Exercise: Benchmarking Derivative-Free Optimization Algorithms

July 5, 2017

CEA/EDF/Inria summer school "Numerical Analysis"

Université Pierre-et-Marie-Curie, Paris, France

Anne Auger, Asma Atamna, and Dimo Brockhoff

Inria Saclay – Ile-de-France

CMAP, Ecole Polytechnique

Requirements:

python 2.6 or 2.7

use Anaconda if not already installed:
https://www.continuum.io/downloads)

note: python 3.x not yet supported

matplotlib >2.0

in principle via `pip install matplotlib`
otherwise, see http://matplotlib.org/users/installing.html

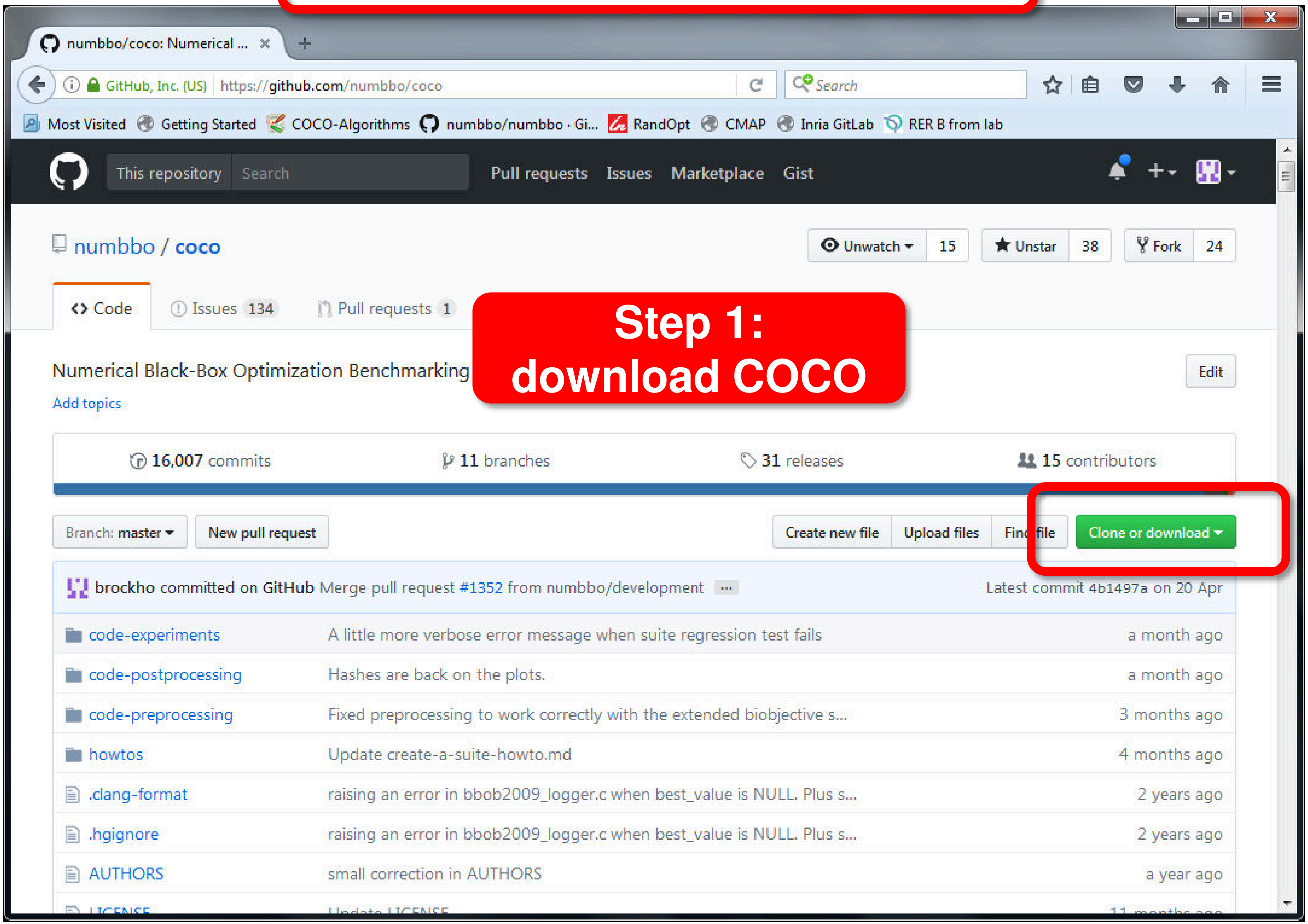
Objectives of the exercise:

❶ Get acquainted with the COCO platform
at least with its postprocessing and visualization

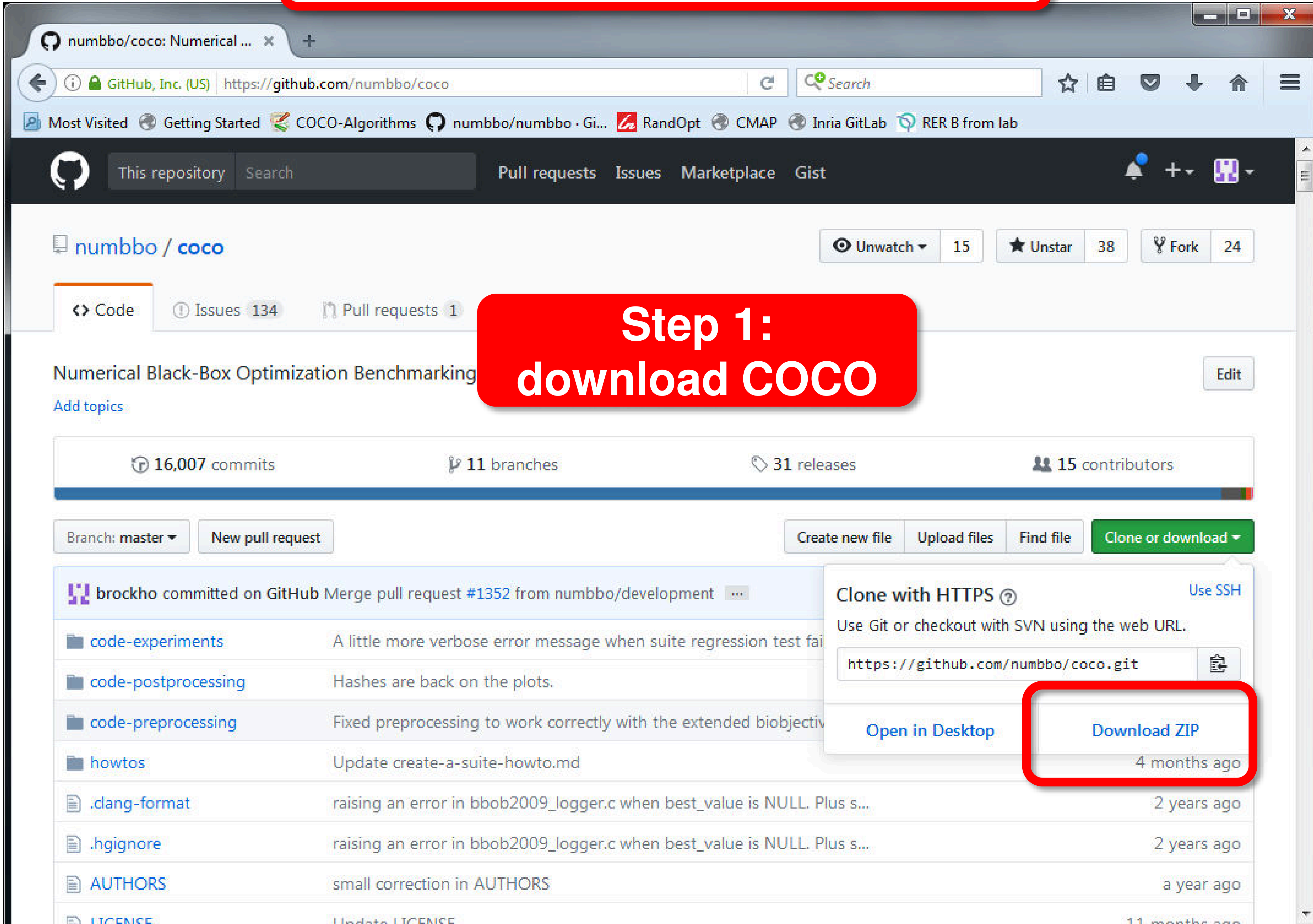
❷ Gain insights into data
of some of the 150+ algorithm data sets of COCO

https://github.com/numbbo/coco

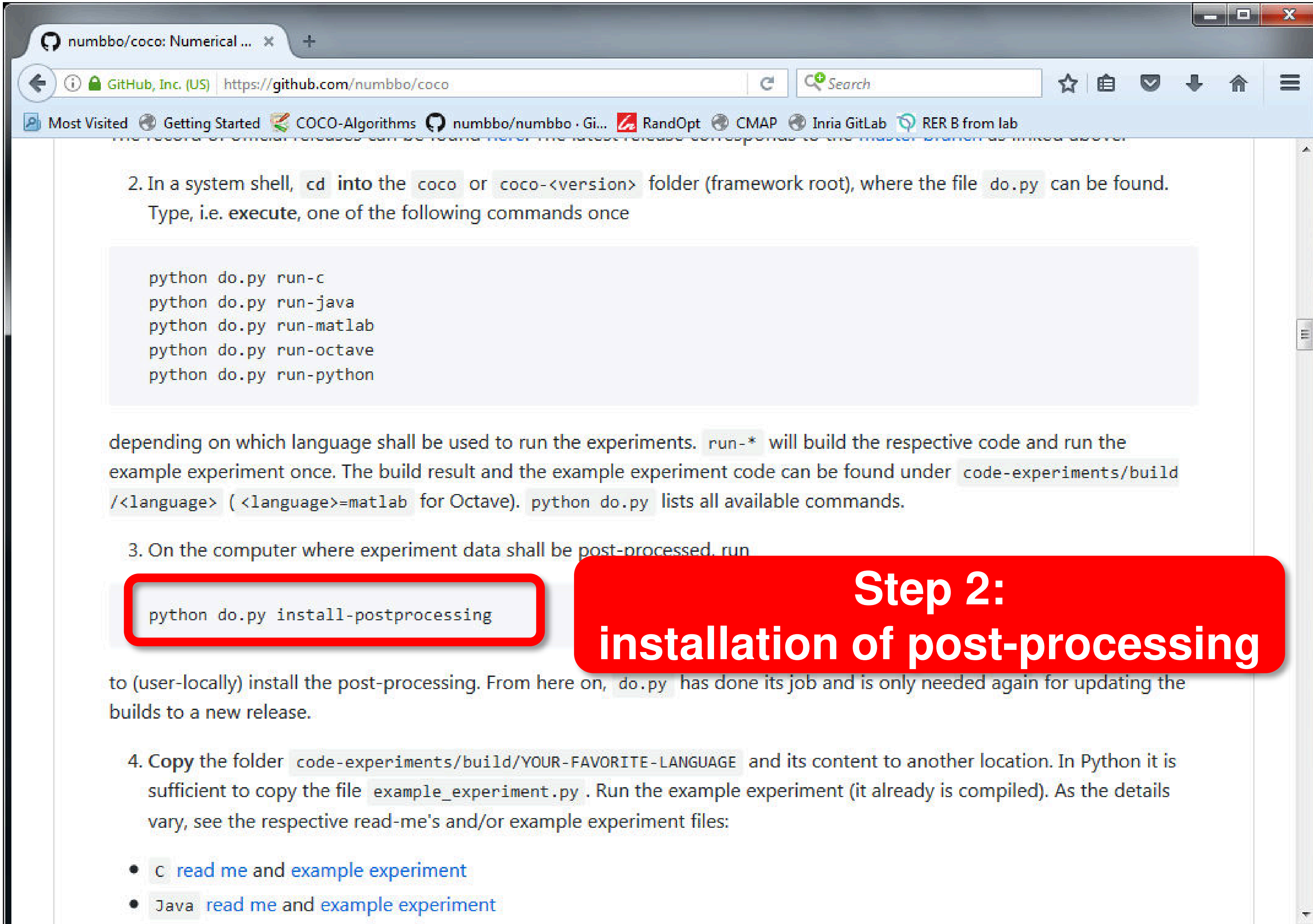
Step 1: download COCO



https://github.com/numbbo/coco



https://github.com/numbbo/coco



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
```

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

Step 2:
installation of post-processing

http://coco.gforge.inria.fr/doku.php?id=algorithms

Step 3: downloading data

[[algorithms]] COMPARING CONTINUOUS OPTIMISERS: COCO

Show pagesource Old revisions Recent changes Sitemap Login

Search

The following table lists all algorithms related to the BBOB workshops and special sessions in the years 2009 till 2015 together with links to their data. In order to sort the table according to some columns, please click on the corresponding table header. If available, the source codes of the algorithms can be downloaded by clicking on the link with the corresponding algorithm name in the second column.

No	Algorithm	Year	Author(s)	Data Noiseless (Raw)	Data Noisy (Raw)	related PDFs and Remarks
1	ALPS	2009	Hornby	noiselessData	noisyData	PDF
2	AMALGAM	2009	Bosman et al.	noiselessData	noisyData	PDFnoiseless PDFnoisy
3	BAYEDA	2009	Gallagher	noiselessData	noisyData	PDFnoiseless PDFnoisy
4	BFGS	2009	Ros	noiselessData	noisyData	PDFnoiseless PDF
5	BIPOP-CMA-ES	2009	Hansen	noiselessData	noisyData	PDFnoiseless PDF
6	Cauchy-EDA	2009	Pošík	noiselessData	n/a	PDF
7	CMA-ESPLUSSEL	2009	Auger and Hansen	noiselessData	noisyData	PDFnoiseless PDF
8	DASA	2009	Korošec and Šilc	noiselessData	noisyData	PDFnoiseless PDF
9	DE-PSO	2009	García-Nieto et al.	noiselessData	noisyData	PDFnoiseless PDFno
10	DIRECT	2009	Pošík	noiselessData	n/a	PDF algorithm is deterministic and thus, only run on each instance once
11	EDA-PSO	2009	El-Abd and Kamel	noiselessData	noisyData	PDF

Navigation

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

for the moment: IPOP-CMA-ES (algorithm 56)

https://github.com/numbbo/coco

coco/README.md at devel... | 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

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]
```

postprocess

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

```
python -m cocopp IPOP-CMA-ES_ros_noiseless.tar.gz
```

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

https://github.com/numbbo/coco

postprocess

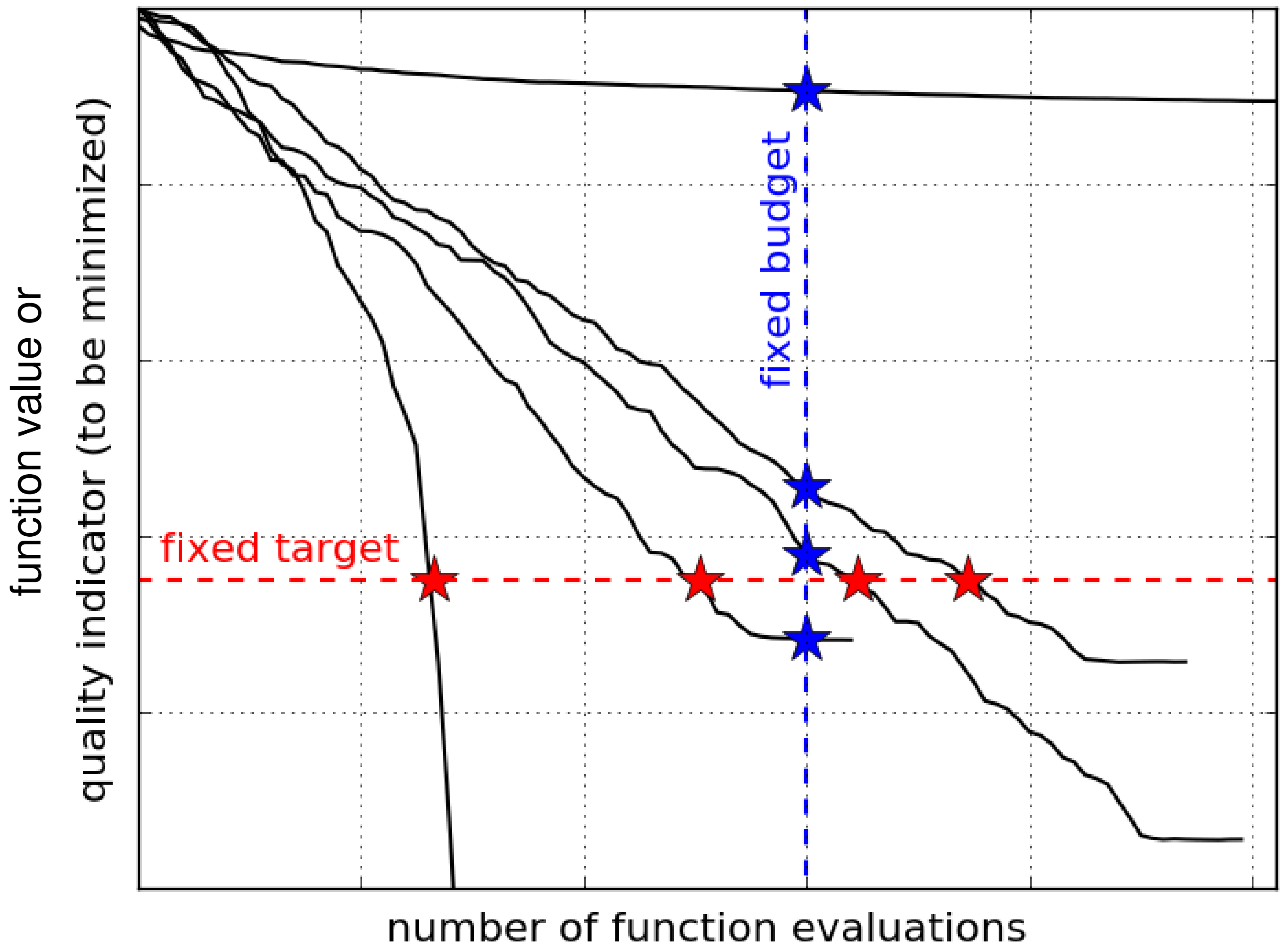
```
python -m cocopp IPOP-CMA-ES_ros_noiseless.tar.gz
```

Alternative within (I)python:

```
> import cocopp
> cocopp.main("IPOP-CMA-ES_ros_noiseless.tar.gz")
```

Reminder:
Measuring Performance Empirically

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

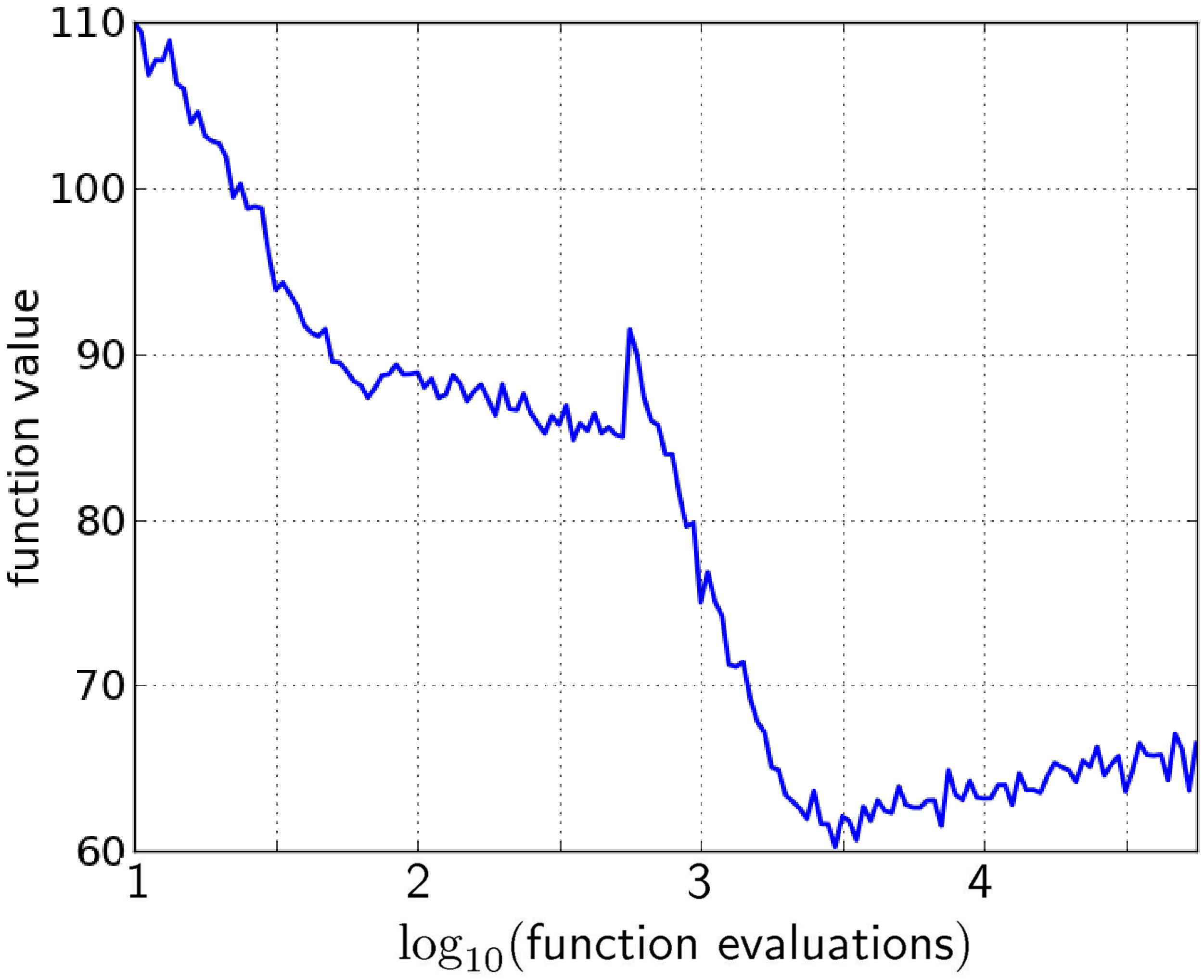


ECDF:

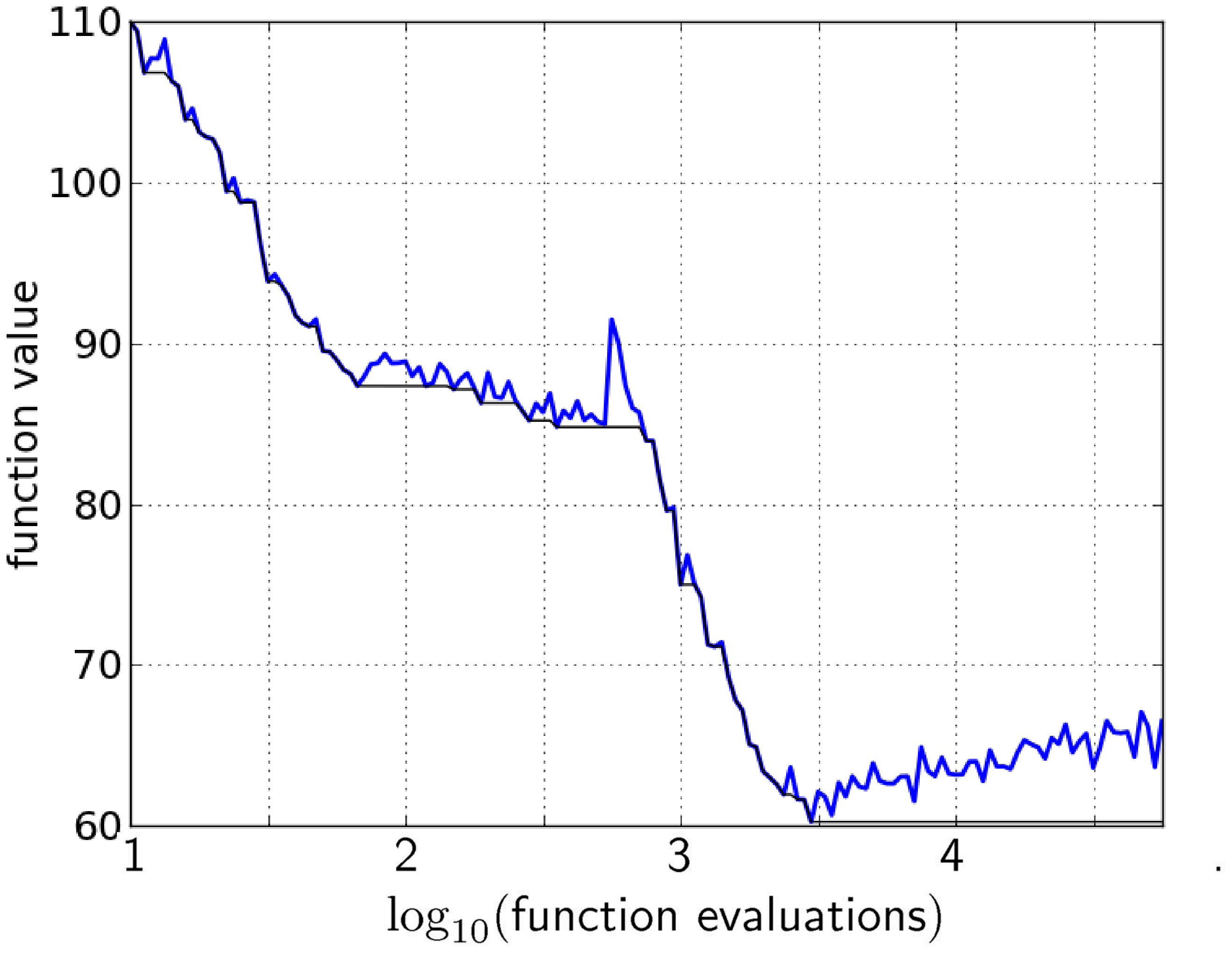
Empirical Cumulative Distribution Function of the Runtime

[aka data profile]

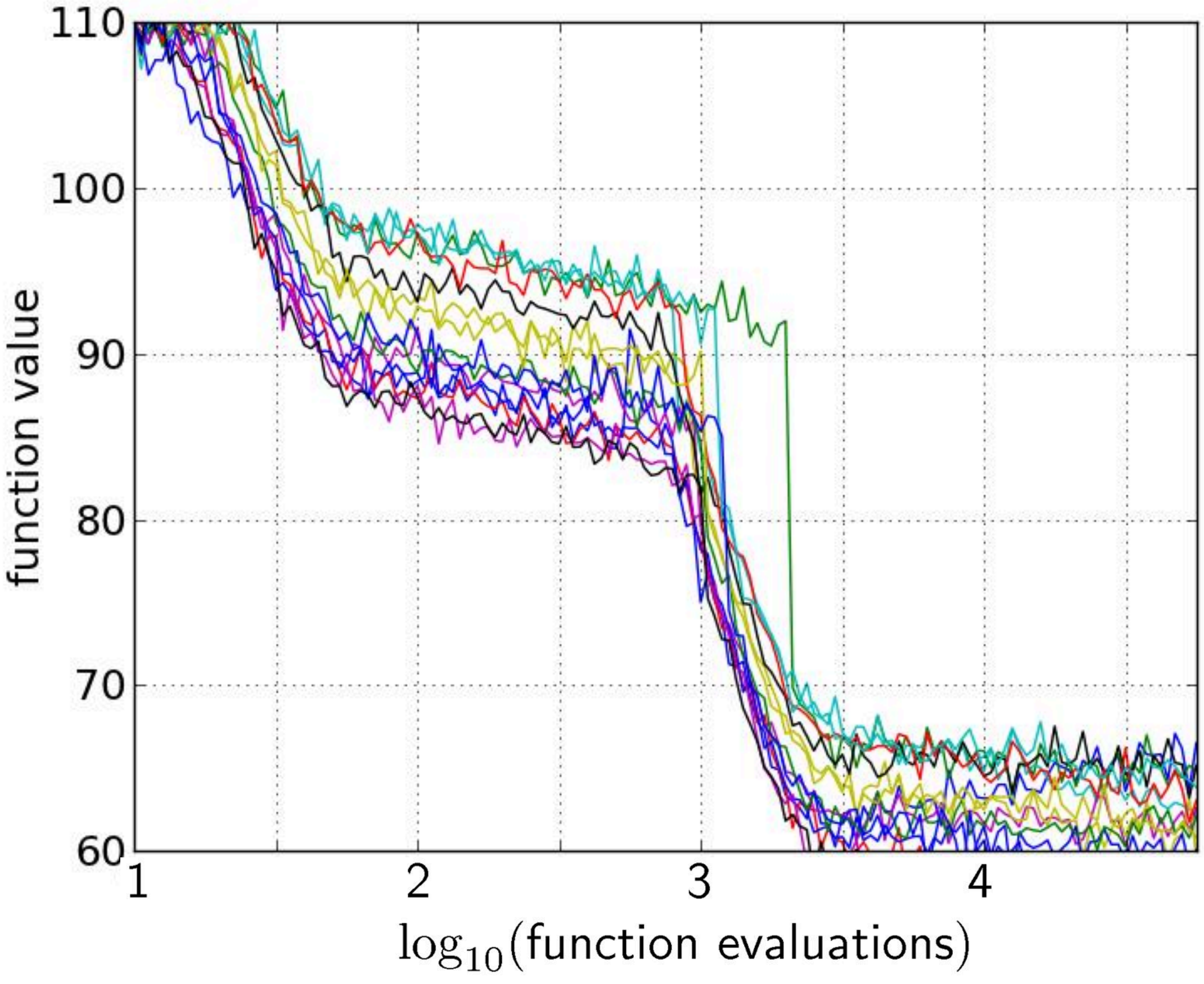
A Convergence Graph



First Hitting Time is Monotonous

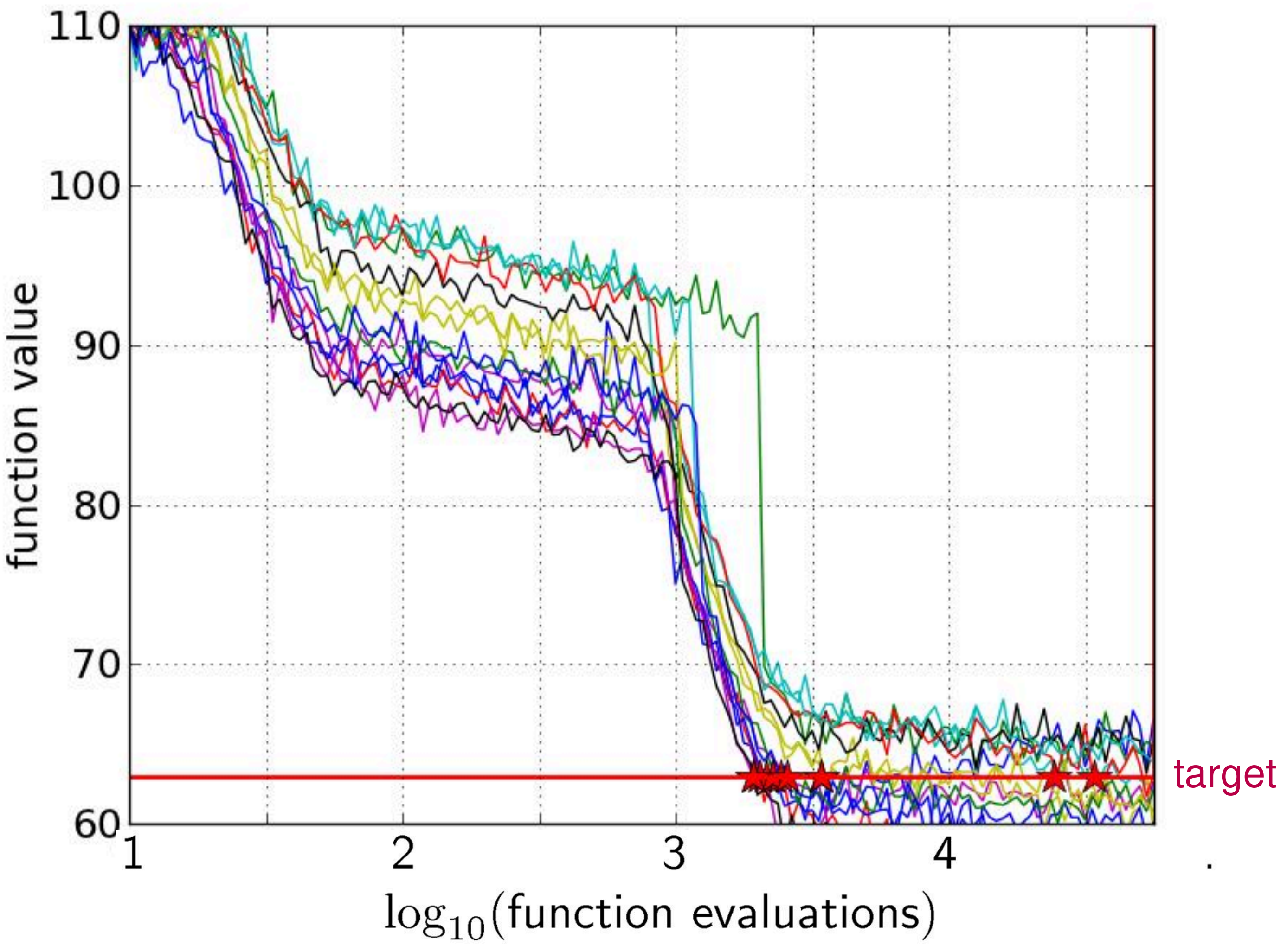


15 Runs

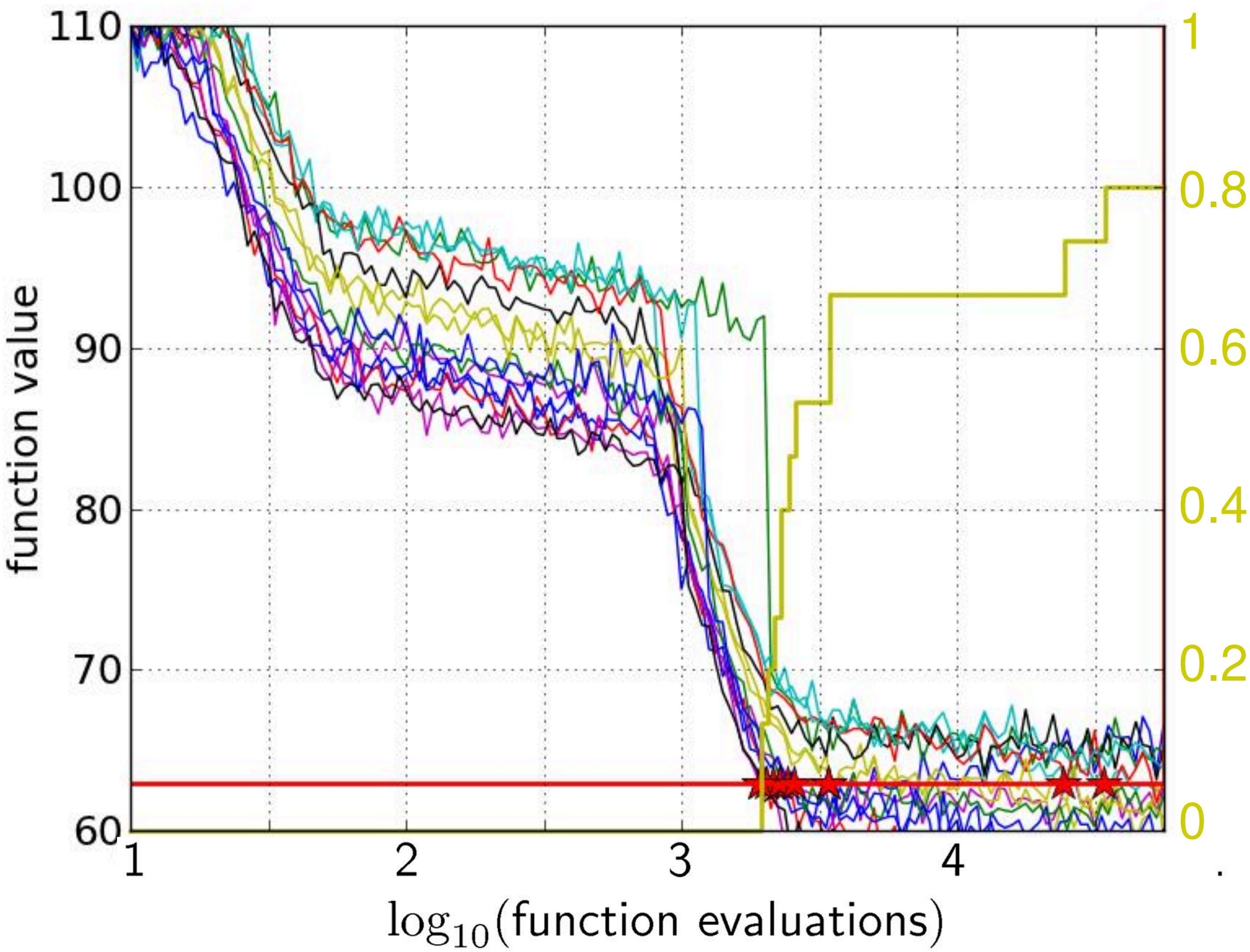


110
100
90
80
70
60
function value
1
2
3
4
$\log_{10}$(function evaluations)

15 Runs ≤ 15 Runtime Data Points

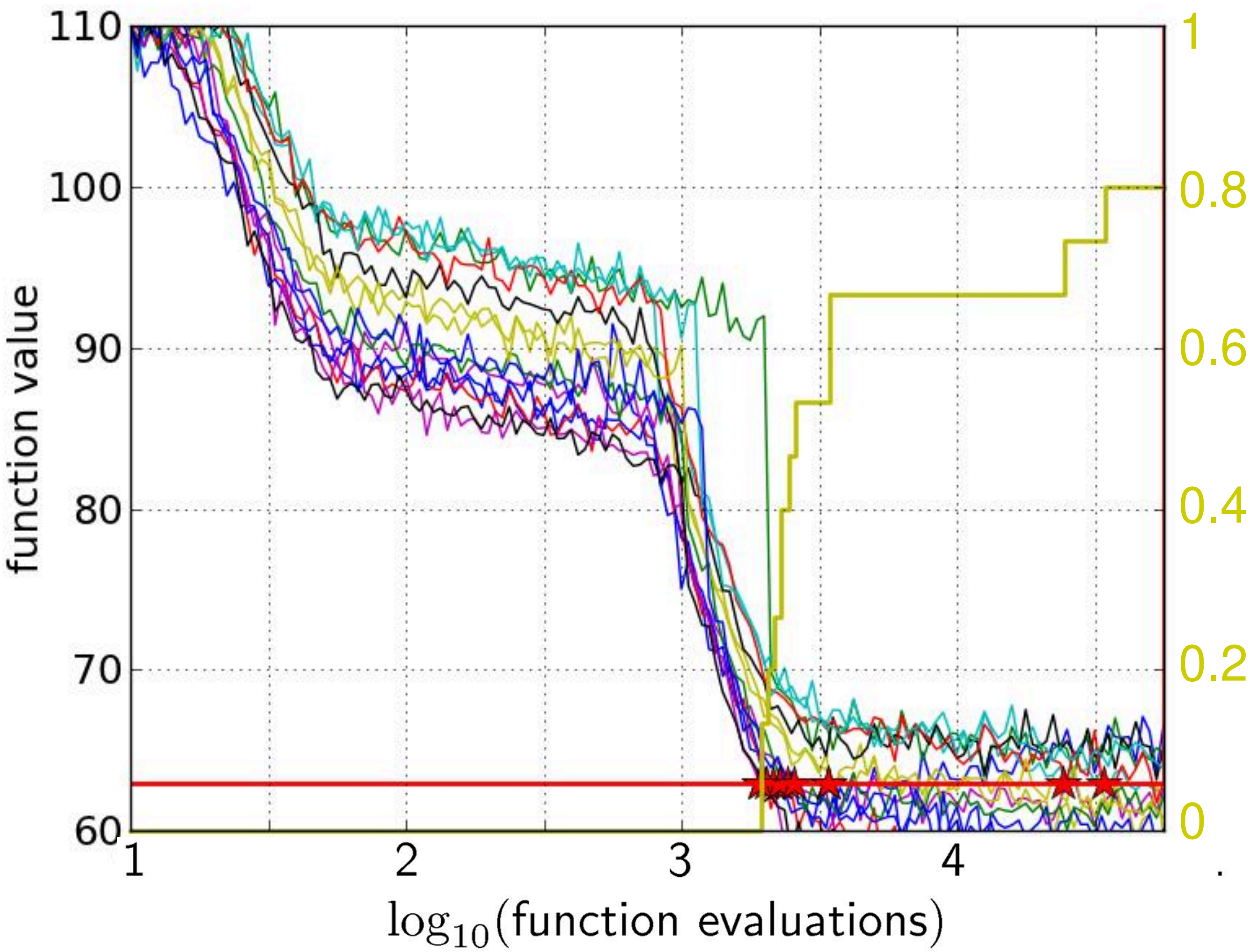


Empirical Cumulative Distribution



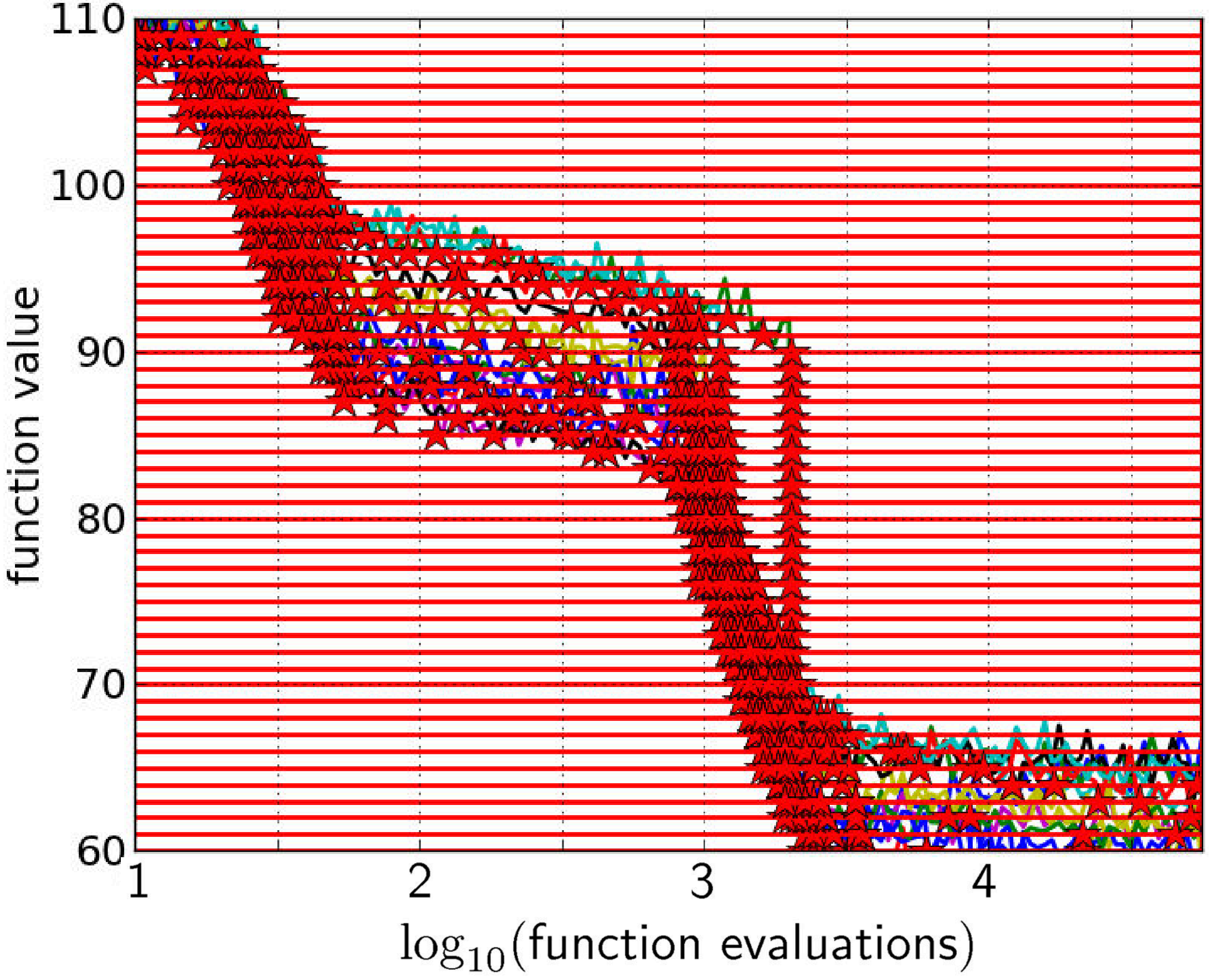
- the ECDF of run lengths to reach the target
- has for each data point a vertical step of constant size
- displays for each x-value (budget) the count of observations to the left (first hitting times)

Empirical Cumulative Distribution



- the higher the better (= more targets solved)
- the more to the left the better (= targets solved quicker)

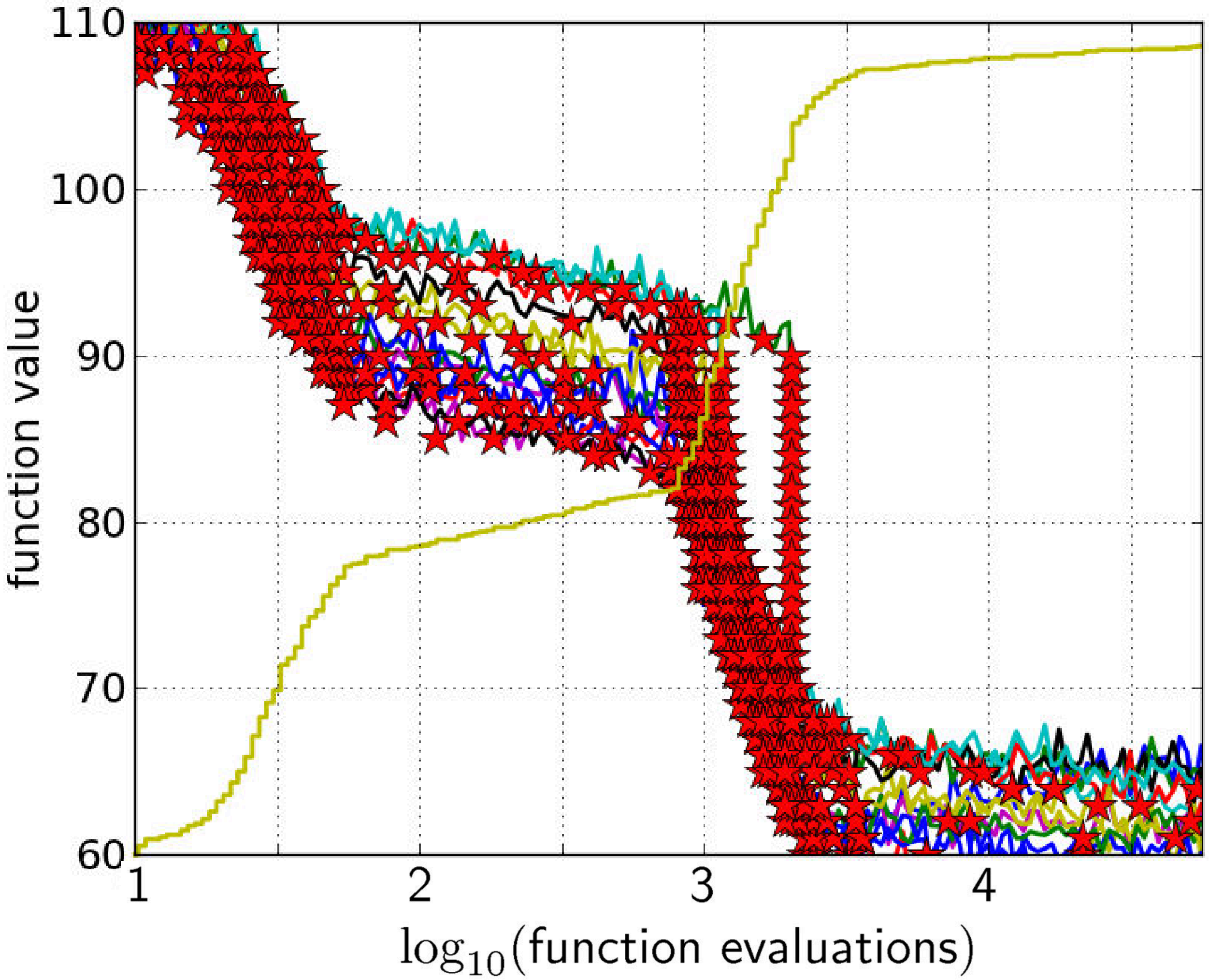
Aggregation



15 runs

50 targets

Aggregation

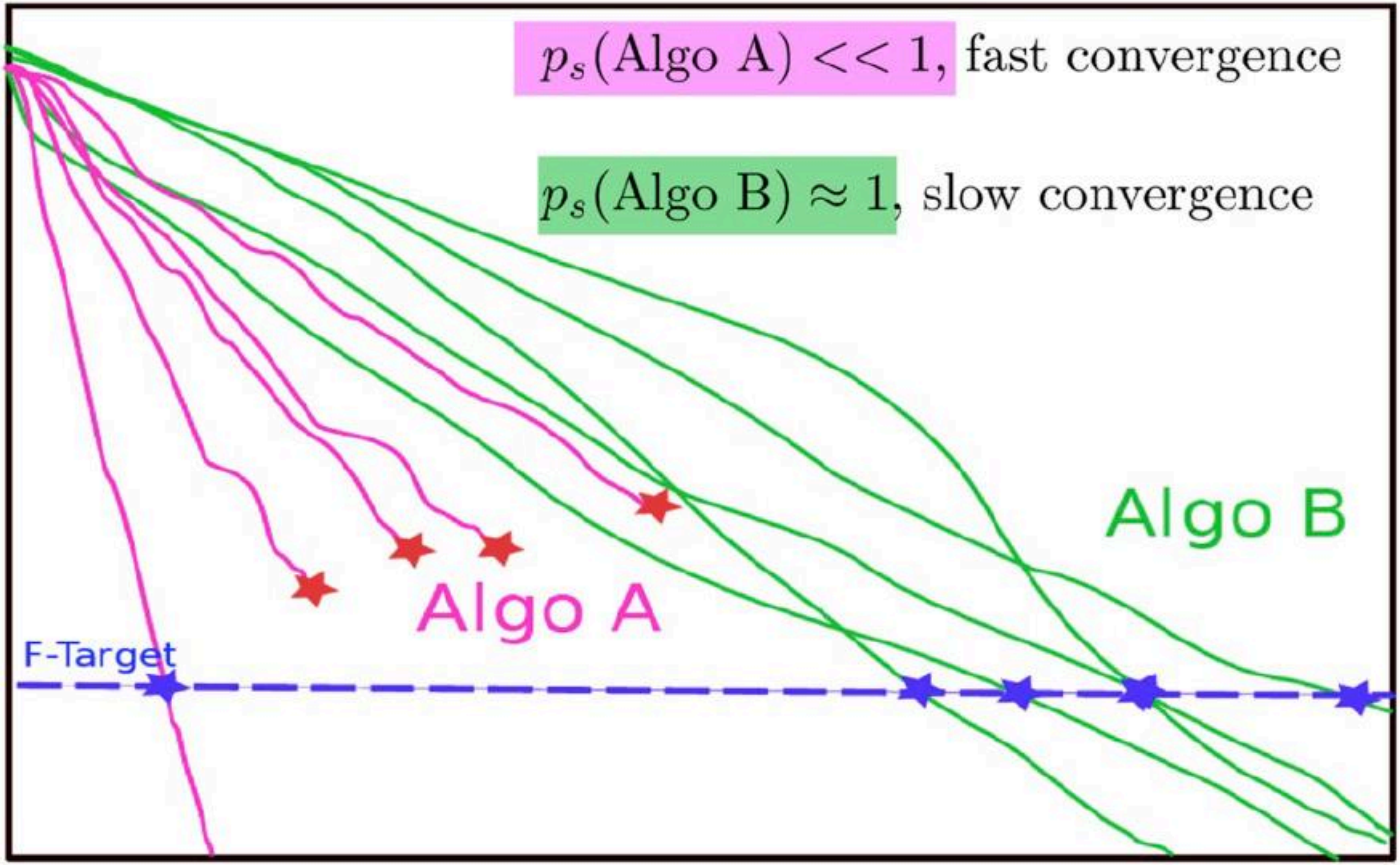


15 runs

50 targets

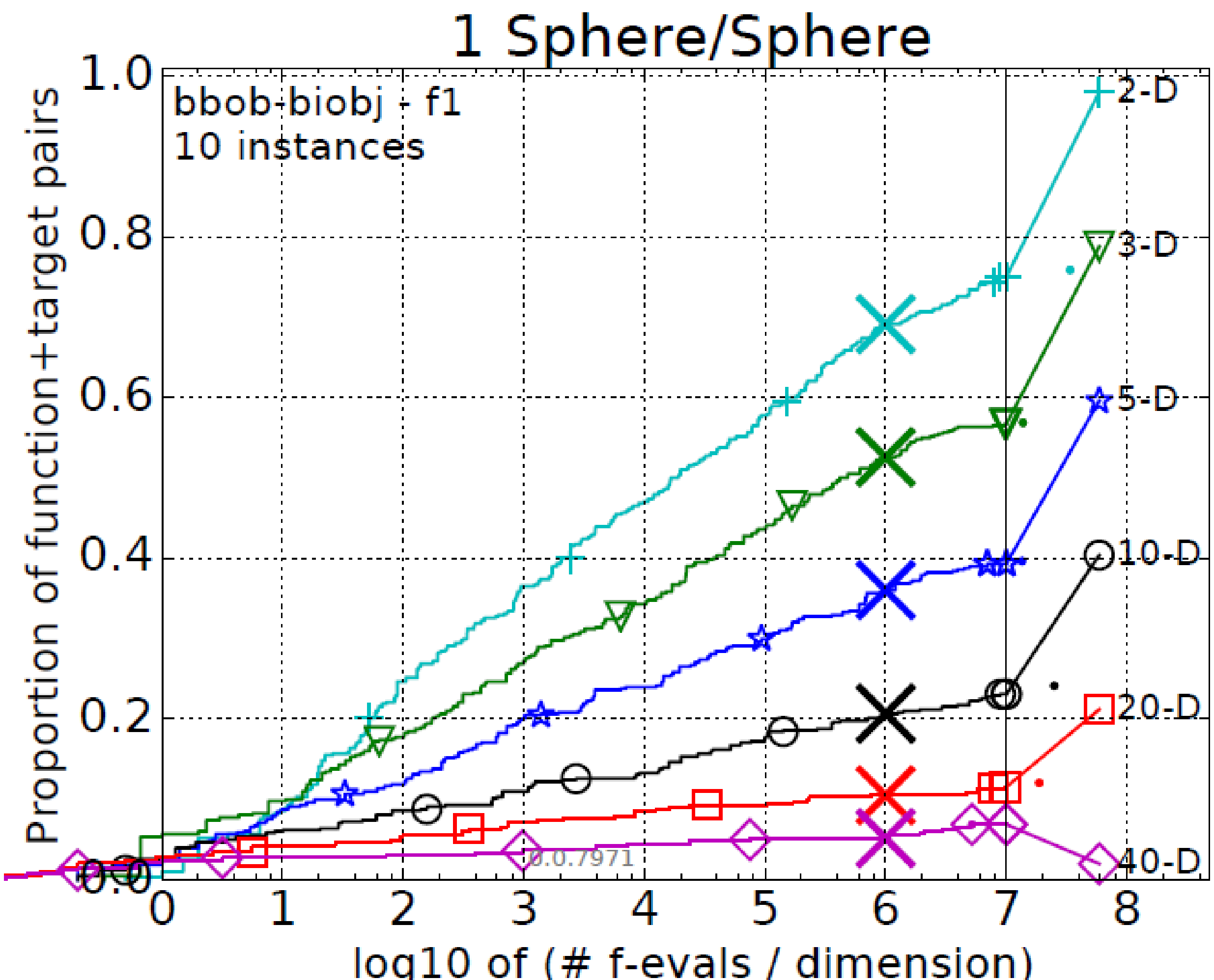
ECDF with 750 steps

Fixed-target: Simulated Restarts



ECDFs with Simulated Restarts

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



Exercise

Objective:

investigate the performance of algorithms, available at

http://coco.gforge.inria.fr/doku.php?id=algorithms

56	IPOP-CMA-ES: CMA-ES with restarts and increasing popsize
5	BIPOP-CMA-ES: two different popsize regimes
22	Nelder-Mead simplex (use better "NelderDoerr" version here)
4	BFGS: quasi-Newton
14	Genetic Algorithm: discretization of cont. variables ("GA")
25	ONEFIFTH: (1+1)-ES with 1/5 rule

postprocess (now) and investigate the data (after a few more slides)

tip: use **`--omit-single`** option to save time

The single-objective BBOB functions

The bbob Testbed

- 24 functions in 5 groups:

1 Separable Functions	
f1	Sphere Function
f2	Ellipsoidal Function
f3	Rastrigin Function
f4	Büche-Rastrigin Function
f5	Linear Slope
2 Functions with low or moderate conditioning	
f6	Attractive Sector Function
f7	Step Ellipsoidal Function
f8	Rosenbrock Function, original
f9	Rosenbrock Function, rotated
3 Functions with high conditioning and unimodal	
f10	Ellipsoidal Function
f11	Discus Function
f12	Bent Cigar Function
f13	Sharp Ridge Function
f14	Different Powers Function

4 Multi-modal functions with adequate global structure	
f15	Rastrigin Function
f16	Weierstrass Function
f17	Schaffers F7 Function
f18	Schaffers F7 Functions, moderately ill-conditioned
f19	Composite Griewank-Rosenbrock Function F8F2
5 Multi-modal functions with weak global structure	
f20	Schwefel Function
f21	Gallagher's Gaussian 101-me Peaks Function
f22	Gallagher's Gaussian 21-hi Peaks Function
f23	Katsuura Function
f24	Lunacek bi-Rastrigin Function

- 6 dimensions: 2, 3, 5, 10, 20, (40 optional)

Notion of Instances

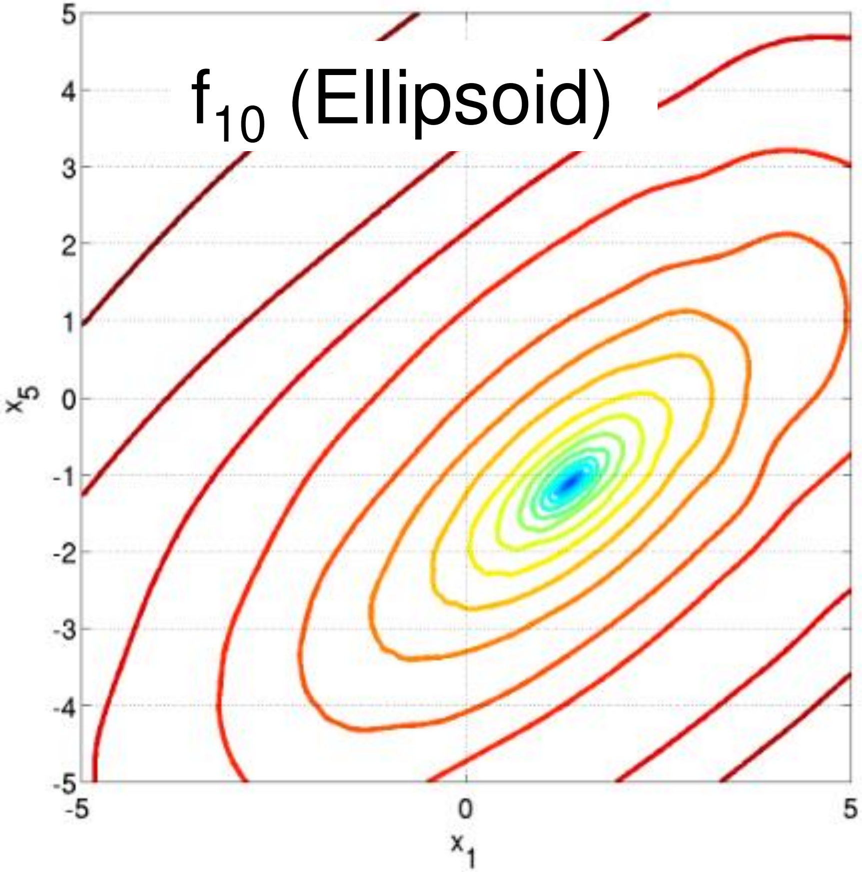
- All COCO problems come in form of instances
  - e.g. as translated/rotated versions of the same function
- Prescribed instances typically change from year to year
  - avoid overfitting
  - 5 instances are always kept the same

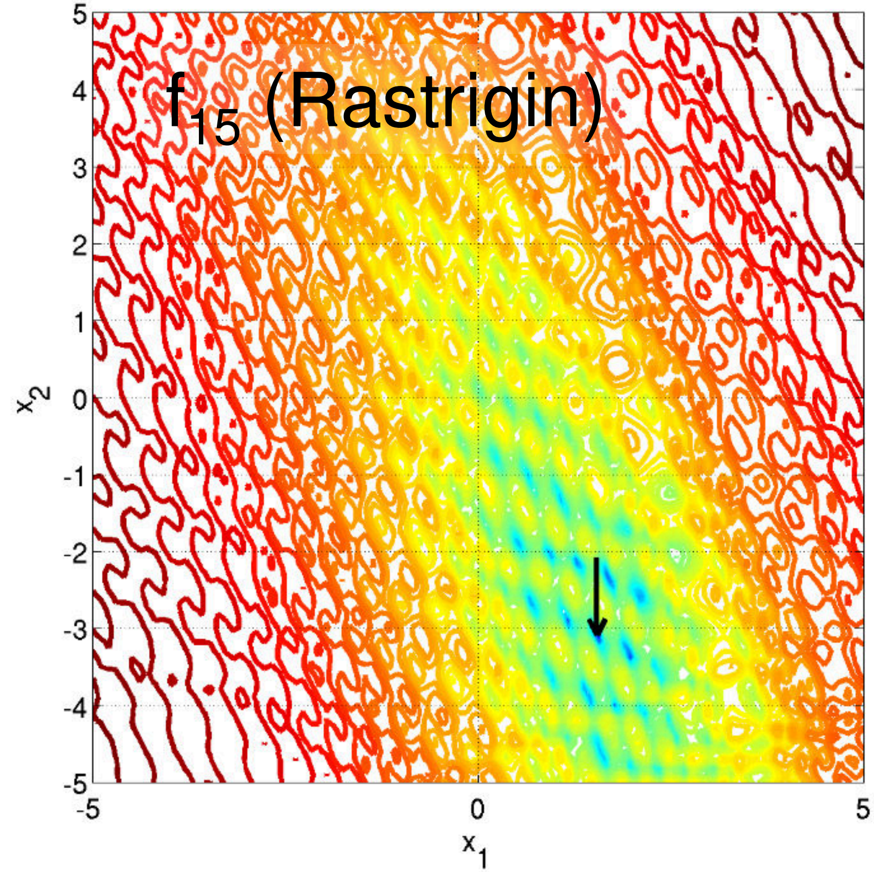
Plus:

  - the bbob functions are locally perturbed by non-linear transformations

Notion of Instances

• All COCO problems come in form of instances





linear transformations

The single-objective noisy BBOB functions

`bbob-noisy` Testbed

- 30 functions with various kinds of noise types and strengths
  - 3 noise types: Gaussian, uniform, and seldom Cauchy
  - Functions with moderate noise
  - Functions with severe noise
  - Highly multi-modal functions with severe noise
  - `bbob` functions included: Sphere, Rosenbrock, Step ellipsoid, Ellipsoid, Different Powers, Schaffers' F7, Composite Griewank-Rosenbrock
- 6 dimensions: 2, 3, 5, 10, 20, (40 optional)

COCO extended to multiobjective optimization

bbob-biobj Testbed

- 55 functions by combining 2 bbob functions

1 Separable Functions	
f1	Sphere Function ✓
f2	Ellipsoidal Function ✓
f3	Rastrigin Function
f4	Büche-Rastrigin Function
f5	Linear Slope
2 Functions with low or moderate conditioning	
f6	Attractive Sector Function ✓
f7	Step Ellipsoidal Function
f8	Rosenbrock Function, original ✓
f9	Rosenbrock Function, rotated
3 Functions with high conditioning and unimodal	
f10	Ellipsoidal Function
f11	Discus Function
f12	Bent Cigar Function
f13	Sharp Ridge Function ✓
f14	Different Powers Function ✓

4 Multi-modal functions with adequate global structure	
f15	Rastrigin Function ✓
f16	Weierstrass Function
f17	Schaffers F7 Function ✓
f18	Schaffers F7 Functions, moderately ill-conditioned
f19	Composite Griewank-Rosenbrock Function F8F2
5 Multi-modal functions with weak global structure	
f20	Schwefel Function ✓
f21	Gallagher's Gaussian 101-me Peaks Function ✓
f22	Gallagher's Gaussian 21-hi Peaks Function
f23	Katsuura Function
f24	Lunacek bi-Rastrigin Function

bbob-biobj Testbed

- 55 functions by combining 2 **bbob** functions

1 Separable Functions	
f1	Sphere Function ✓
f2	Ellipsoidal Function ✓
f3	Rastrigin Function
f4	Büche-Rastrigin Function
f5	Linear Slope
2 Functions with low or moderate conditionir	
f6	Attractive Sector Function ✓
f7	Step Ellipsoidal Function
f8	Rosenbrock Function, original ✓
f9	Rosenbrock Function, rotated
3 Functions with high conditioning and unimo	
f10	Ellipsoidal Function
f11	Discus Function
f12	Bent Cigar Function
f13	Sharp Ridge Function ✓
f14	Different Powers Function ✓

4 Multi-modal functions with adequate global structure	
f15	Rastrigin Function ✓
f16	Weierstrass Function
f17	Schaffers F7 Function ✓

	f_1	f_2	f_6	f_8	f_{13}	f_{14}	f_{15}	f_{17}	f_{20}	f_{21}
f_1	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10
f_2		f11	f12	f13	f14	f15	f16	f17	f18	f19
f_6			f20	f21	f22	f23	f24	f25	f26	f27
f_8				f28	f29	f30	f31	f32	f33	f34
f_{13}					f35	f36	f37	f38	f39	f40
f_{14}						f41	f42	f43	f44	f45
f_{15}							f46	f47	f48	f49
f_{17}								f50	f51	f52
f_{20}									f53	f54
f_{21}										f55

`bbob-biobj` Testbed

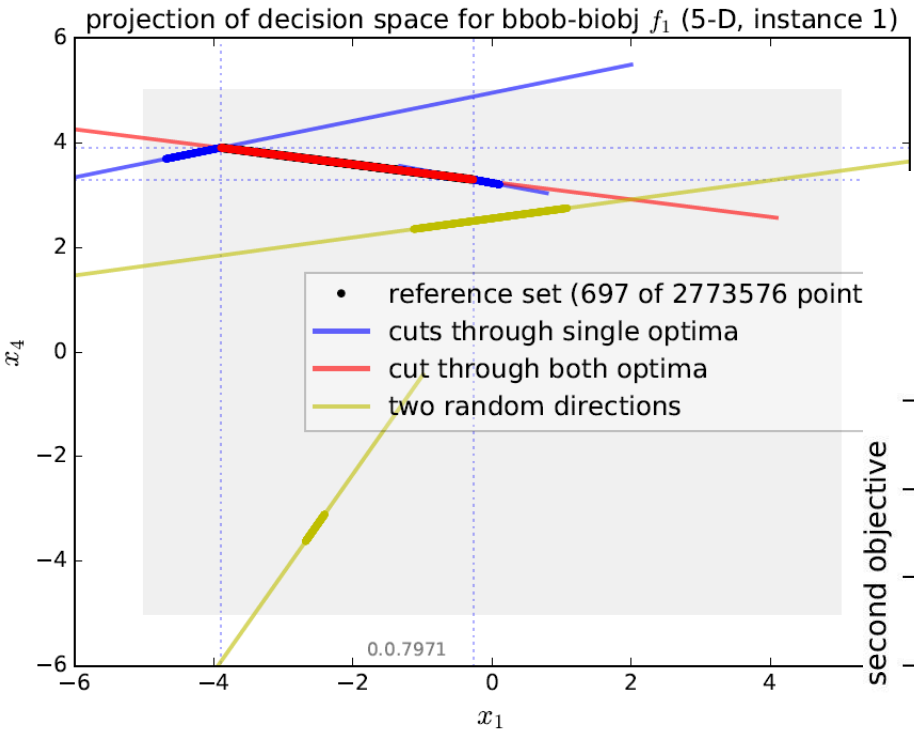
- 55 functions by combining 2 `bbob` functions
- 15 function groups with 3-4 functions each
  - separable – separable, separable – moderate, separable - ill-conditioned, ...
- 6 dimensions: 2, 3, 5, 10, 20, (40 optional)
- instances derived from `bbob` instances:
  - more or less 2i+1 for 1st objective and 2i+2 for 2nd objective
  - exceptions: instances 1 and 2 and when optima are too close
- no normalization (algo has to cope with different orders of magnitude)
- for performance assessment: ideal/nadir points known

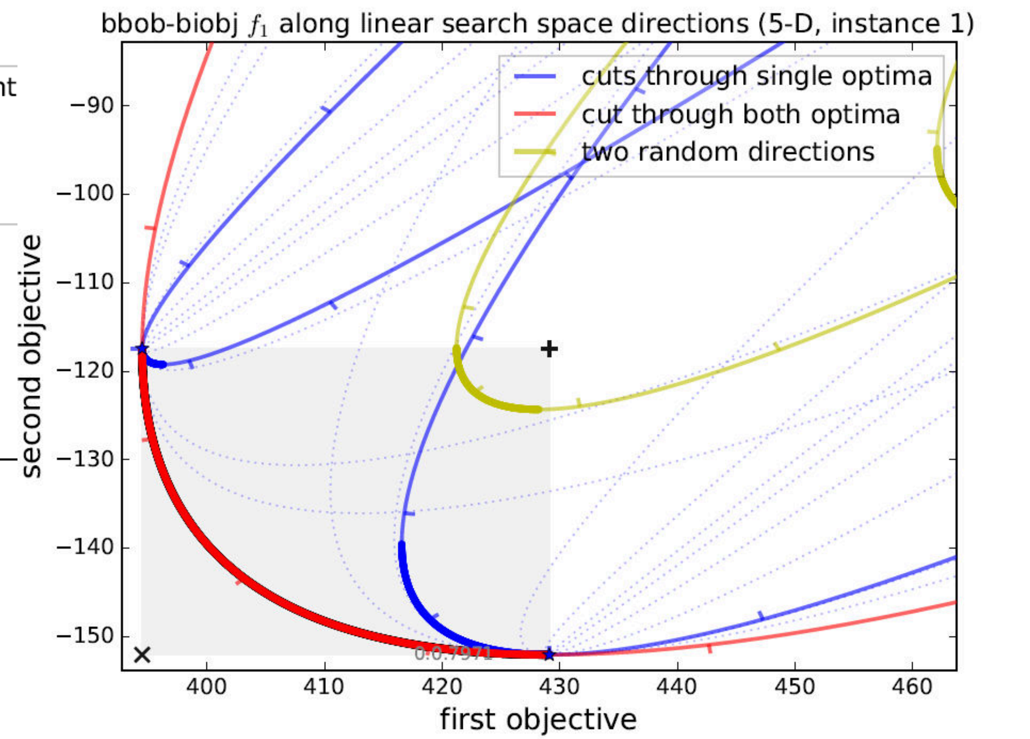
`bbob-biobj` Testbed (cont'd)

- Pareto set and Pareto front unknown
  - but we have a good idea of where they are by running quite some algorithms and keeping track of all non-dominated points found so far
- Various types of shapes

bbob-biobj Testbed (cont'd)

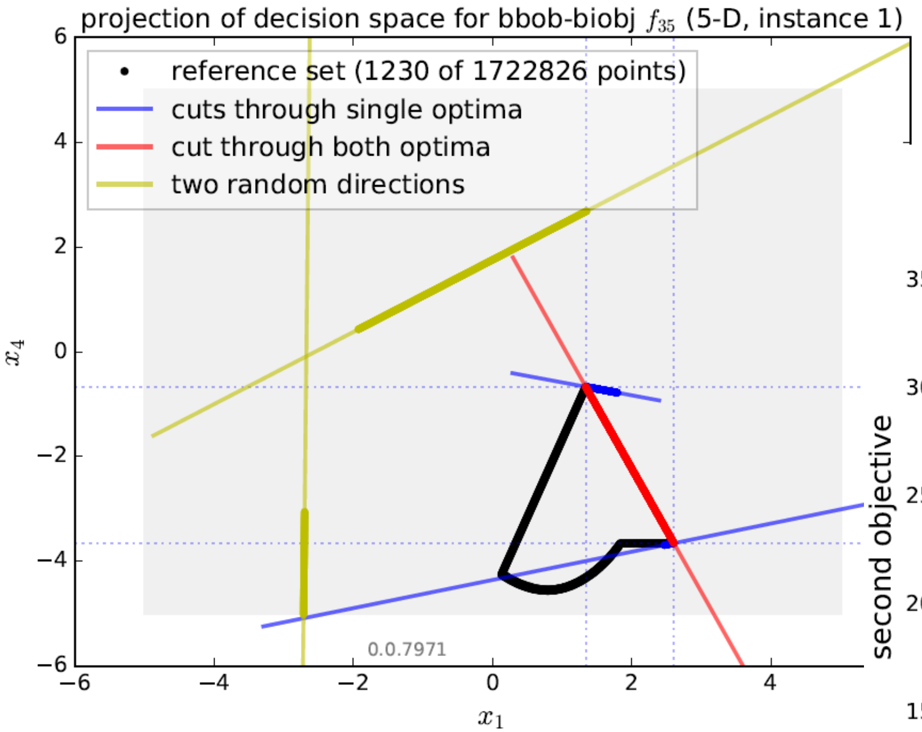
Example: sphere with sphere

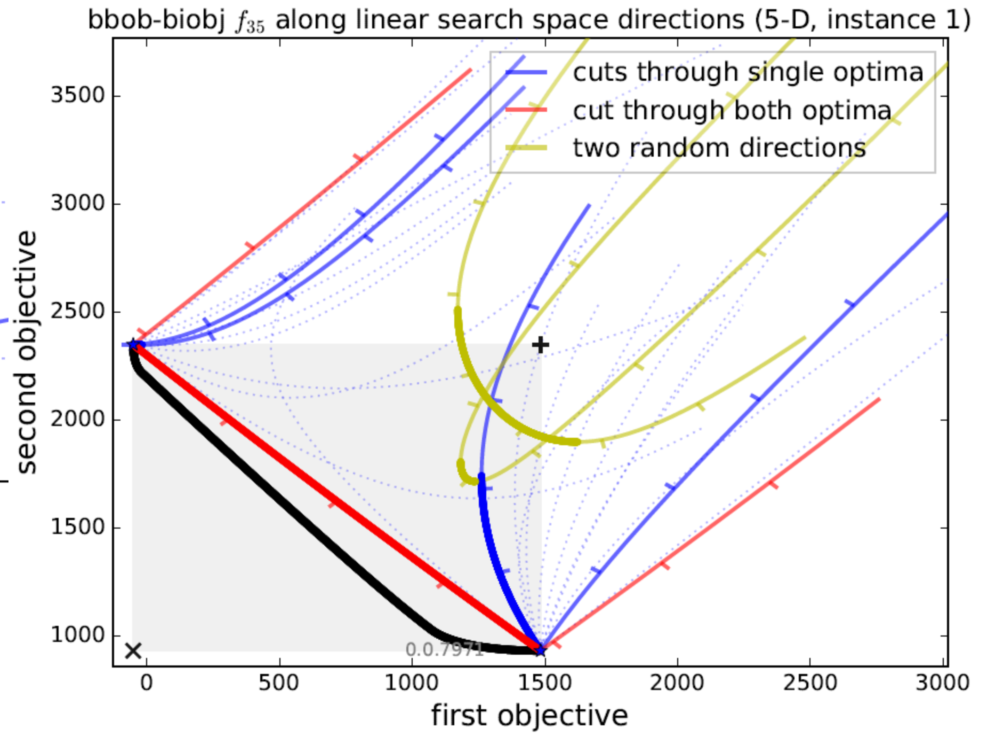




bbob-biobj Testbed (cont'd)

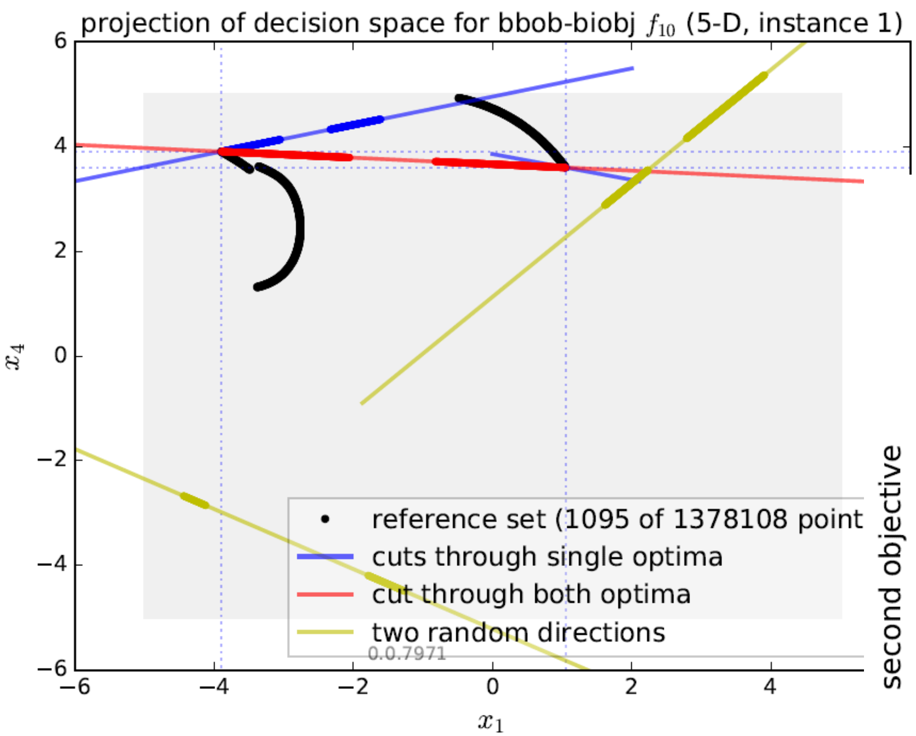
Example: sharp ridge with sharp ridge

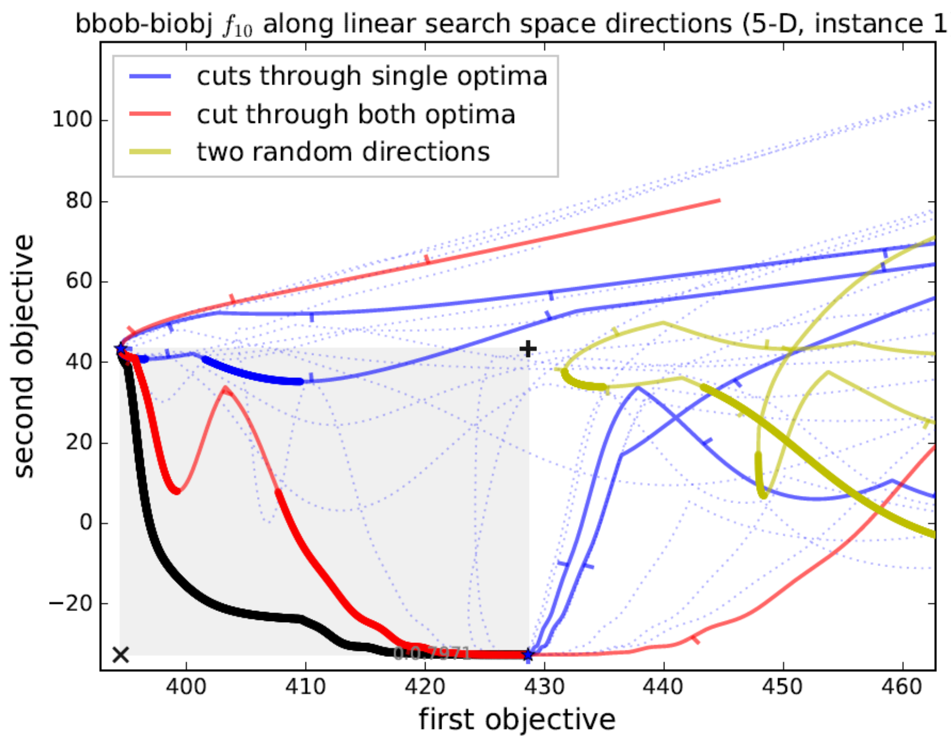




bbob-biobj Testbed (cont'd)

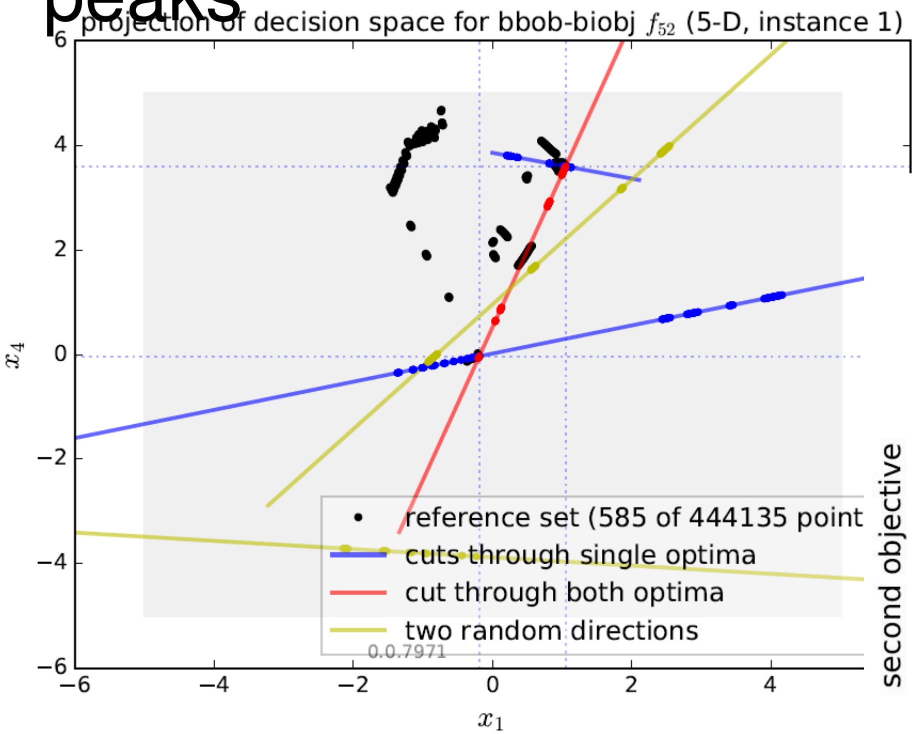
Example: sphere with Gallagher 101 peaks

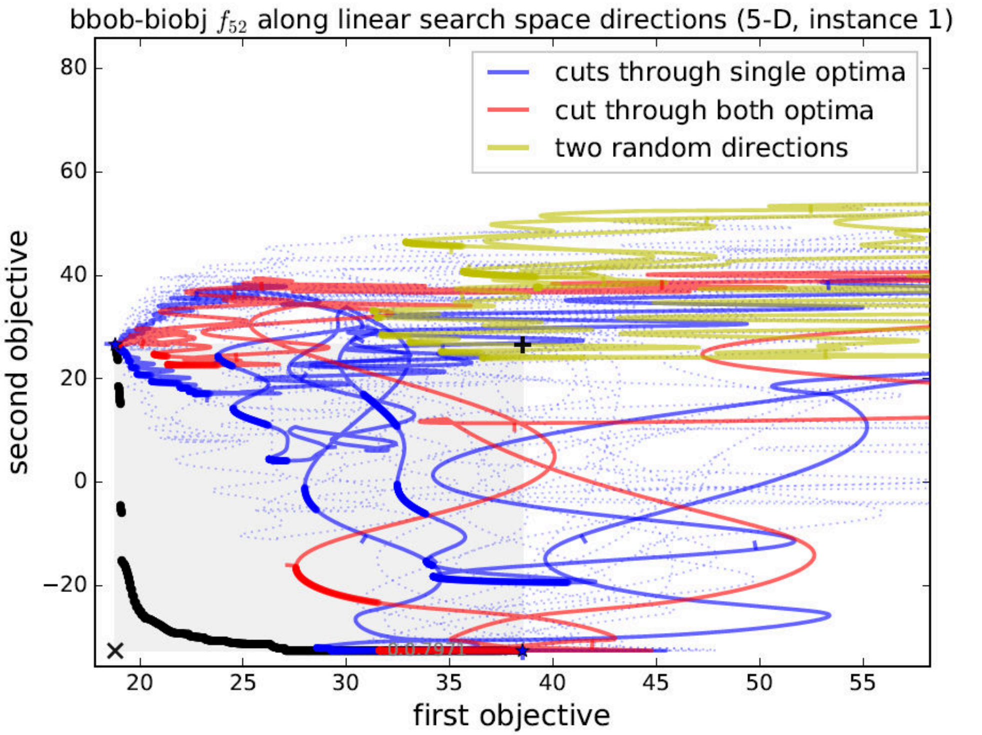




bbob-biobj Testbed (cont'd)

Example: Schaffer F7, cond. 10 with Gallagher 101 peaks





Exercise

Objective:

investigate the data you just postprocessed:

a) which algorithms are the best ones?
b) does this depend on the dimension?
c) look at single graphs: can we say something about the algorithms' invariances, e.g. wrt. rotations of the search space?
d) what's the impact of covariance-matrix-adaptation?
e) what do you think: are the displayed algorithms well-suited for problems with larger dimension?

Paper Discussion:
"Dynamic Search in Fireworks Algorithm"

http://eprints.cs.univie.ac.at/4082/1/PID3181839.pdf

Objectives of the exercise:

❶ Learn how to read an optimization paper critically

in order to be able to do high-quality reviews

but also to get the most information from it for practical purposes

❷ Understand the reasoning behind the COCO concepts

by looking at how others benchmark optimization algorithms