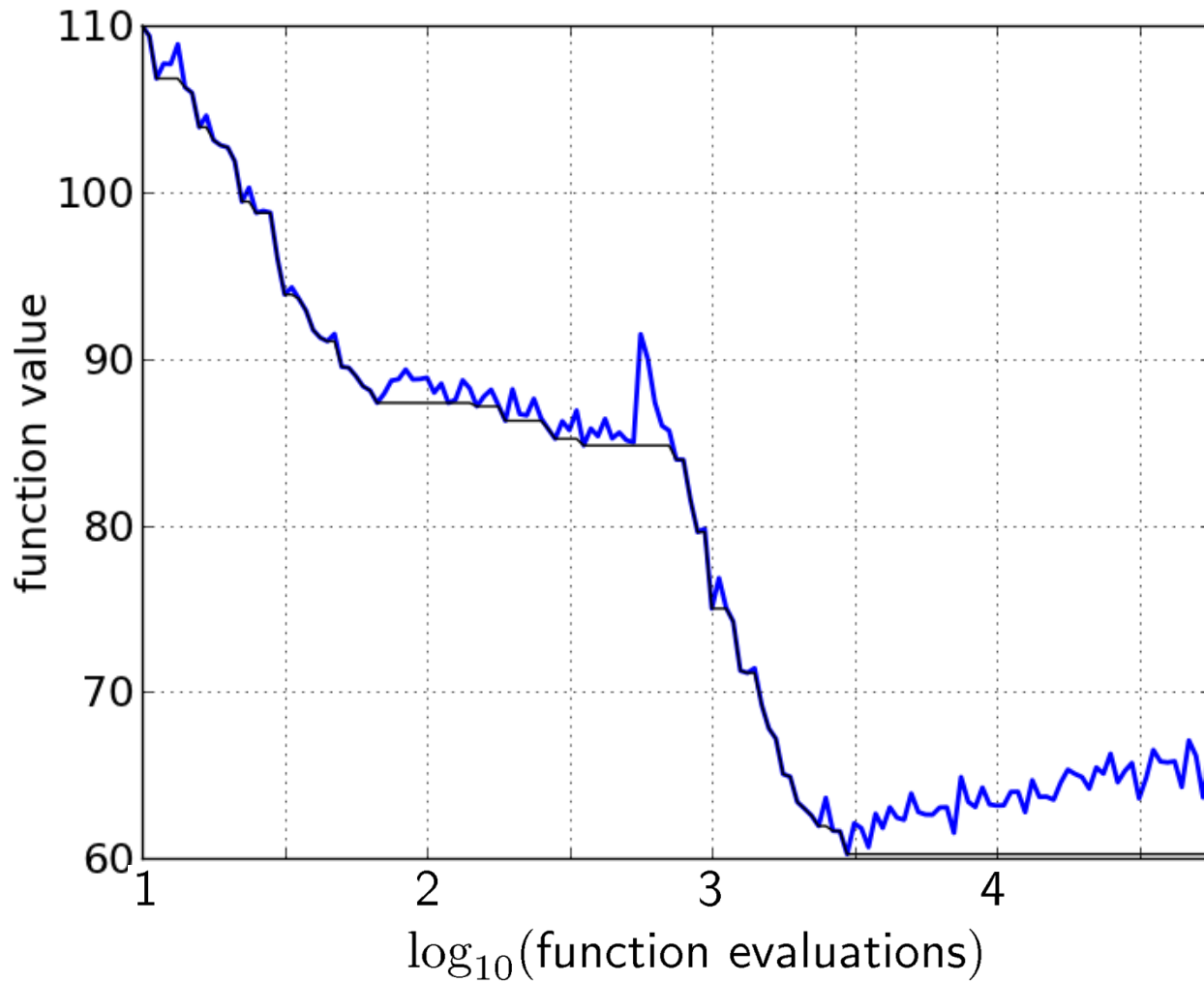
ECDF:

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

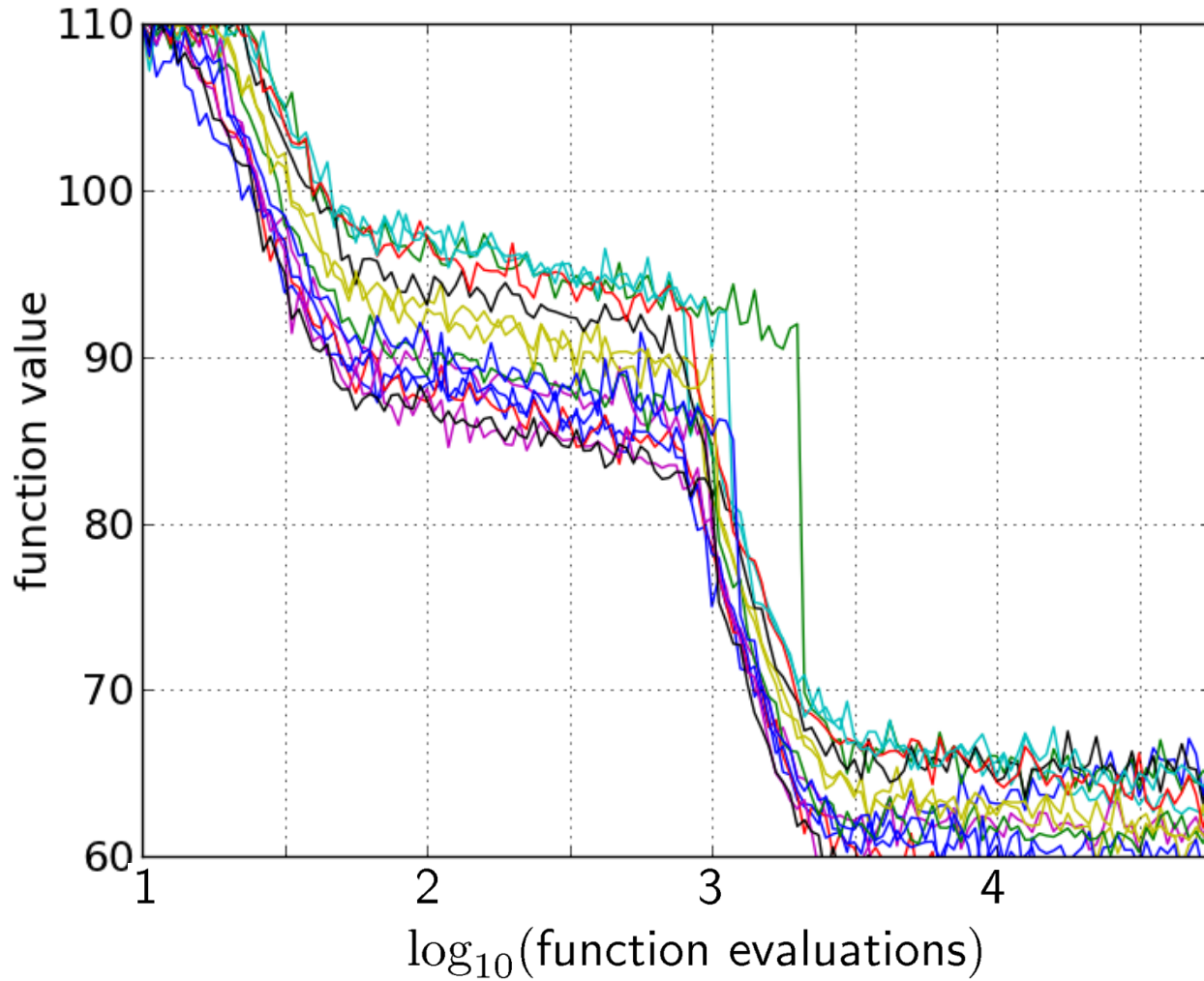
A Convergence Graph



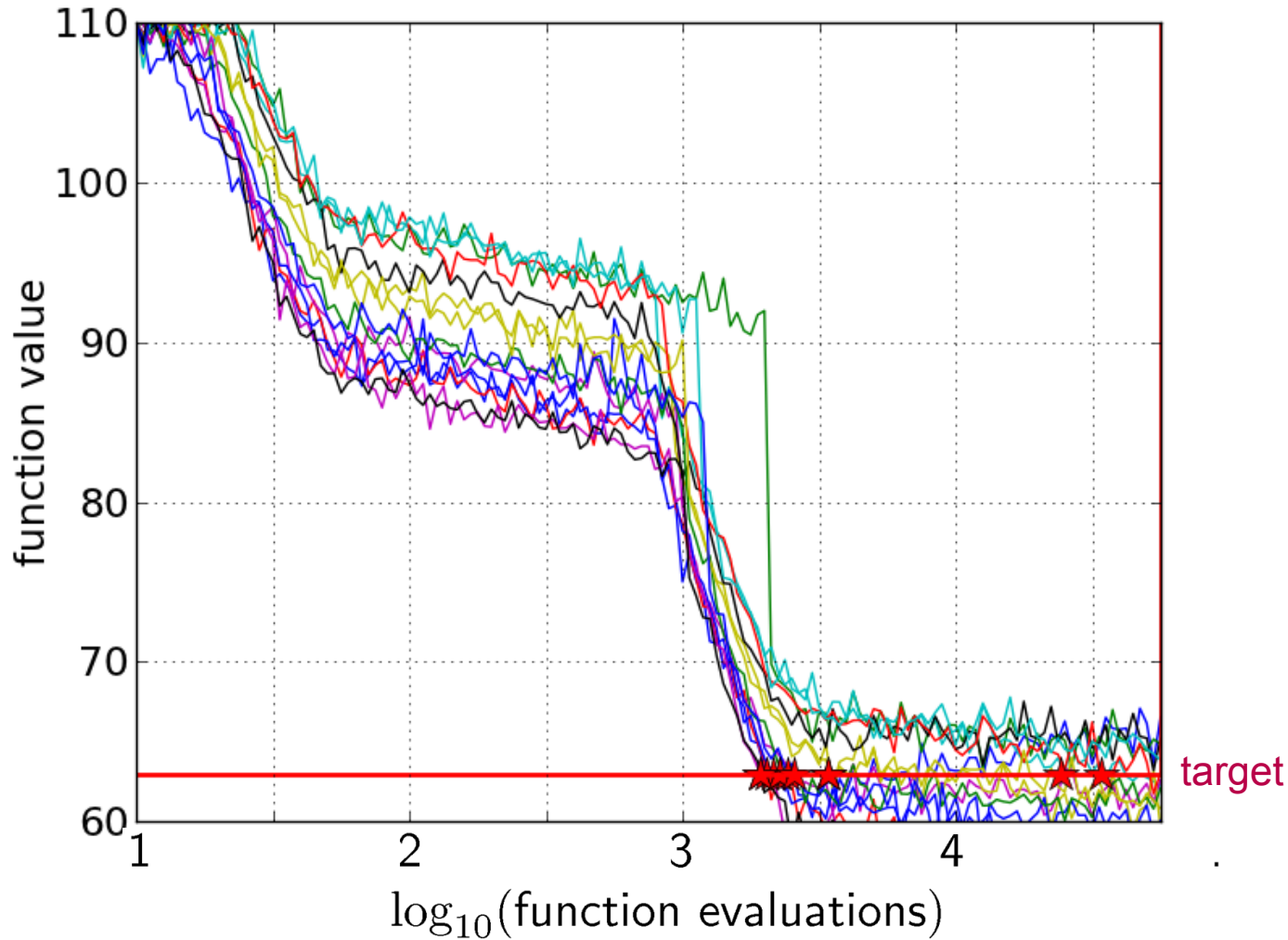
First Hitting Time is Monotonous



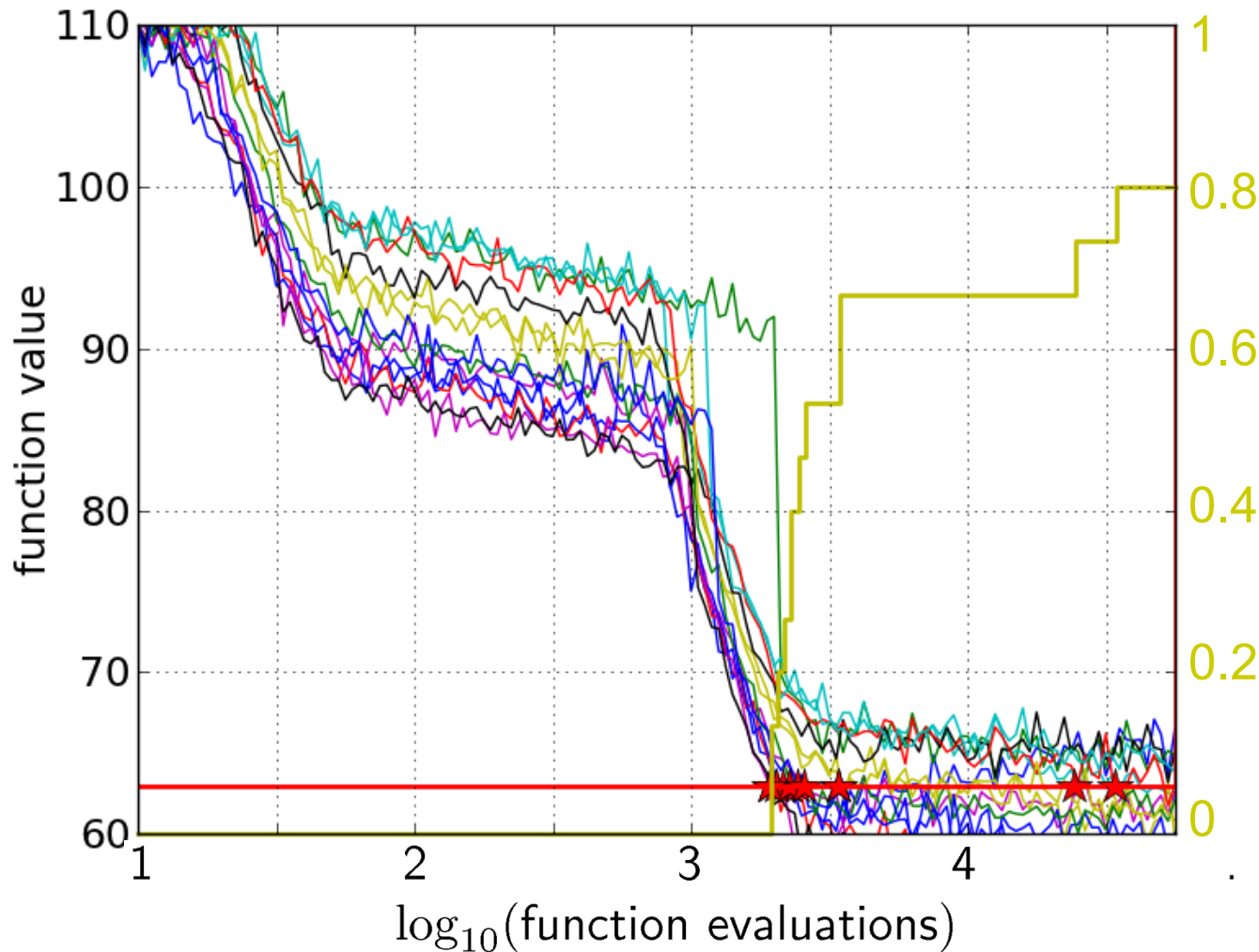
15 Runs



15 Runs \leq 15 Runtime Data Points



Empirical Cumulative Distribution

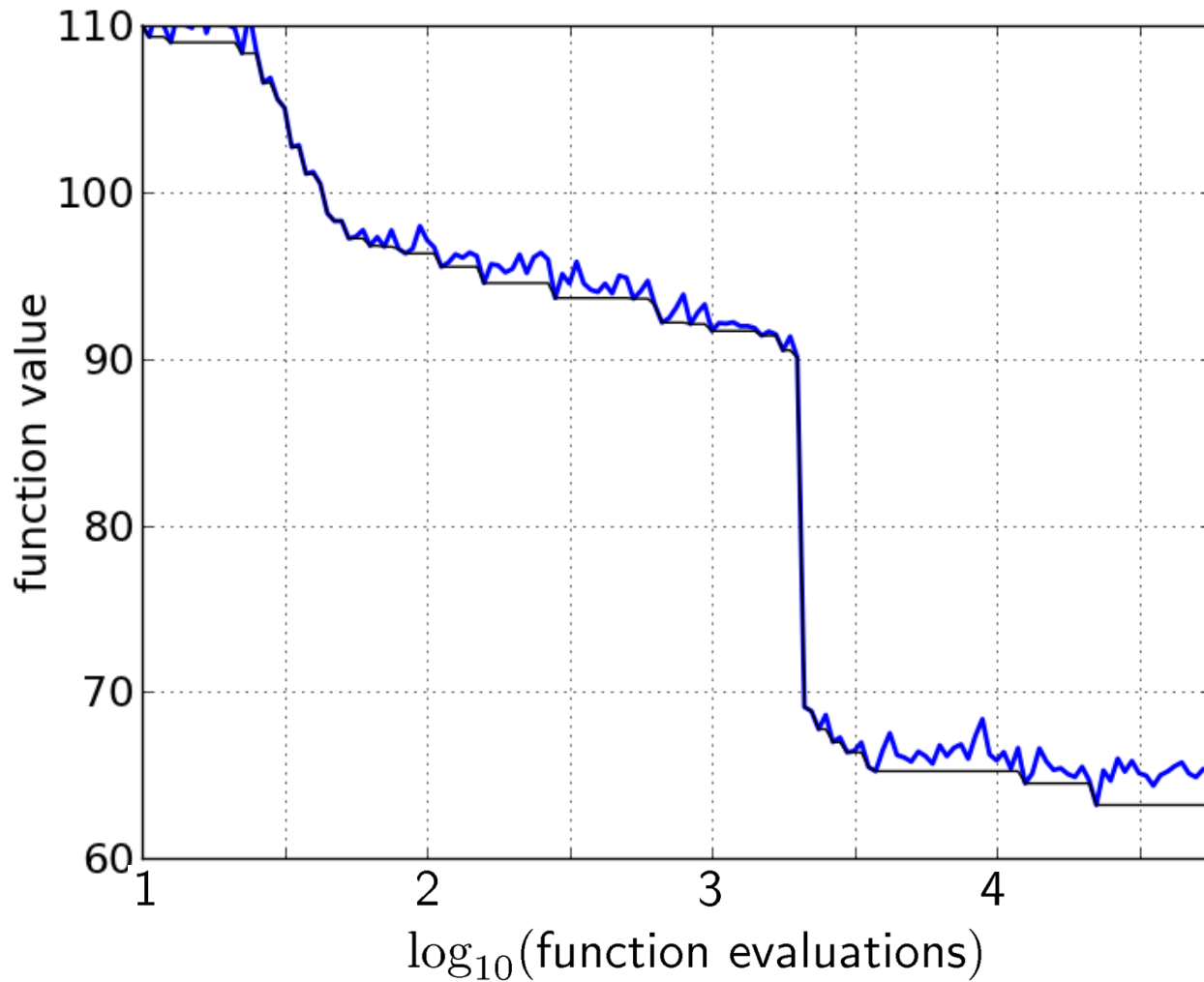


- 1 the **ECDF** of run lengths to reach the target
- has for each data point a **vertical step of constant size**
- displays for each x-value (budget) the count of observations to the left (first hitting times)

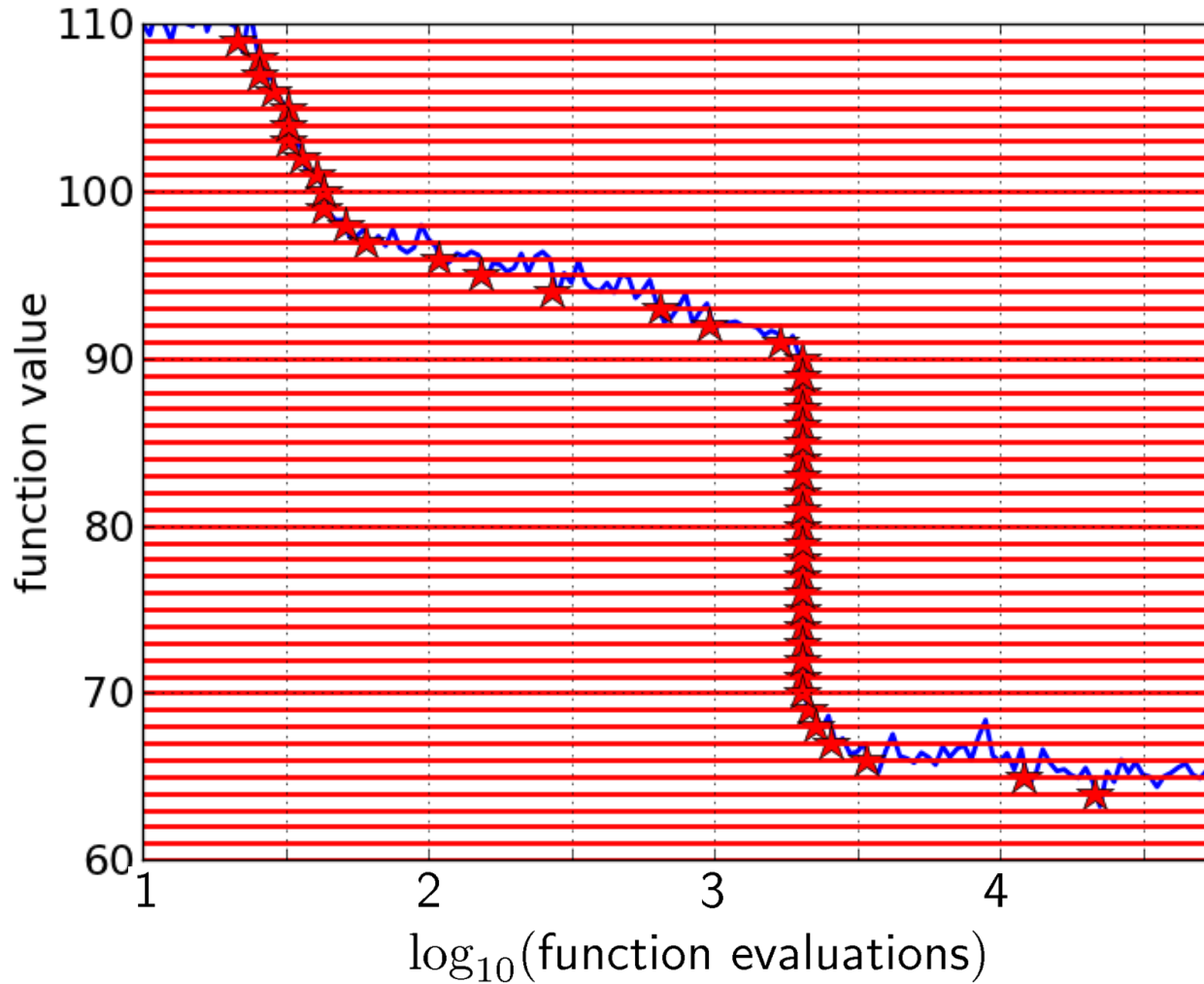
e.g. 60% of the runs need between 2000 and 4000 evaluations

80% of the runs reached the target

Reconstructing A Single Run

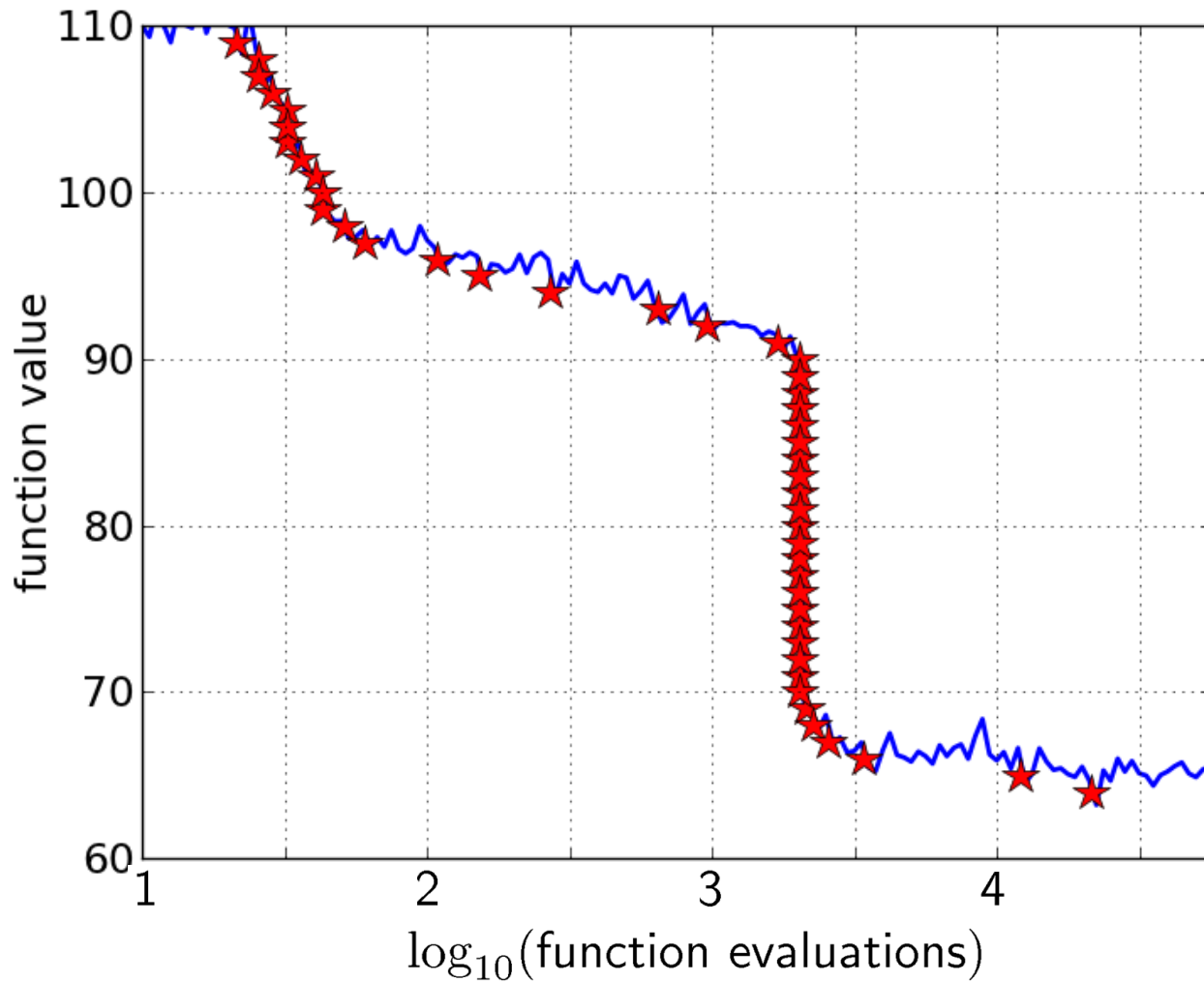


Reconstructing A Single Run

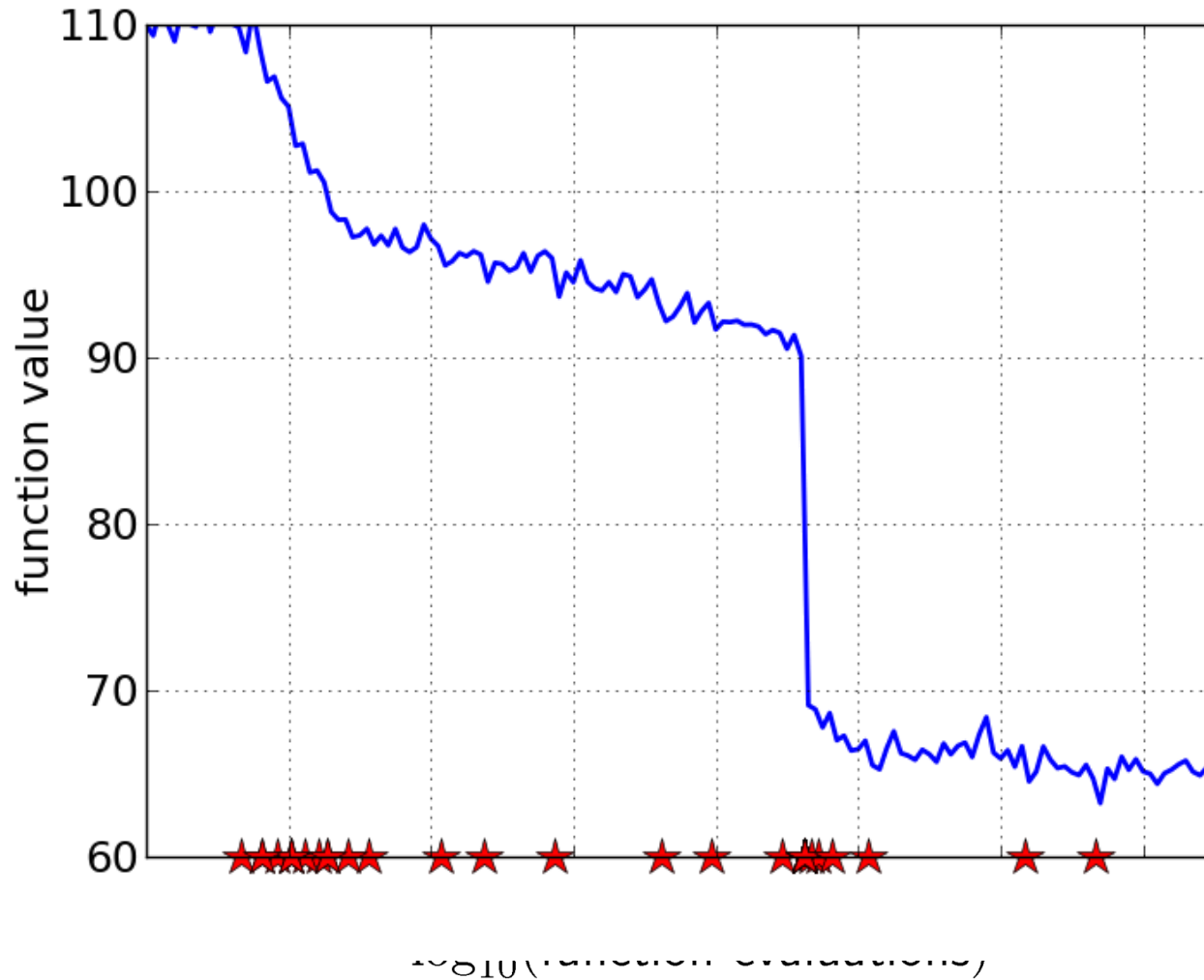


50 equally spaced targets

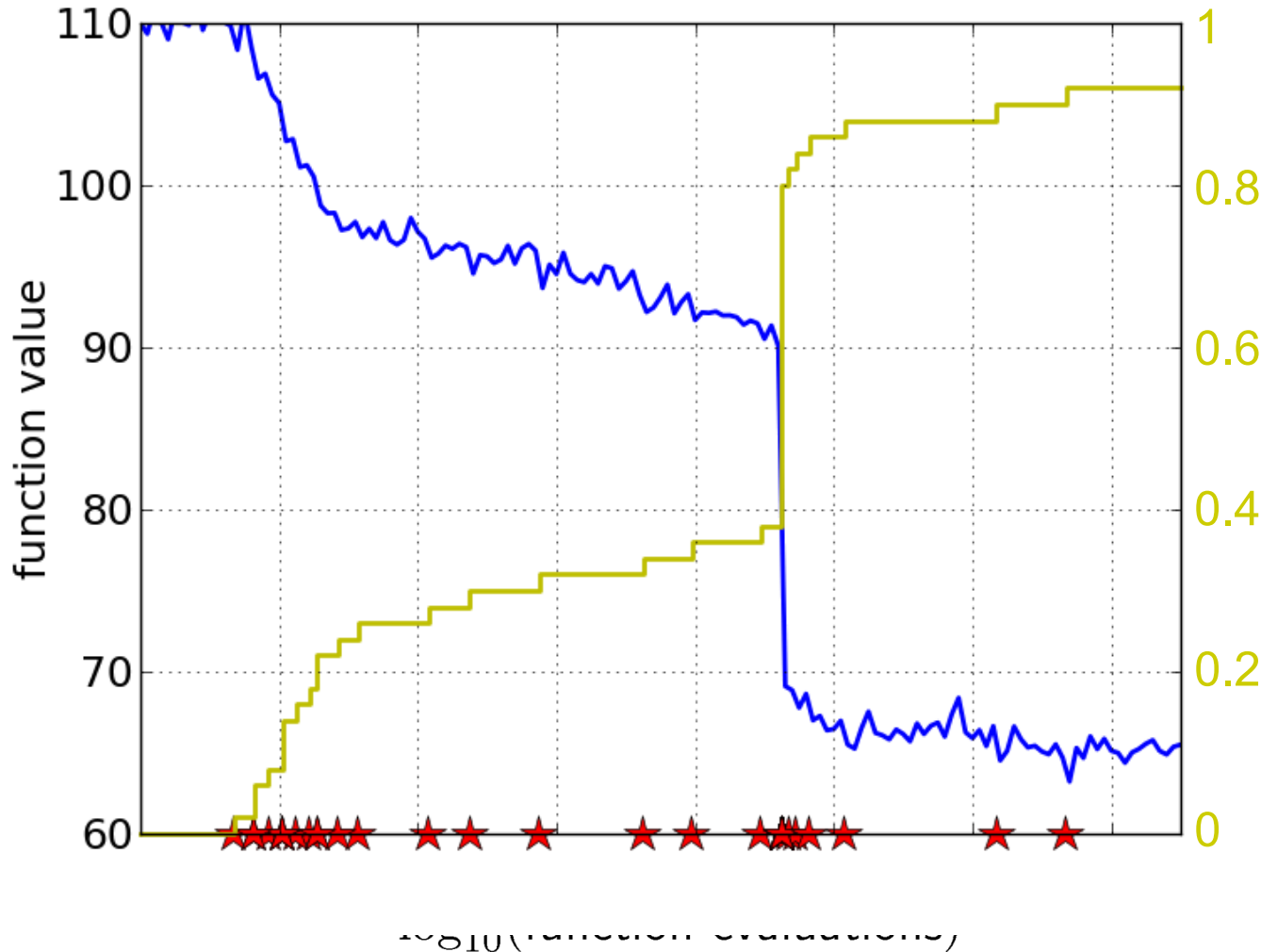
Reconstructing A Single Run



Reconstructing A Single Run

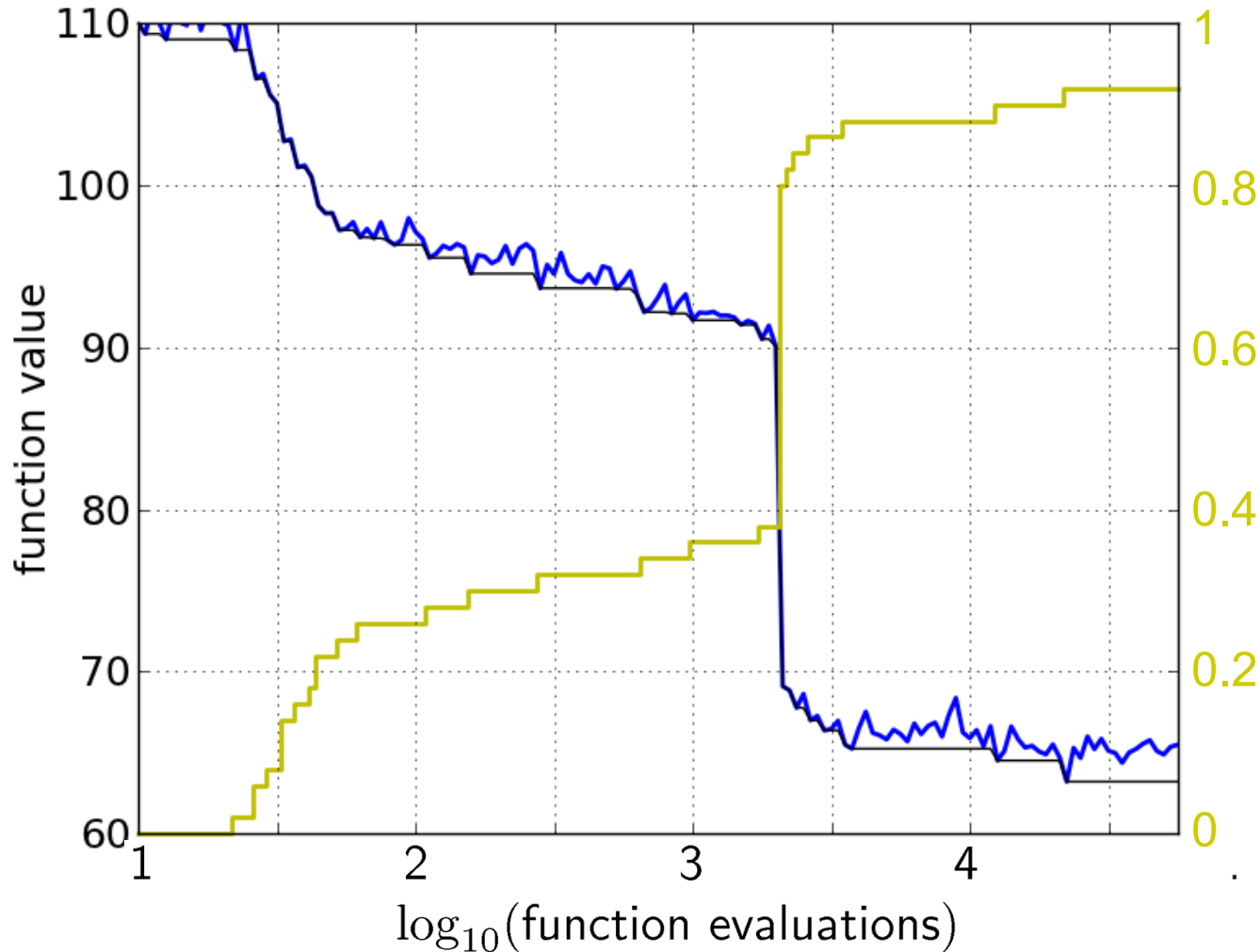


Reconstructing A Single Run



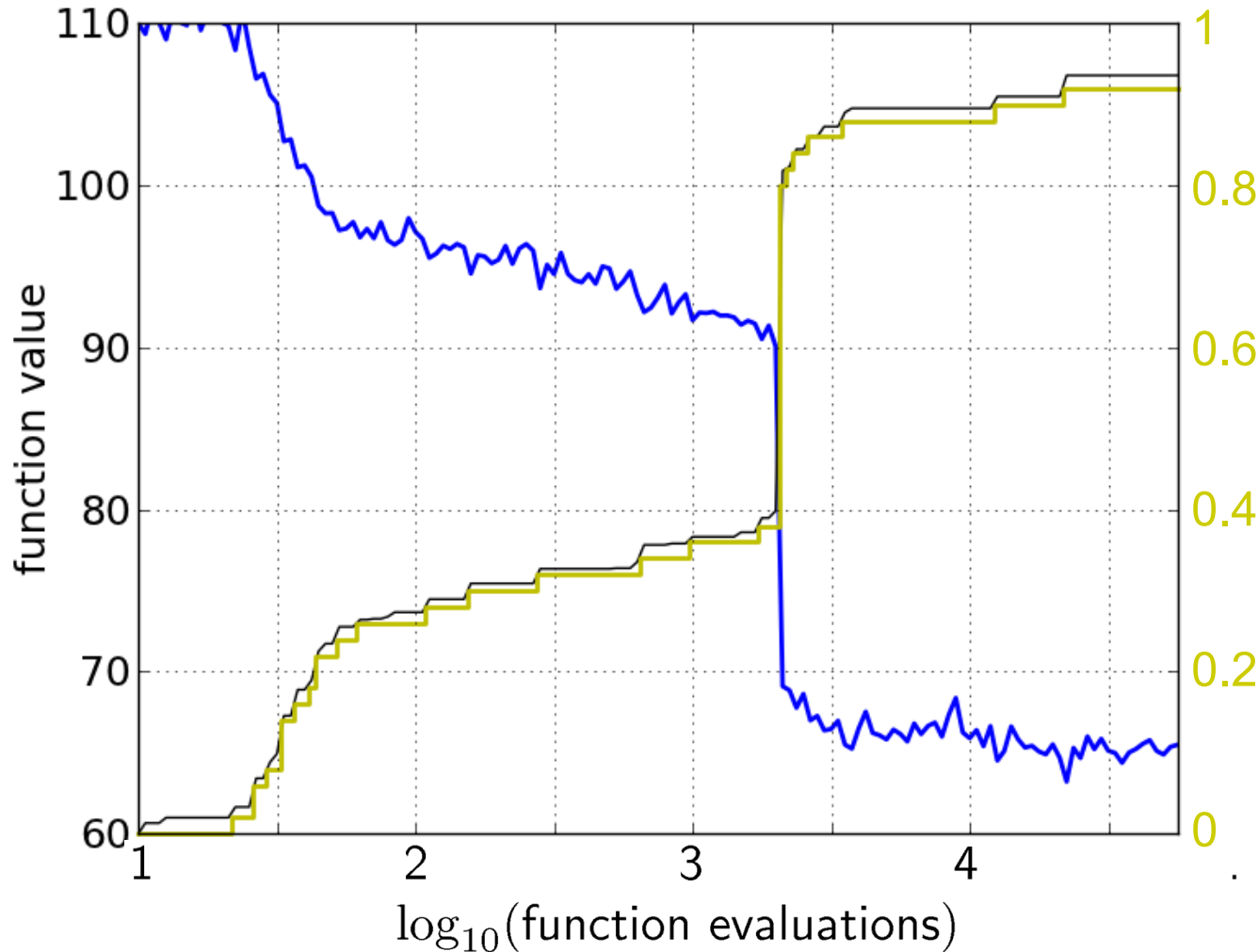
The empirical CDF makes a step for each star, is monotonous and displays for each budget the fraction of targets achieved within the budget

Reconstructing A Single Run



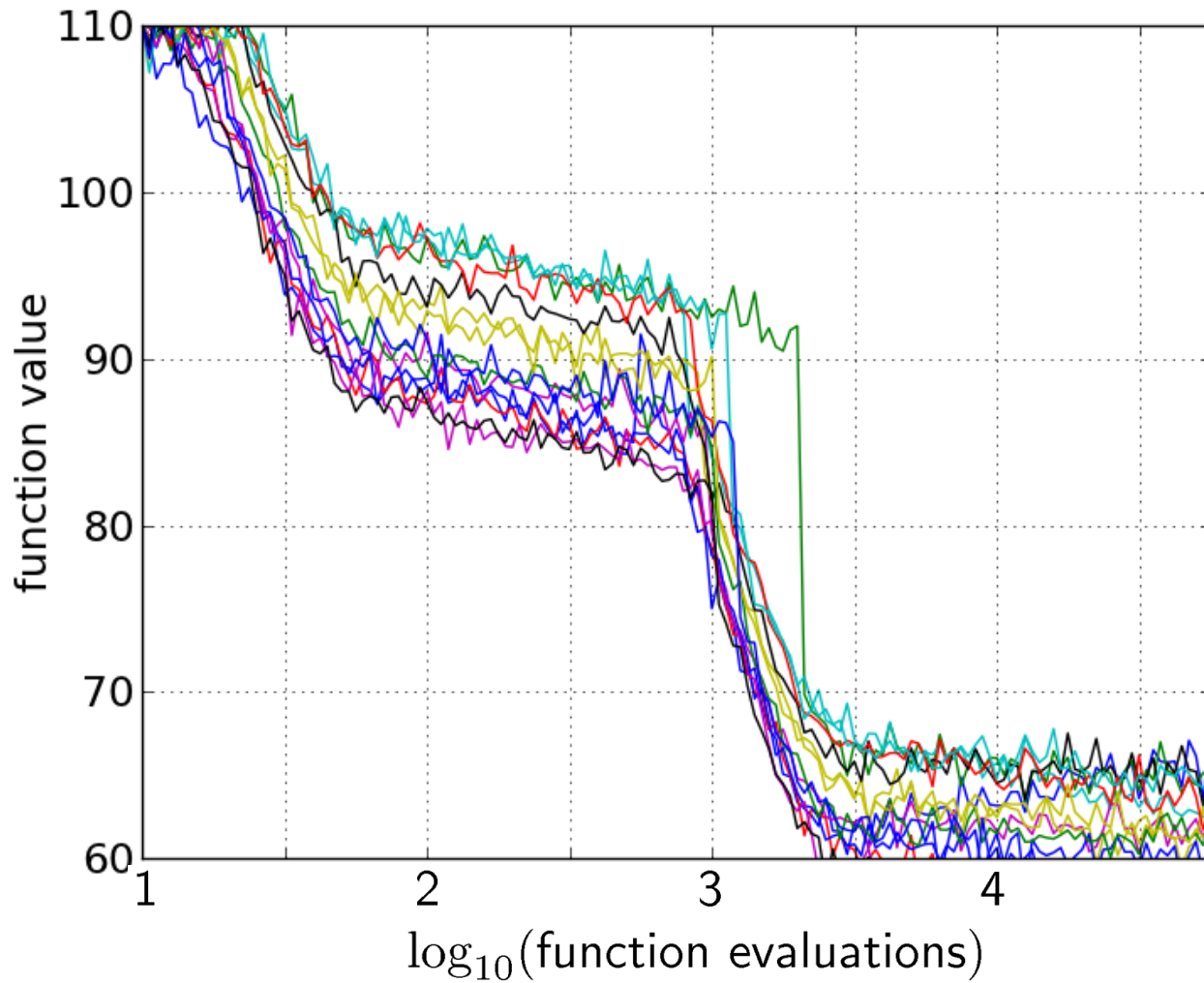
the ECDF recovers
the monotonous
graph,
discretised and
flipped

Reconstructing A Single Run



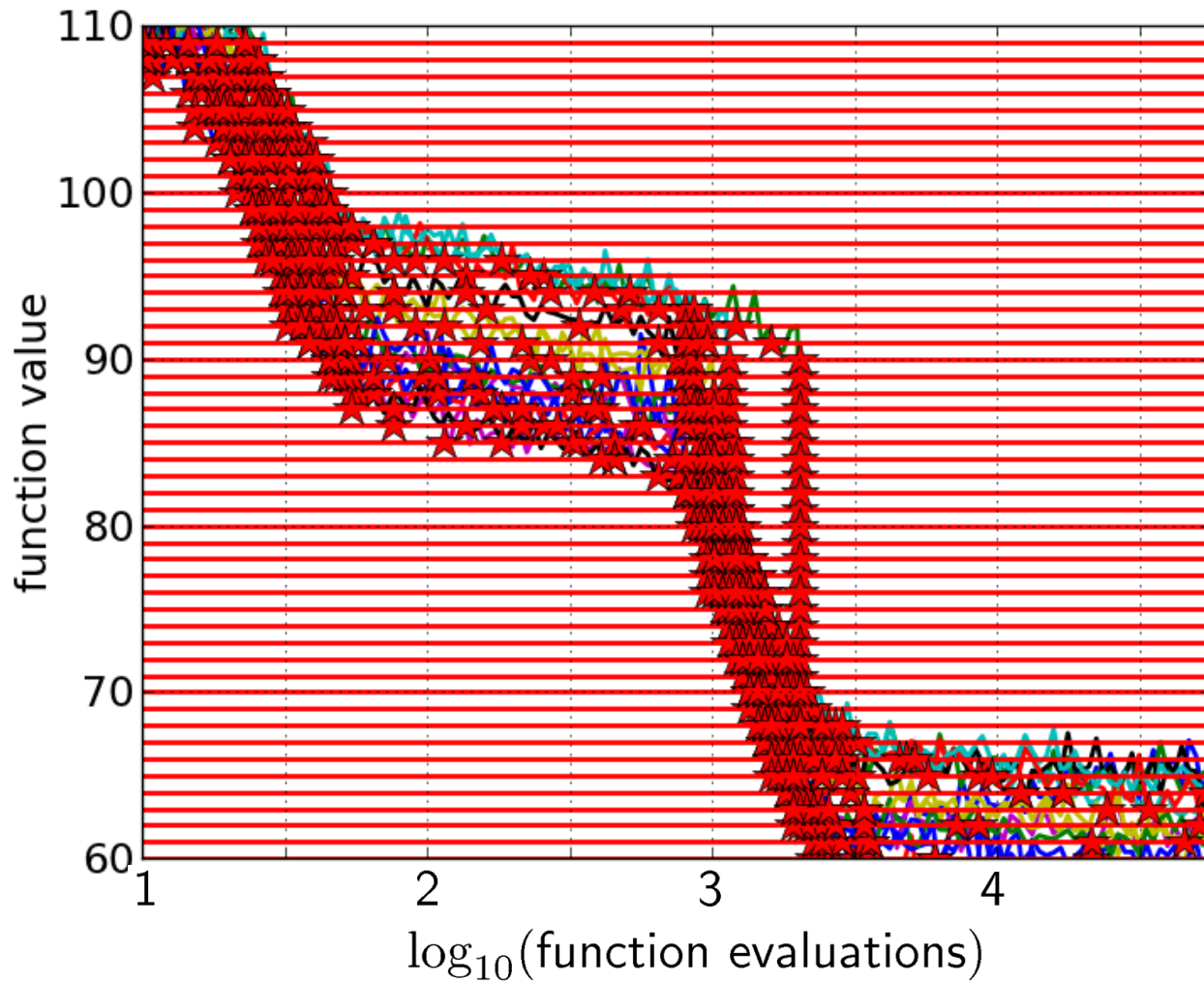
the ECDF recovers
the monotonous
graph,
discretised and
flipped

Aggregation



15 runs

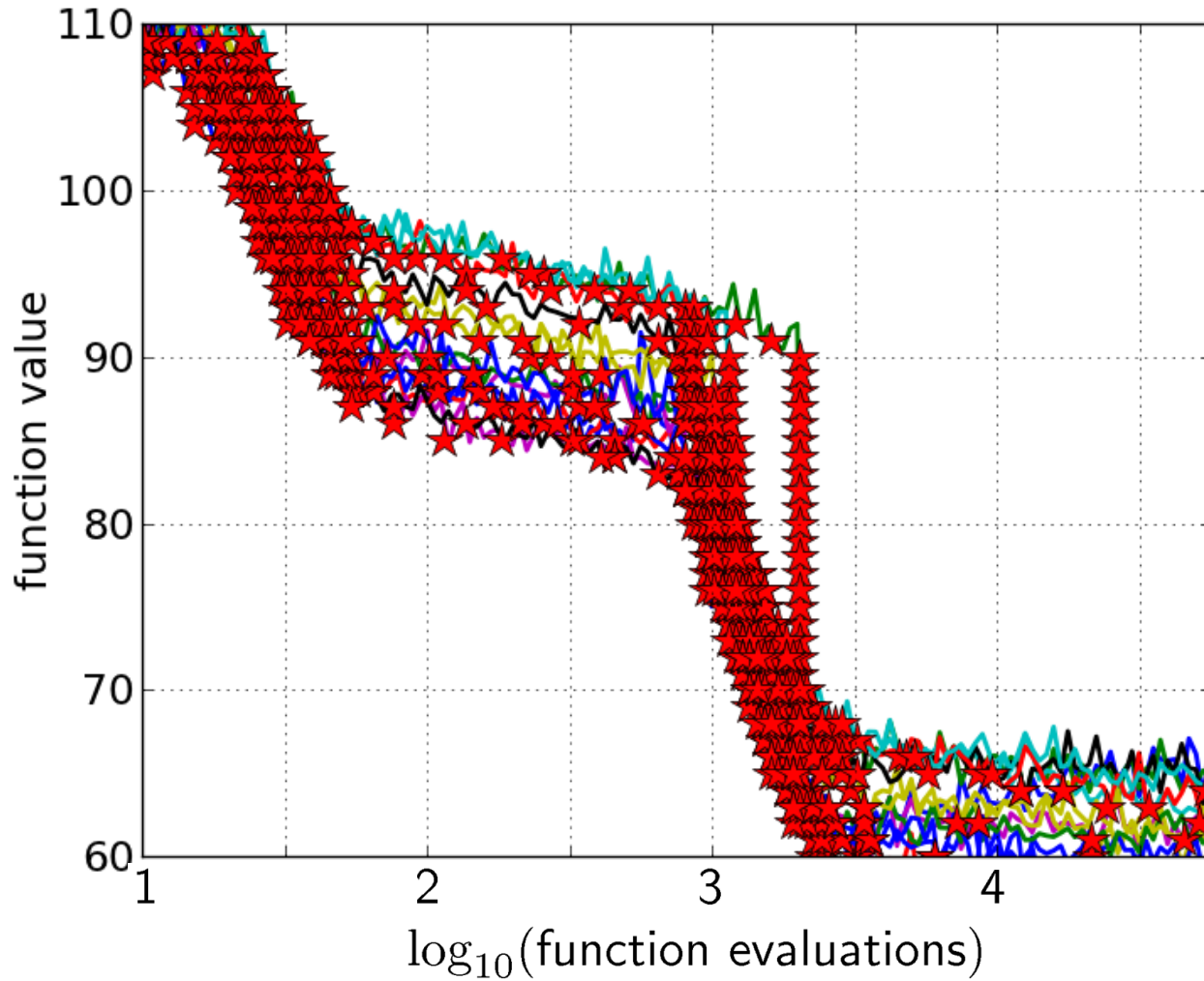
Aggregation



15 runs

50 targets

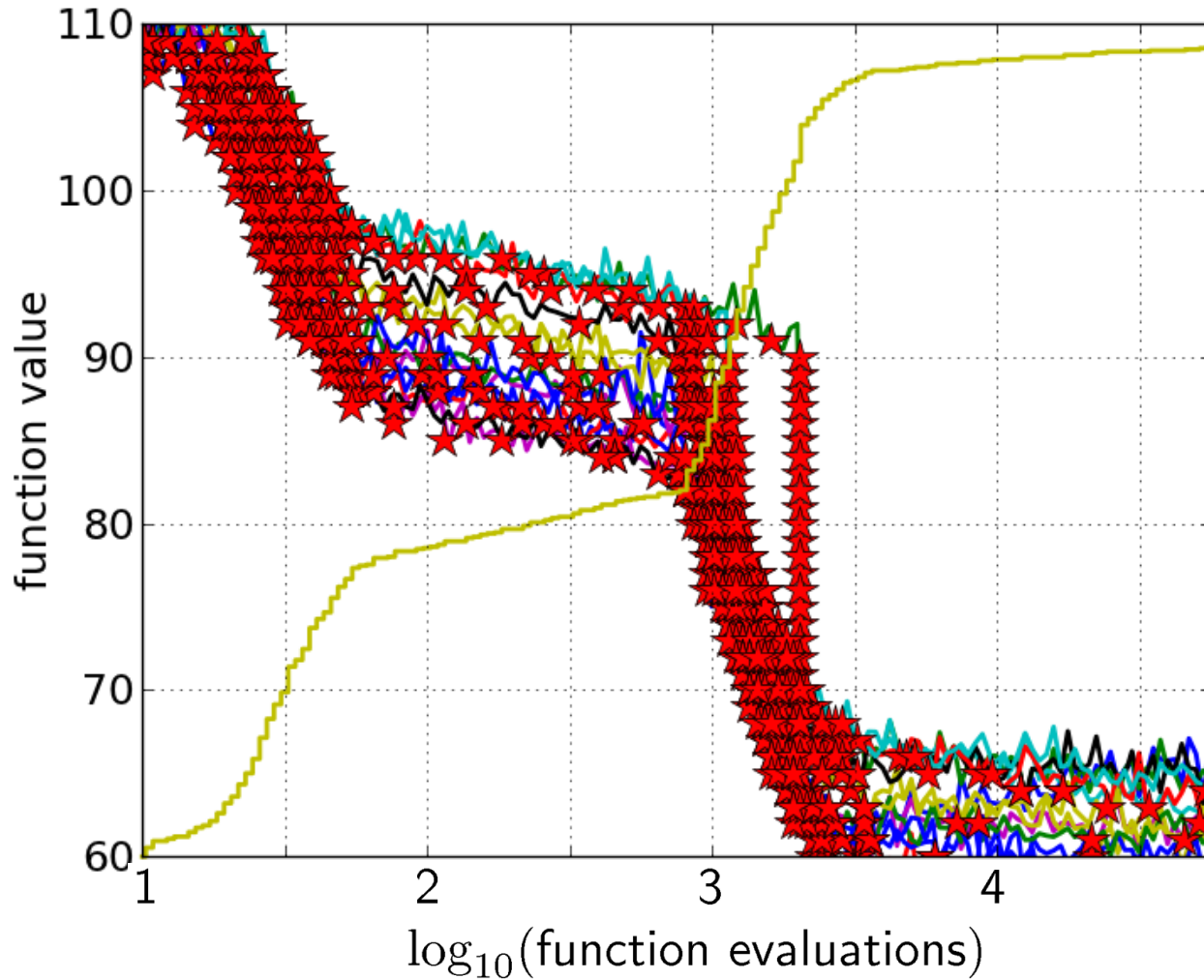
Aggregation



15 runs

50 targets

Aggregation

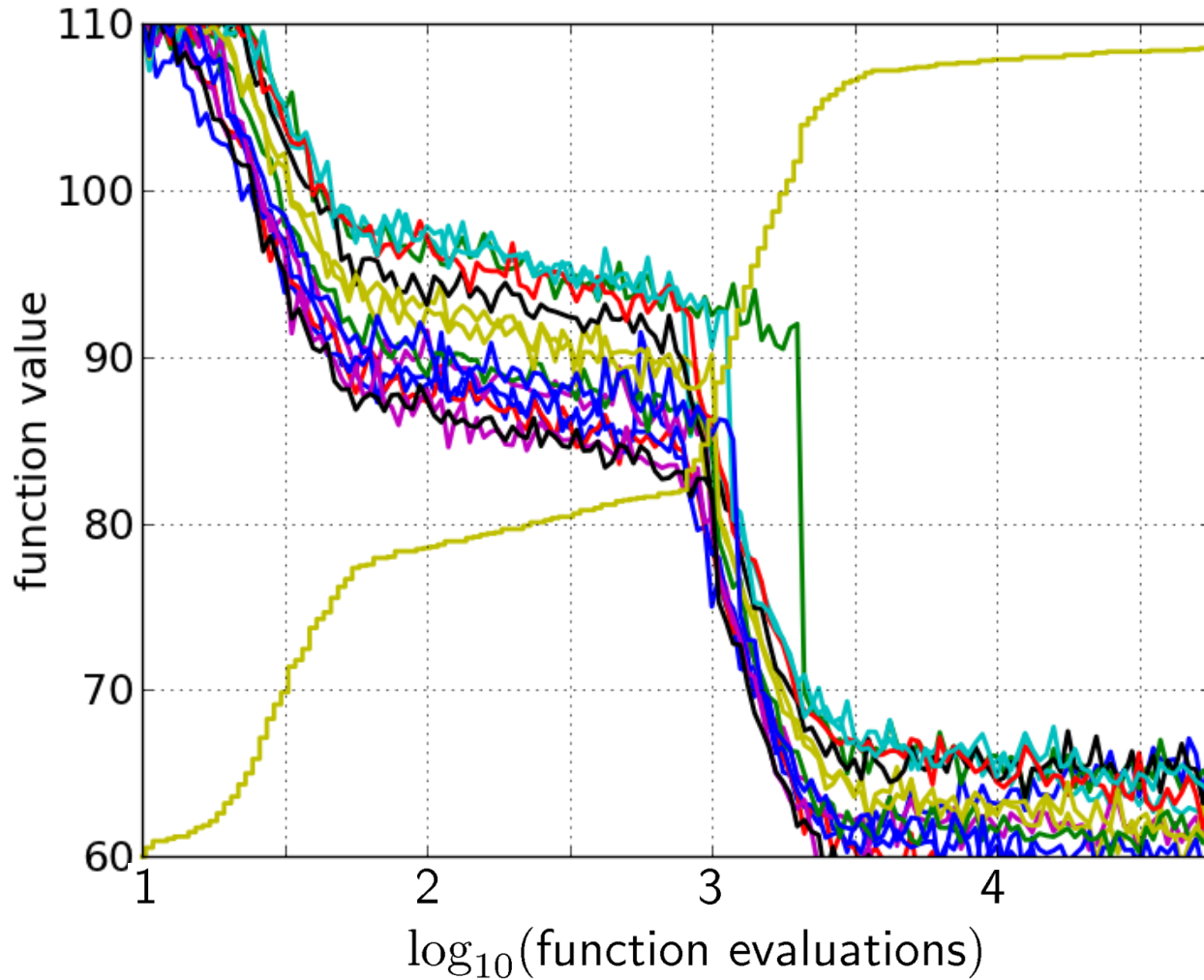


15 runs

50 targets

ECDF with 750
steps

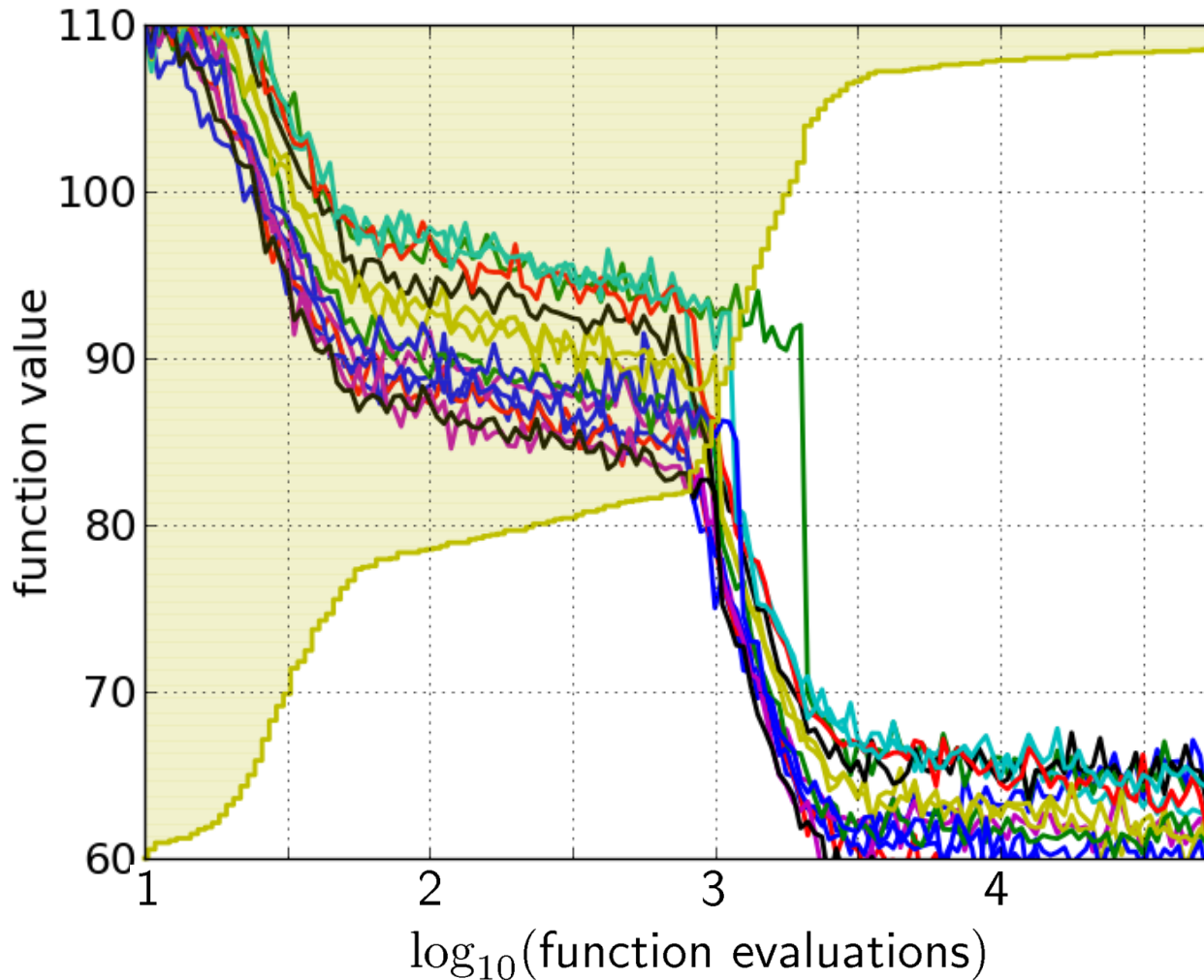
Aggregation



50 targets from
15 runs

...integrated in a
single graph

Interpretation



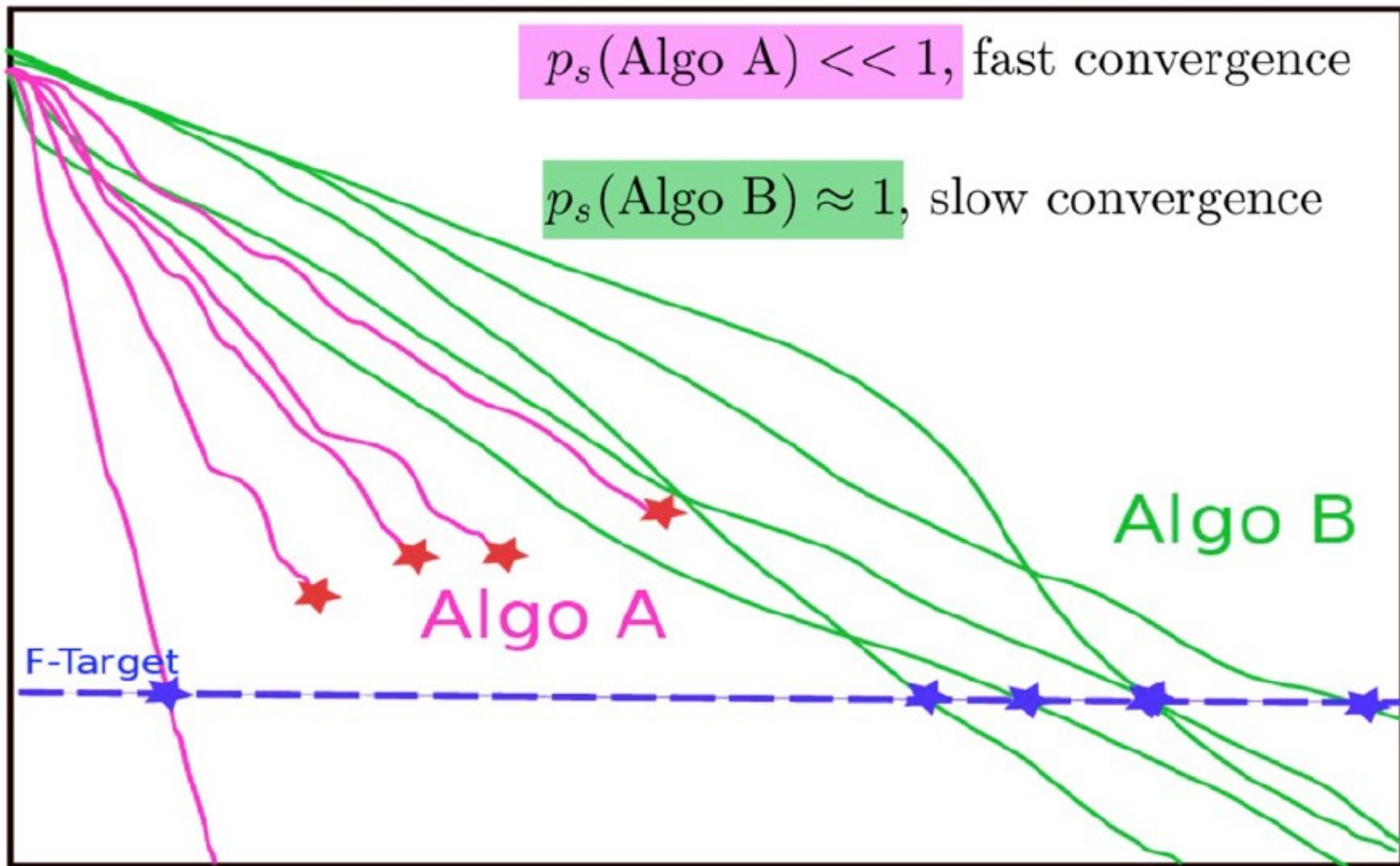
50 targets from
15 runs
integrated in a
single graph

area over the ECDF
curve

=

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

Fixed-target: Measuring Runtime



Fixed-target: Measuring Runtime

- Algo Restart A:



- Algo Restart B:



Fixed-target: Measuring Runtime

- Expected running time of the restarted algorithm:

$$E[RT^r] = \frac{1 - p_s}{p_s} E[RT_{unsuccessful}] + E[RT_{successful}]$$

- Estimator average running time (aRT):

$$\hat{p}_s = \frac{\text{\#successes}}{\text{\#runs}}$$

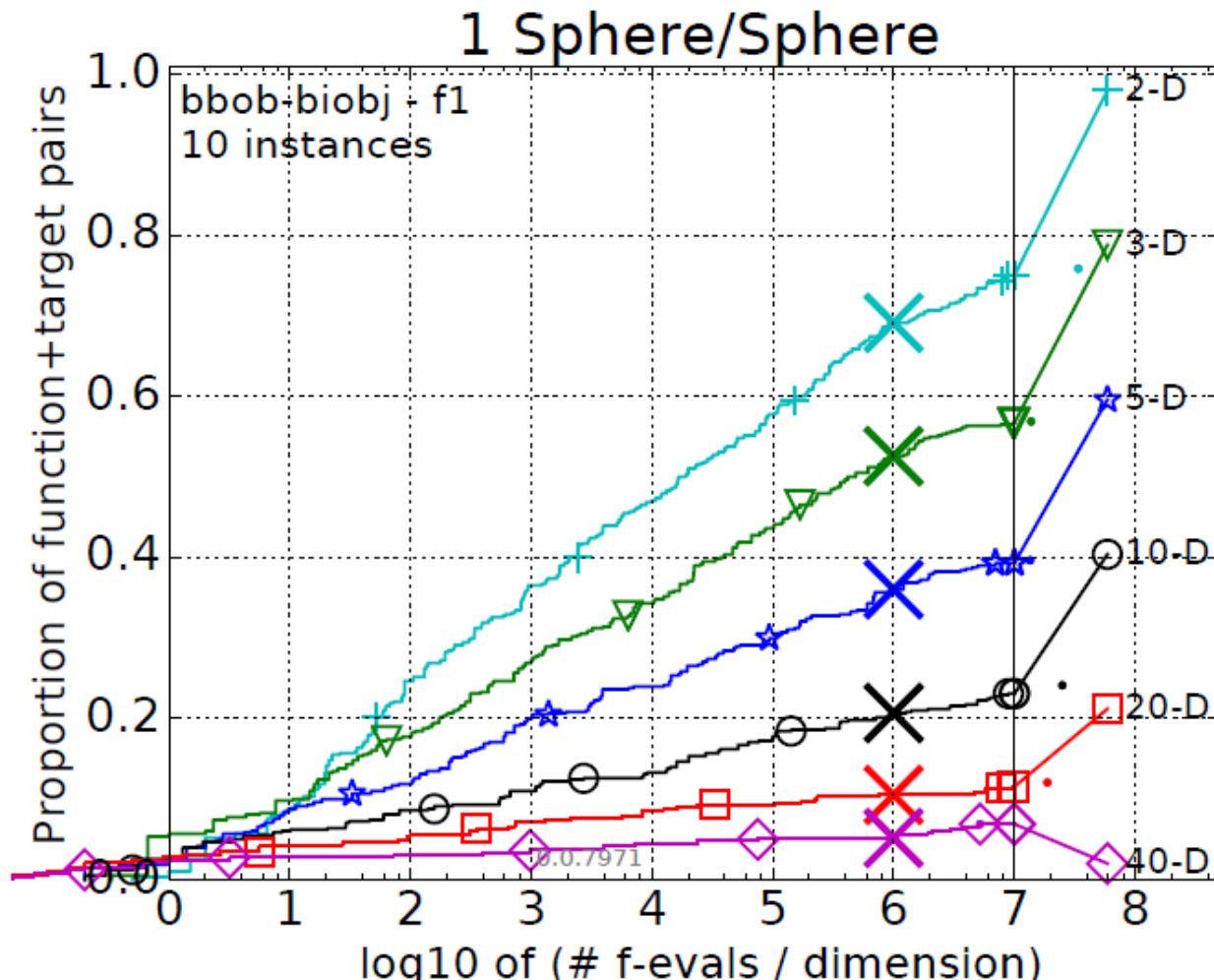
\widehat{RT}_{unsucc} = Average evals of unsuccessful runs

\widehat{RT}_{succ} = Average evals of successful runs

$$aRT = \frac{\text{total \#evals}}{\text{\#successes}}$$

ECDFs with Simulated Restarts

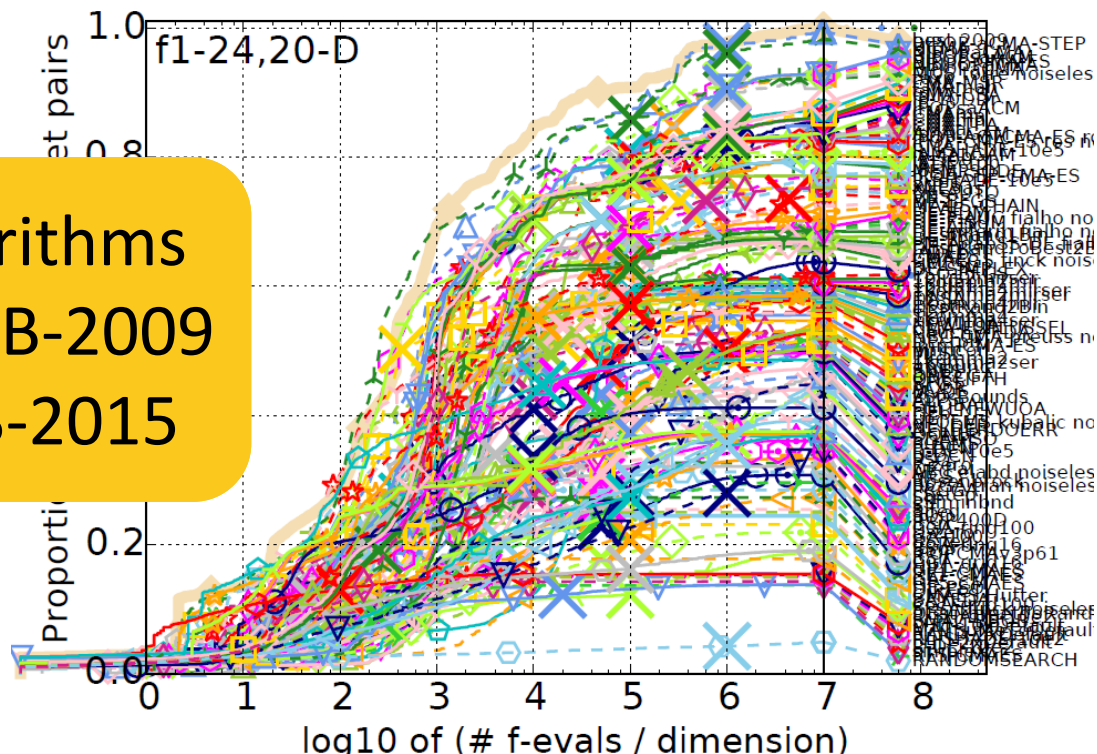
What we typically plot are ECDFs of the simulated restarted algorithms:



Worth to Note: ECDFs in COCO

In COCO, ECDF graphs

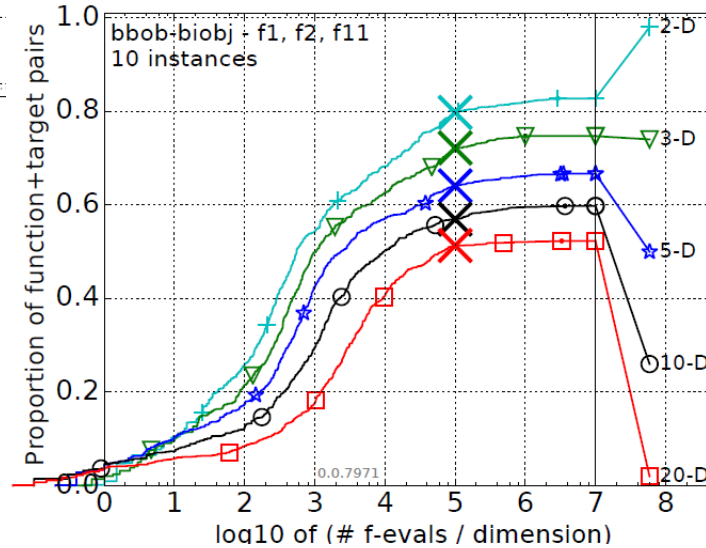
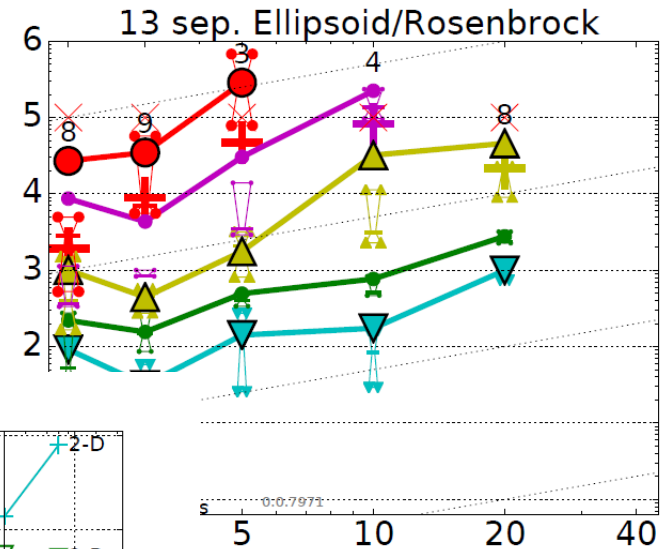
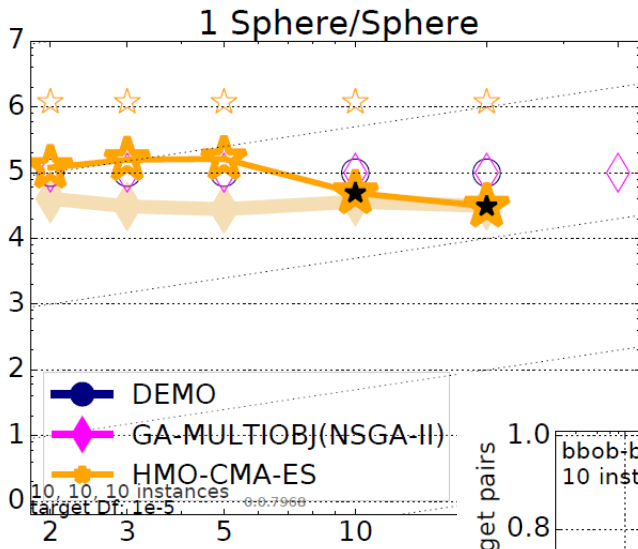
- never aggregate over dimension
 - but often over targets and functions
- can show data of more than 1 algorithm at a time



150 algorithms
from BBOB-2009
till BBOB-2015

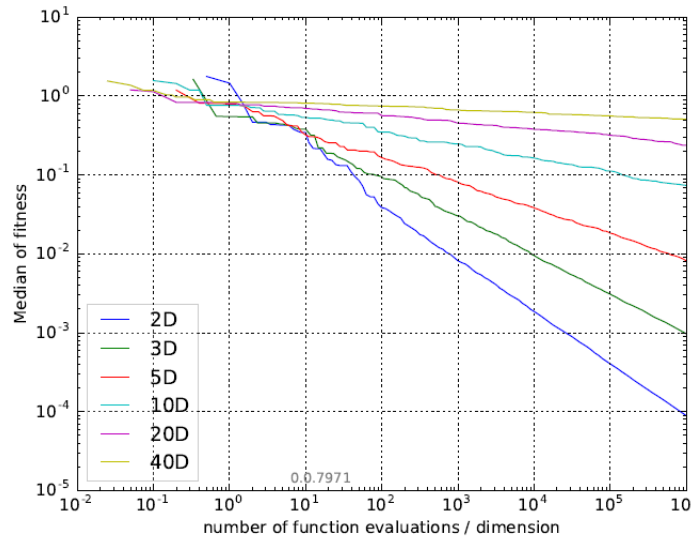
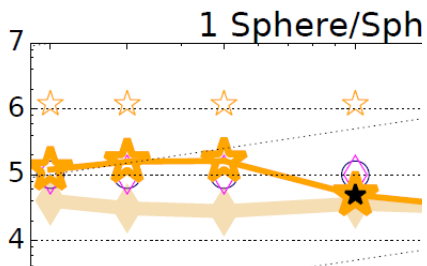
More Automated Plots...

...but no time to explain them here ☹️

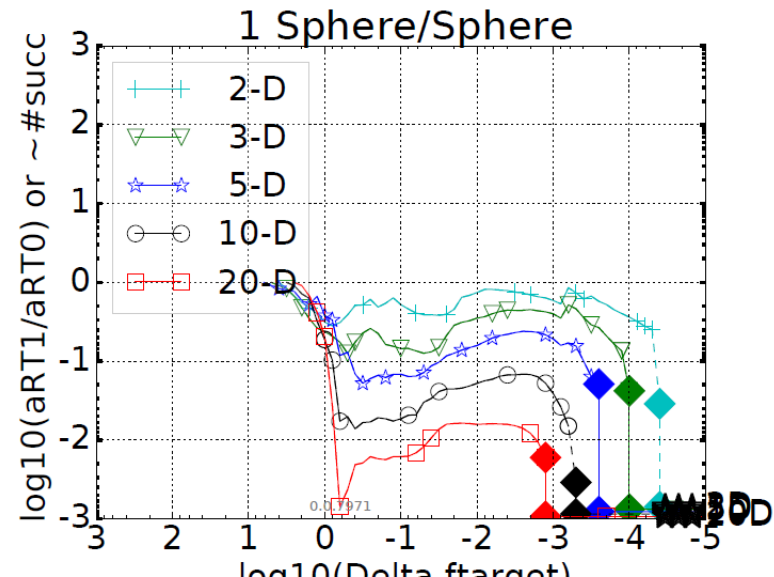
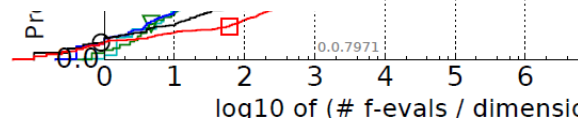
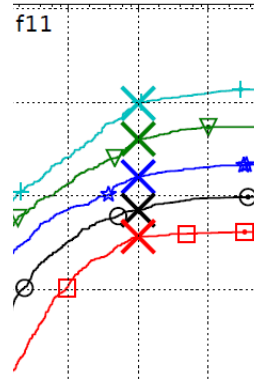
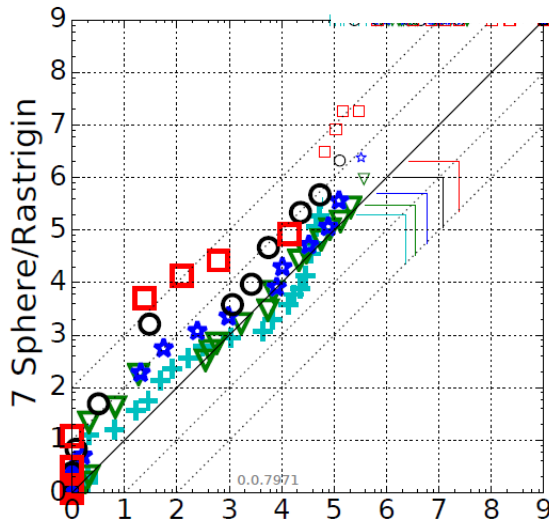
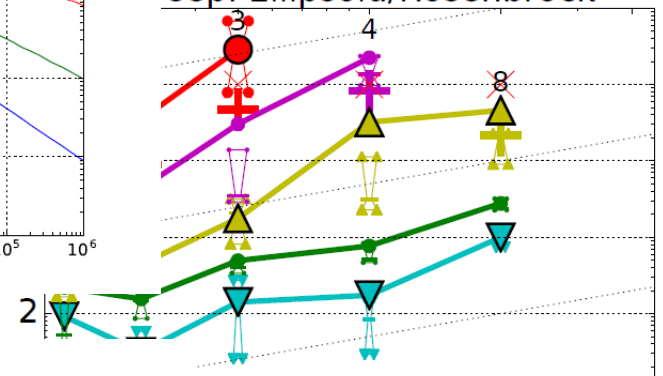


More Automated Plots...

...but no time t



sep. Ellipsoid/Rosenbrock



and now?

BBOB-2018

Session Sunday 15th of July, 2018 - Training Room 1 (2F)

09:30 - 09:45	The BBOBies: A Short Introduction to COCO and BBOB
09:45 - 10:05	Kouhei Nishida* and Youhei Akimoto: Benchmarking the PSA-CMA-ES on the BBOB Noiseless Testbed
10:05 - 10:25	Duc Manh Nguyen: Benchmarking a Variant of the CMAES-APOP on the BBOB Noiseless Testbed
10:25 - 10:40	Aurore Blelly, Matheus Felipe-Gomes, Anne Auger, and Dimo Brockhoff*: Stopping Criteria, Initialization, and Implementations of BFGS and their Effect on the BBOB Test Suite
10:40 - 11:00	Aljoša Vodopija, Tea Tušar*, Bogdan Filipič: Comparing Black-Box Differential Evolution and Classic Differential Evolution
11:00 - 11:10	The BBOBies: Workshop Wrapup and Discussion

http://coco.gforge.inria.fr/

The screenshot shows a web browser window displaying the COCO website. The browser's address bar shows the URL `http://coco.gforge.inria.fr/`. The page title is `COMPARING CONTINUOUS OPTIMISERS: COCO`. The main content area features a graph titled `COMPARING CONTINUOUS OPTIMISERS: COCO` showing the proportion of functions solved versus running length/dimension for various optimizers. The x-axis is logarithmic, ranging from 10^0 to 10^8 . The y-axis is linear, ranging from 0.0 to 1.0. The legend lists 28 optimizers, including BIPOP-CMA-ES, AMALGAM IDEA, IAmALGAM IDEA, MA-LS-Chain, VNS (Garcia), IPOPOP-SQP-CMA-ES, ALPS-GA, POEMS, Cauchy/EDA, EDA-PSO, (1+1)-CMA-ES, DASA, NELDER (Han), PSO Bounds, NELDER (Doe), PSO, (1+1)-ES, Full NEWUA, GLOBAL, BFGS, Rosenbrock, MCS, simple GA, Sminband, L-Step, DIRECT, DE-PSO, BayEAcG, and Monte Carlo. A navigation menu on the right lists links for Home, Documentation, download latest old code, new code homepage, download new code directly, BBOB 2016, BBOB 2015 @ GECCO, BBOB 2015 @ CEC, BBOB 2013, BBOB 2012, BBOB 2010, and BBOB 2009, each with sub-links for Algorithms, Results, Schedule, and Downloads.

[[start]]

COMPARING CONTINUOUS OPTIMISERS: COCO

Show pagesource Old revisions Recent changes Sitemap Login

COCO (Comparing Continuous Optimisers) is a platform for systematic and sound comparisons of real-parameter global optimisers. COCO provides benchmark function testbeds, experimentation templates which are easy to parallelize, and tools for processing and visualizing data generated by one or several optimizers. The COCO platform has been used for the Black-Box-Optimization-Benchmarking (BBOB) workshops that took place during the GECCO conference in 2009, 2010, 2012, 2013 and 2015. It was also used at the IEEE Congress on Evolutionary Computation (CEC'2015) in Sendai, Japan. The COCO source code is available at the [downloads](#) page.

- Black-Box Optimization Benchmarking (BBOB) 2016
- Black-Box Optimization Benchmarking (BBOB) 2015
- CEC'2015 special session on Black-Box Optimization Benchmarking (CEC-BBOB 2013)
- Black-Box Optimization Benchmarking (BBOB) 2013
- Black-Box Optimization Benchmarking (BBOB) 2012
- Black-Box Optimization Benchmarking (BBOB) 2010
- Black-Box Optimization Benchmarking (BBOB) 2009
- Downloads and documentations

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

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

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
 - Results
 - Downloads