10th GECCO Workshop on Blackbox Optimization Benchmarking (BBOB): Welcome and Introduction to COCO/BBOB

The BBOBies
https://github.com/numbbo/coco

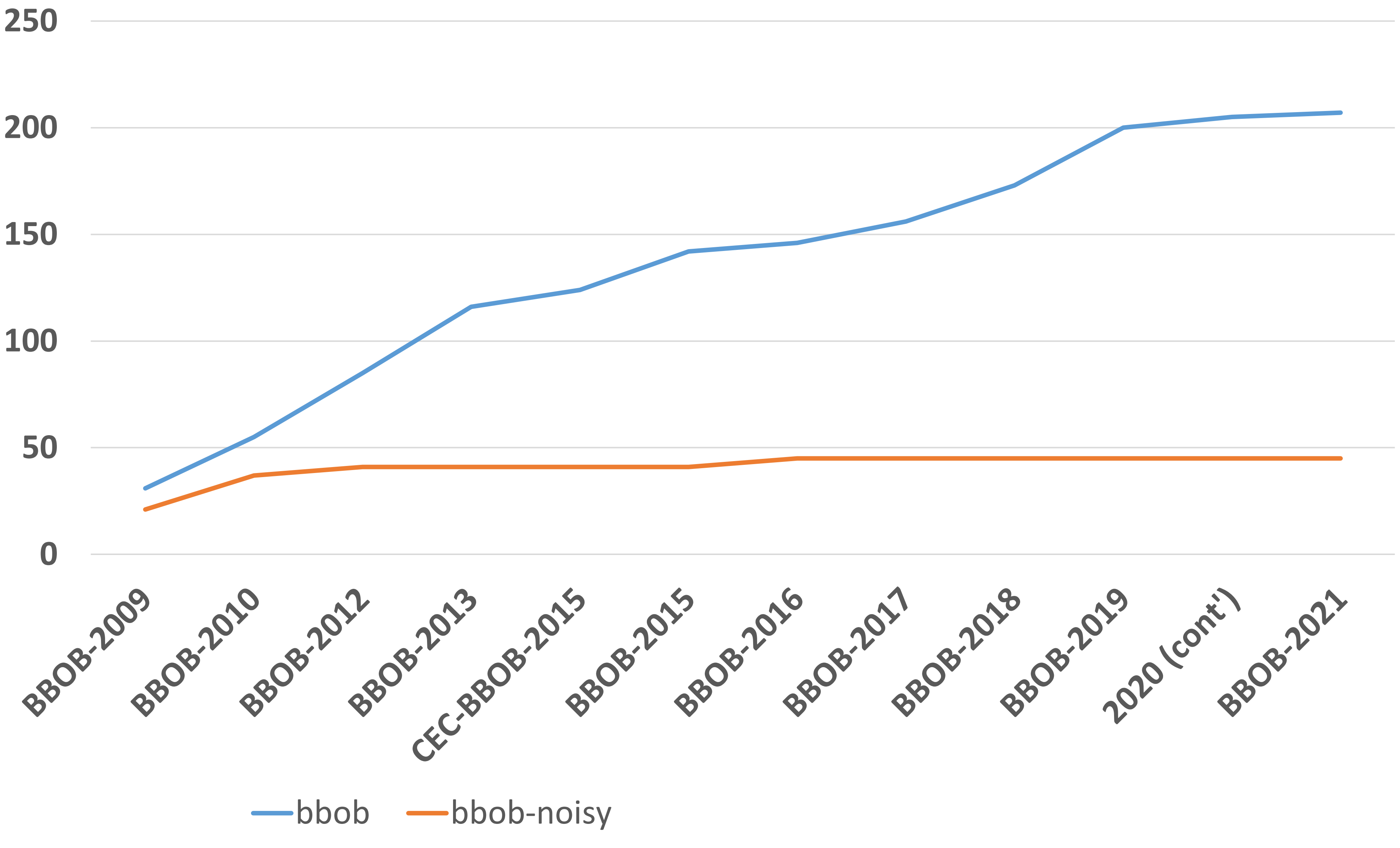
slides based on previous ones by A. Auger, N. Hansen, and D. Brockhoff

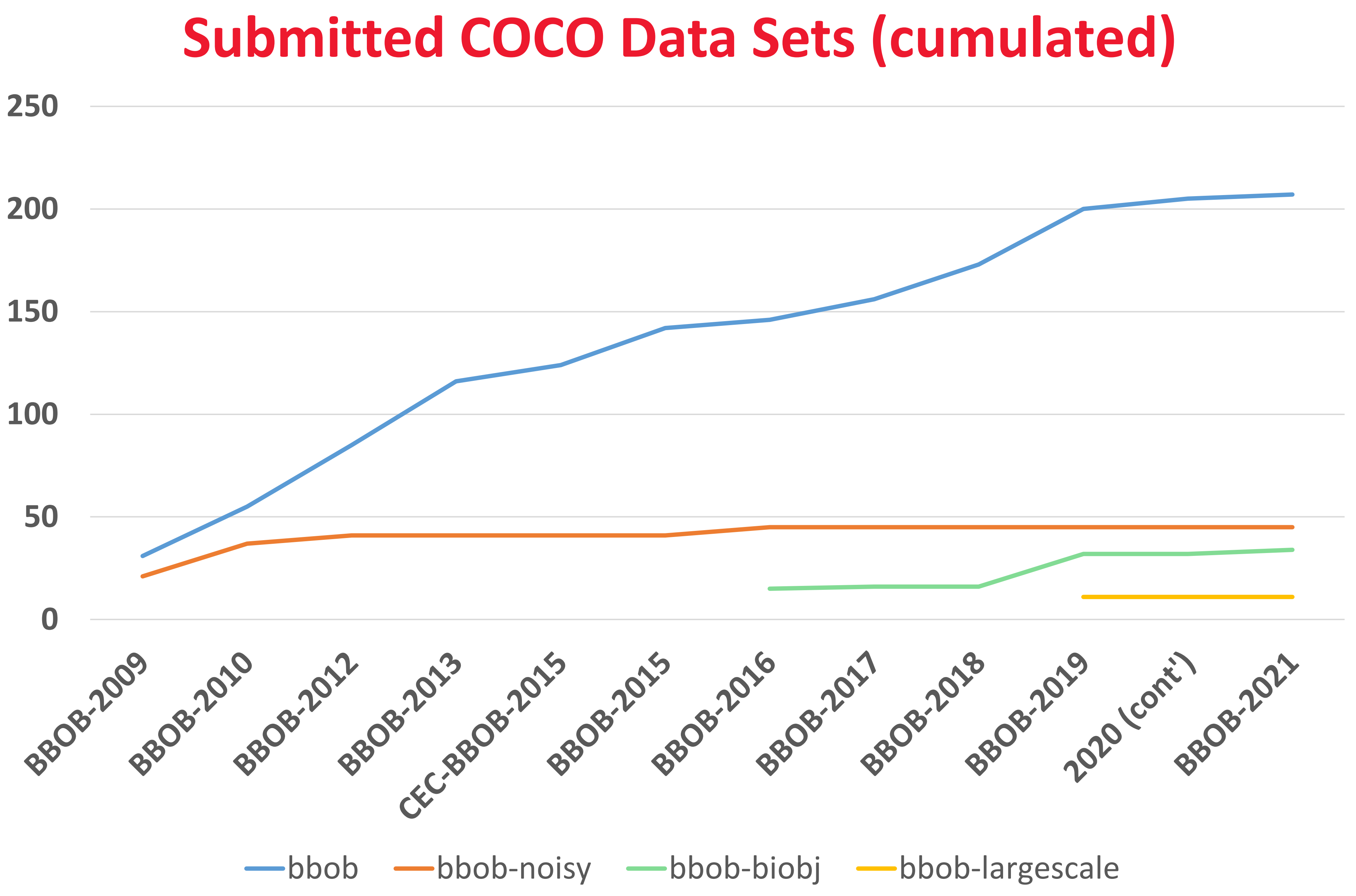
The BBOB Benchmarking Series

- all started at CEC'2005 special session

  organized by P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger and S. Tiwari
- split soon thereafter into the CEC and GECCO "schools"
- CEC: many topics, many test suites, competitions
- BBOB:
  - focus on a few test suites
  - 2 of them unchanged since 2009
  - assisted by the COCO platform: (semi-)automated benchmarking
- recently even more interest in benchmarking

Submitted COCO Data Sets (cumulated)




Submitted COCO Data Sets (cumulated)
250
200
150
100
50
0
BBOB-2009
BBOB-2010
BBOB-2012
BBOB-2013
CEC-BBOB-2015
BBOB-2015
BBOB-2016
BBOB-2017
BBOB-2018
BBOB-2019
2020 (cont')
BBOB-2021
bbob
bbob-noisy
bbob-biobj
bbob-largescale

Numerical Blackbox Optimization

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

derivatives not available or not useful

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

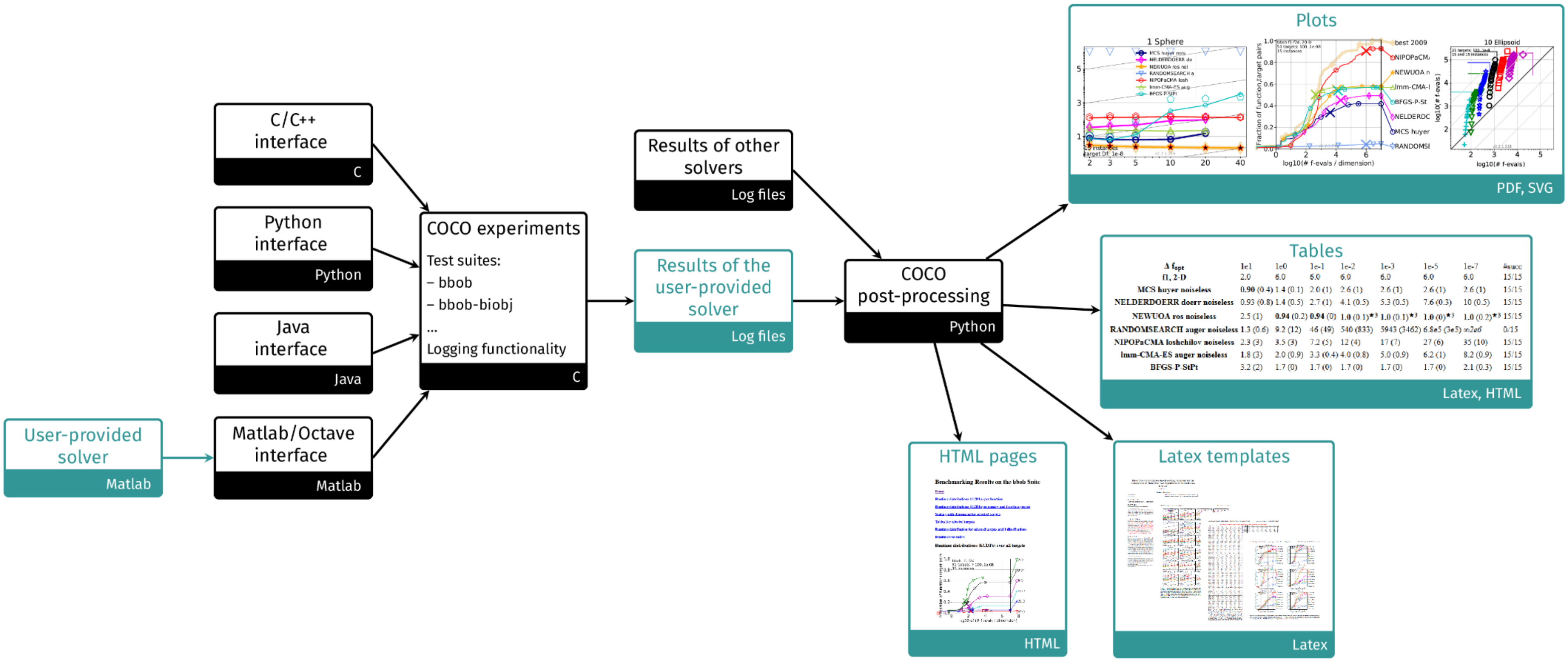
that's where COCO and BBOB come into play

Comparing Continuous Optimizers Platform

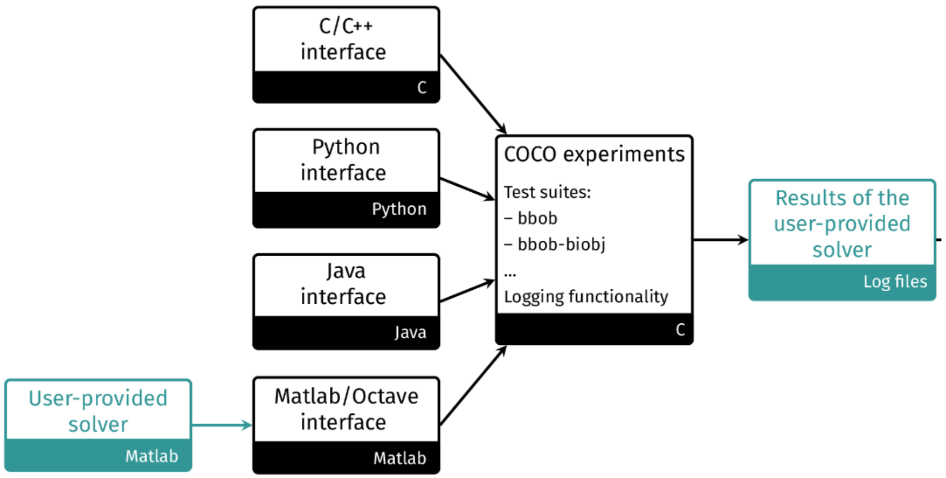
`https://github.com/numbbo/coco`

automatized benchmarking

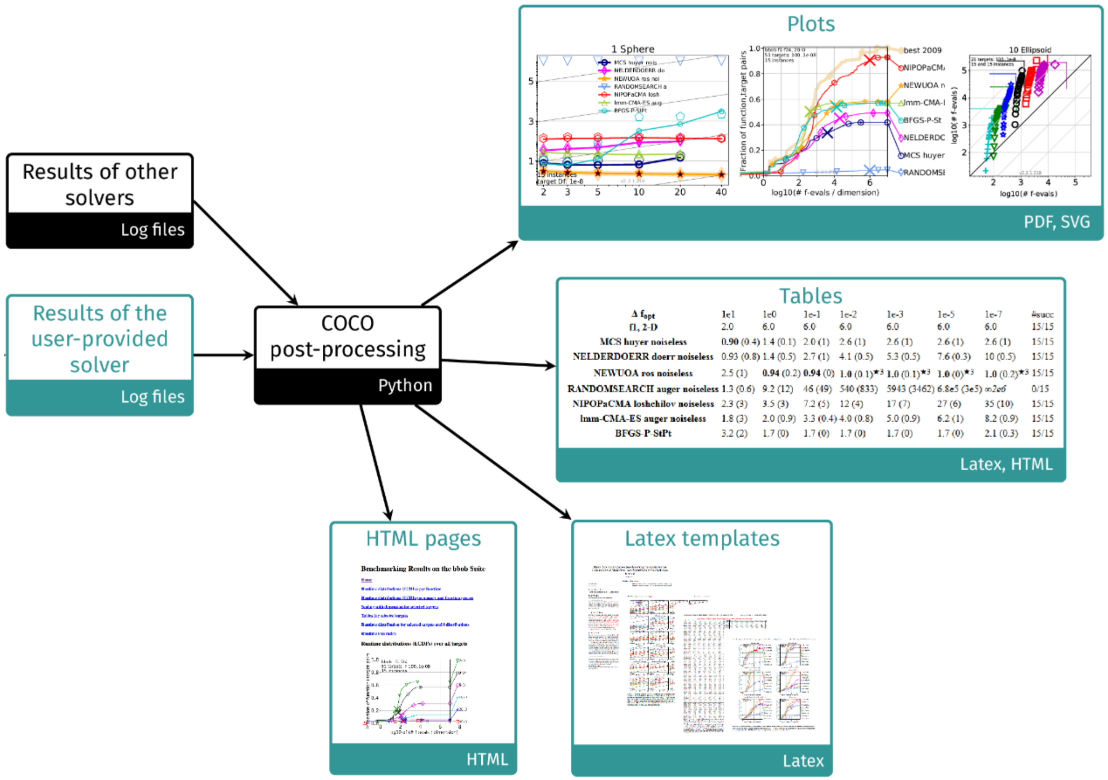
Overview of COCO's Structure



Overview of COCO's Structure



Overview of COCO's Structure



Is Benchmarking Not Trivial?

❶ Choose a set of algorithms

❷ Choose a set of (test) functions

❸ Run the algorithms and compare the results!

the devil is in the details…

hence, COCO implements a
reasonable, well-founded, and
well-documented
pre-chosen methodology

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

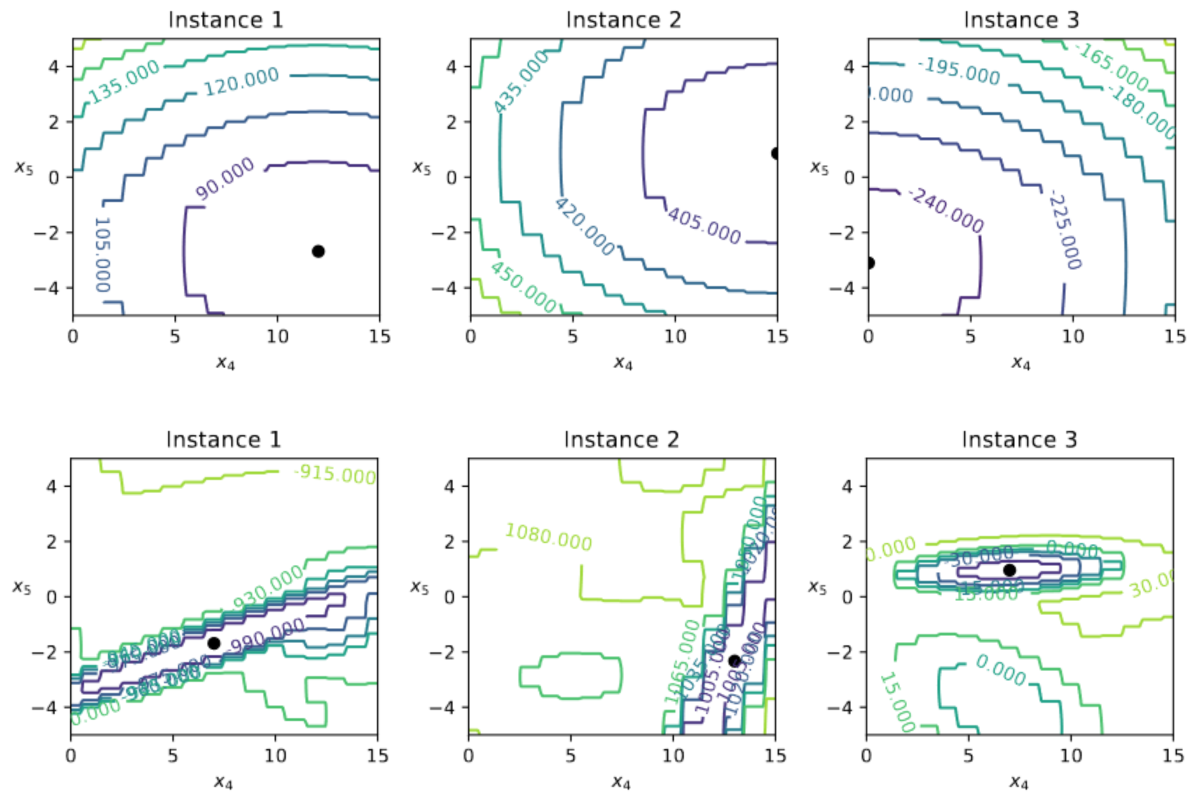
prediction of performance is based on “similarity”,
ideally equivalence classes of functions

Notion of Instances

- All COCO problems come in form of instances
  - e.g. as translated/rotated versions of the same function

Notion of Instances

- All COCO problems come in form of instances
  - e.g. as translated/rotated versions of the same function



Notion of Instances

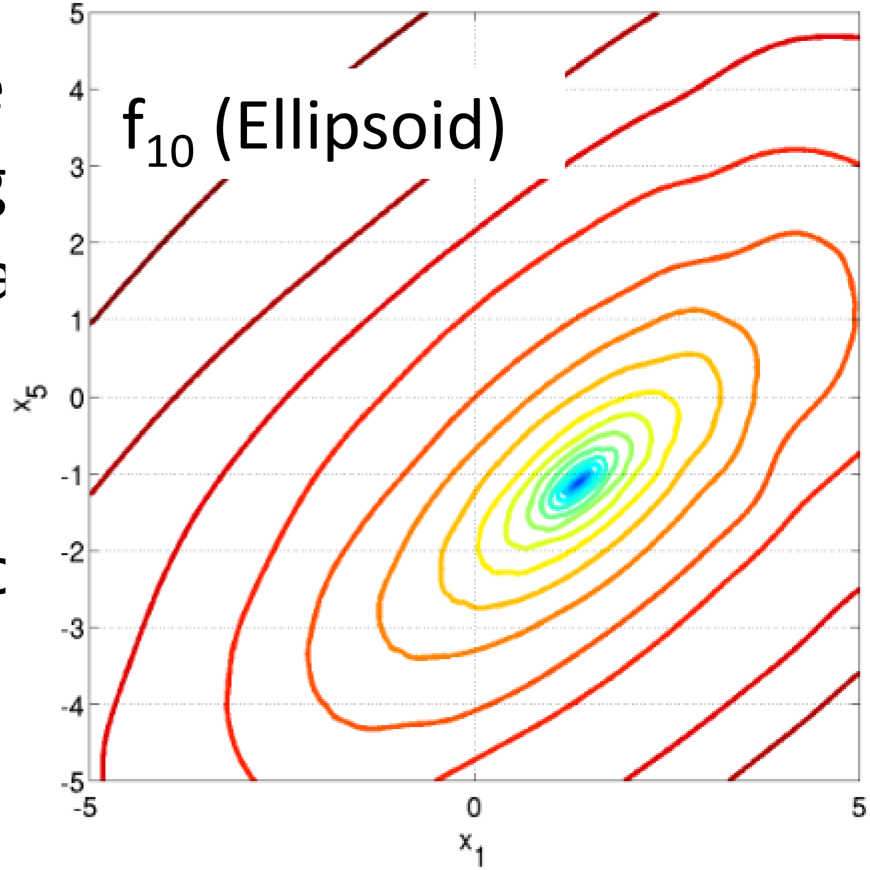
- All COCO problems come in form of instances
  - e.g. as translated/rotated versions of the same function
- Prescribed instances typically change from year to year
  - avoid overfitting
  - 5 instances are always kept the same

Plus:

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

Notion of Instances

- All COCO problems come in form of instances
  - e.g. as translated/rotated ver f_{15} (Rastrigin) ne function
- Prescribed instance ɔ year
  - avoid overfitting
  - 5 instances are a



Plus:

- the bbob functic ɔn-linear transformations

Available Test Suites in COCO

bbob (2009–)	24 noiseless fcts	200+ data sets
bbob-noisy (2009–)	30 noisy fcts	40+ data sets
bbob-biobj (2016–)	55 bi-objective fcts	30+ data sets
bbob-largescale (2019–)	24 noiseless fcts	11 data sets
bbob-mixint (2019–)	24 noiseless fcts	4 data sets
bbob-biobj-mixint (2019–)	92 bi-objective fcts	-

Easy Data Access

```
import cocopp
cocopp.main('BIPOP')

[…]
ValueError: 'BIPOP' has multiple matches in the data
archive:
   2009/BIPOP-CMA-ES_hansen_noiseless.tgz
   2012/BIPOPaCMA_loshchilov_noiseless.tgz
   […]
   2017/KL-BIPOP-CMA-ES-Yamaguchi.tgz
Either pick a single match, or use the `get_all` or
`get_first` method,
or use the ! (first) or * (all) marker and try again.

cocopp.main('BIPOP! BFGS! SLSQP-11')
```

[data access also available via command line]

How Do We Measure Performance?

Meaningful quantitative measure

- quantitative on the ratio scale (highest possible)

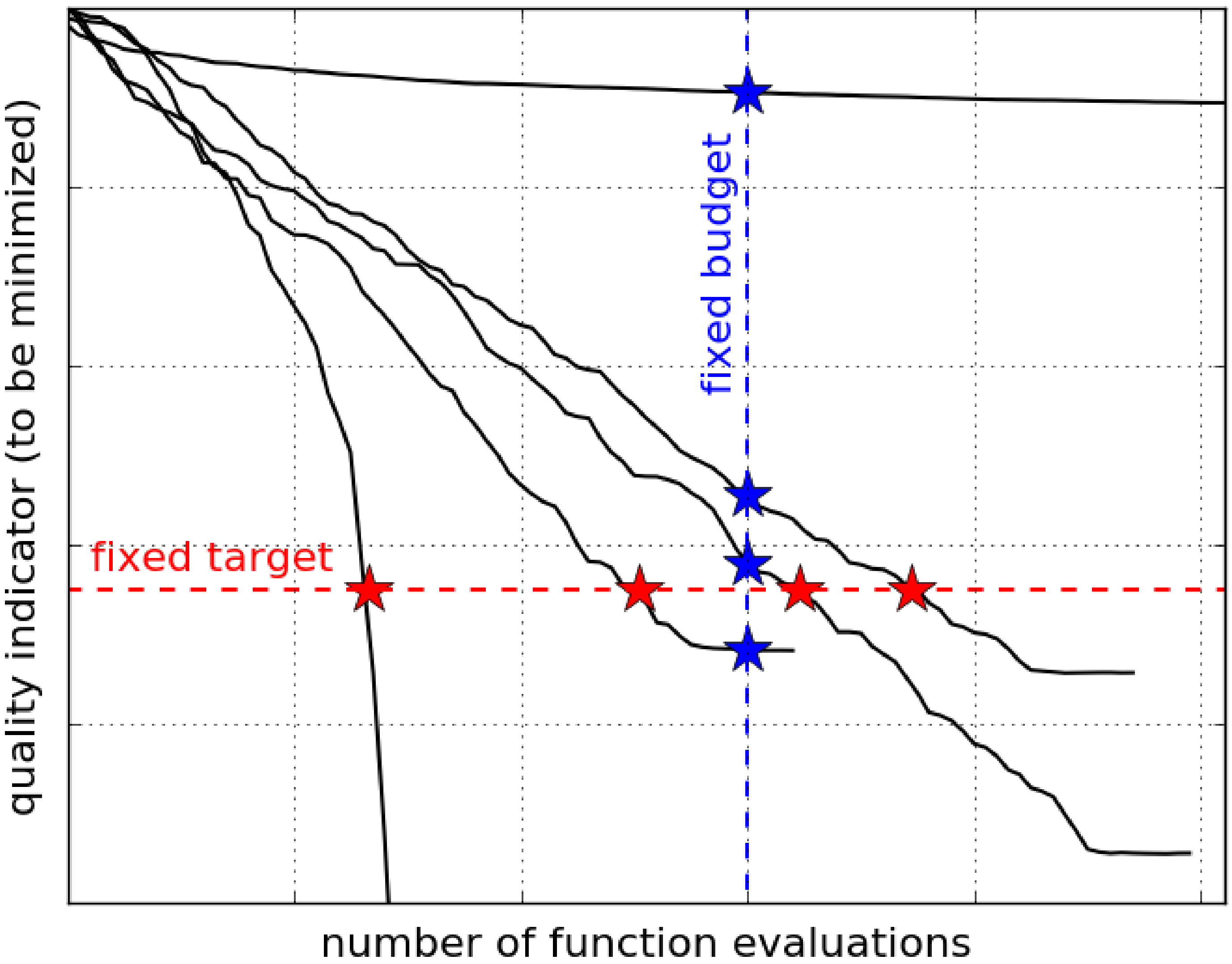
  "algo A is two *times* better than algo B" is a meaningful statement
- assume a wide range of values
- meaningful (interpretable) with regard to the real world

  possible to transfer from benchmarking to real world

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

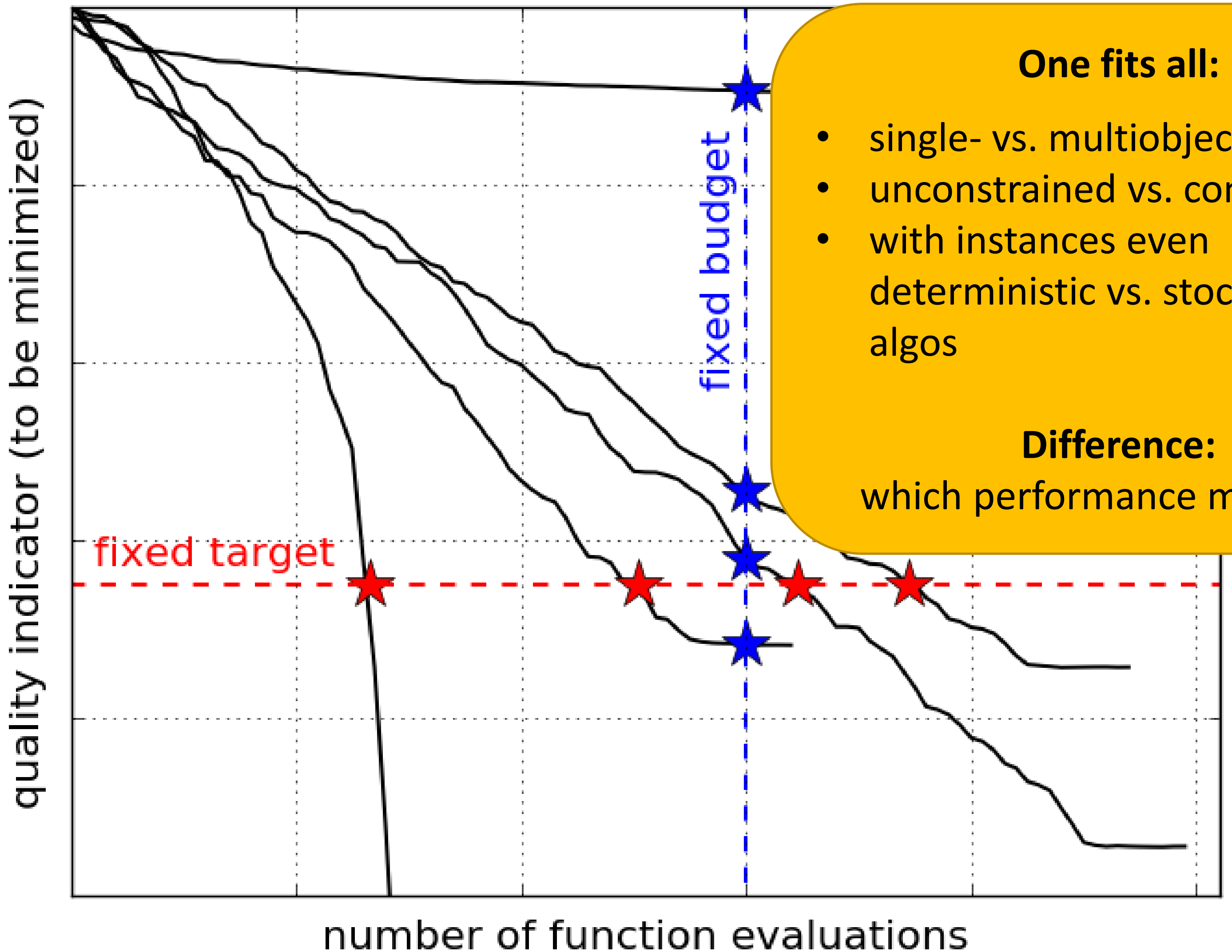
Measuring Performance Empirically

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



Measuring Performance Empirically

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



One fits all:

- single- vs. multiobjective
- unconstrained vs. constrained
- with instances even deterministic vs. stochastic algos

Difference:

which performance measure?

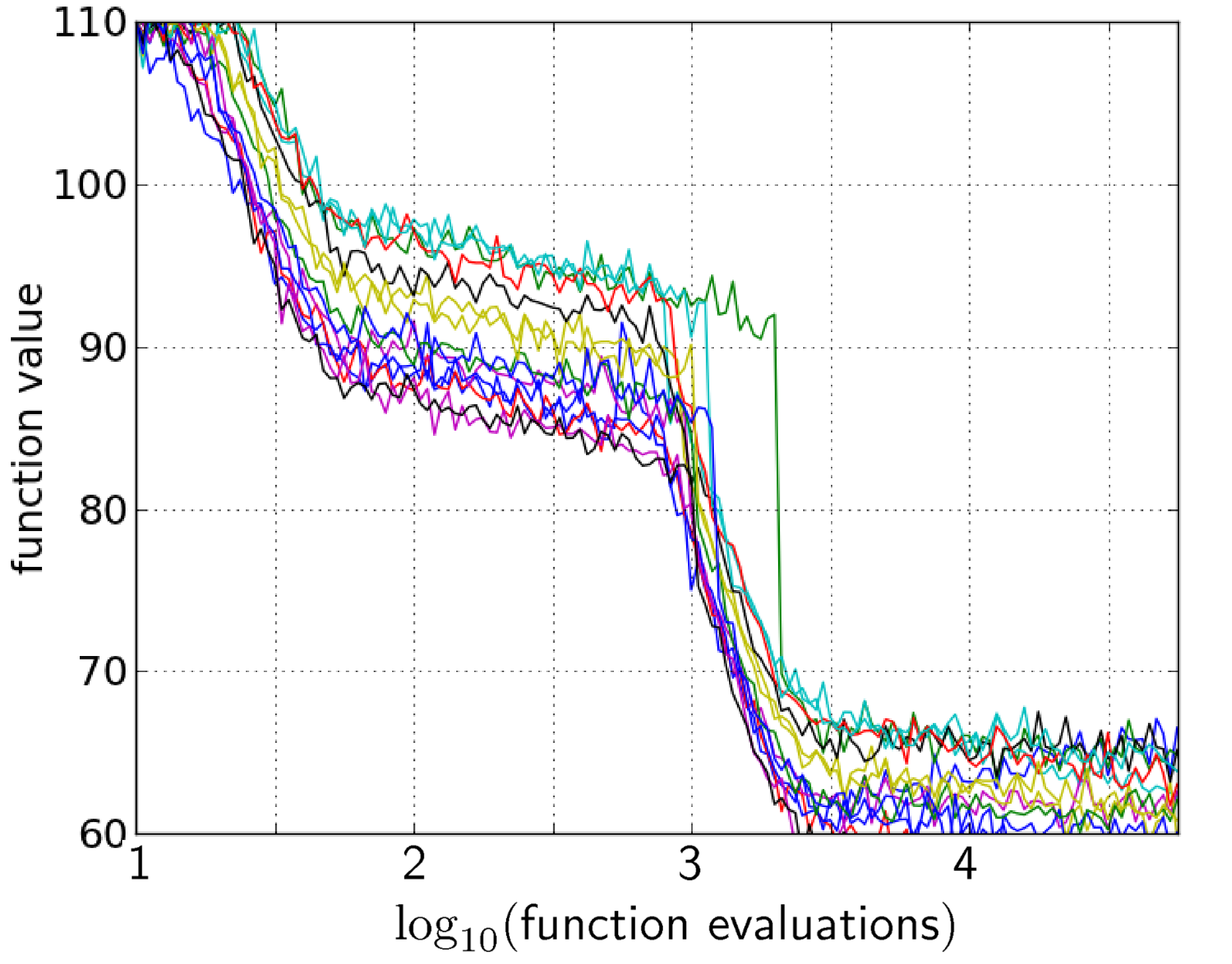
Main Performance Visualization:

Empirical Runtime Distributions

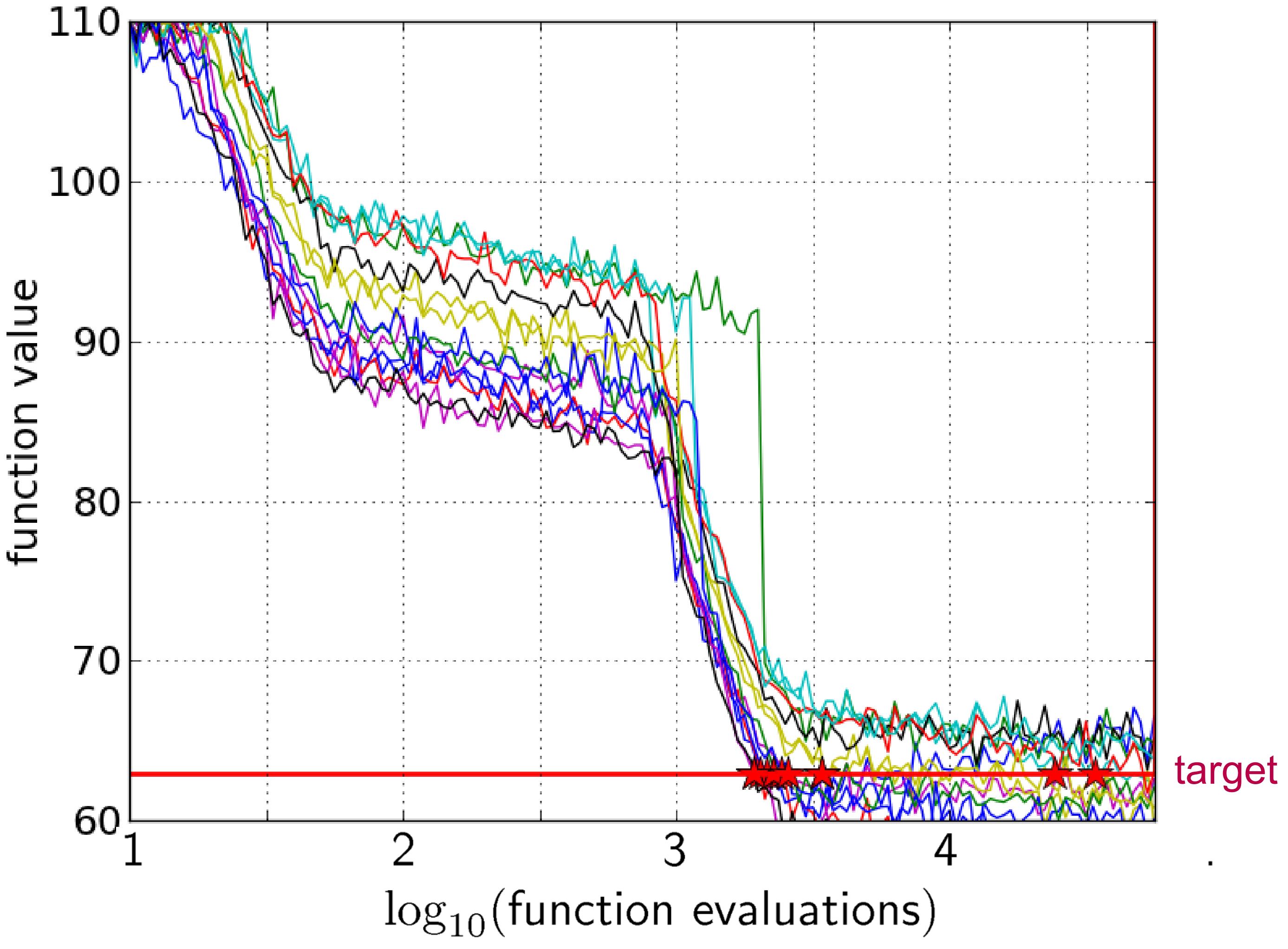
[aka Empirical Cumulative Distribution Function (ECDF) of the Runtime]

[similar to data profiles]

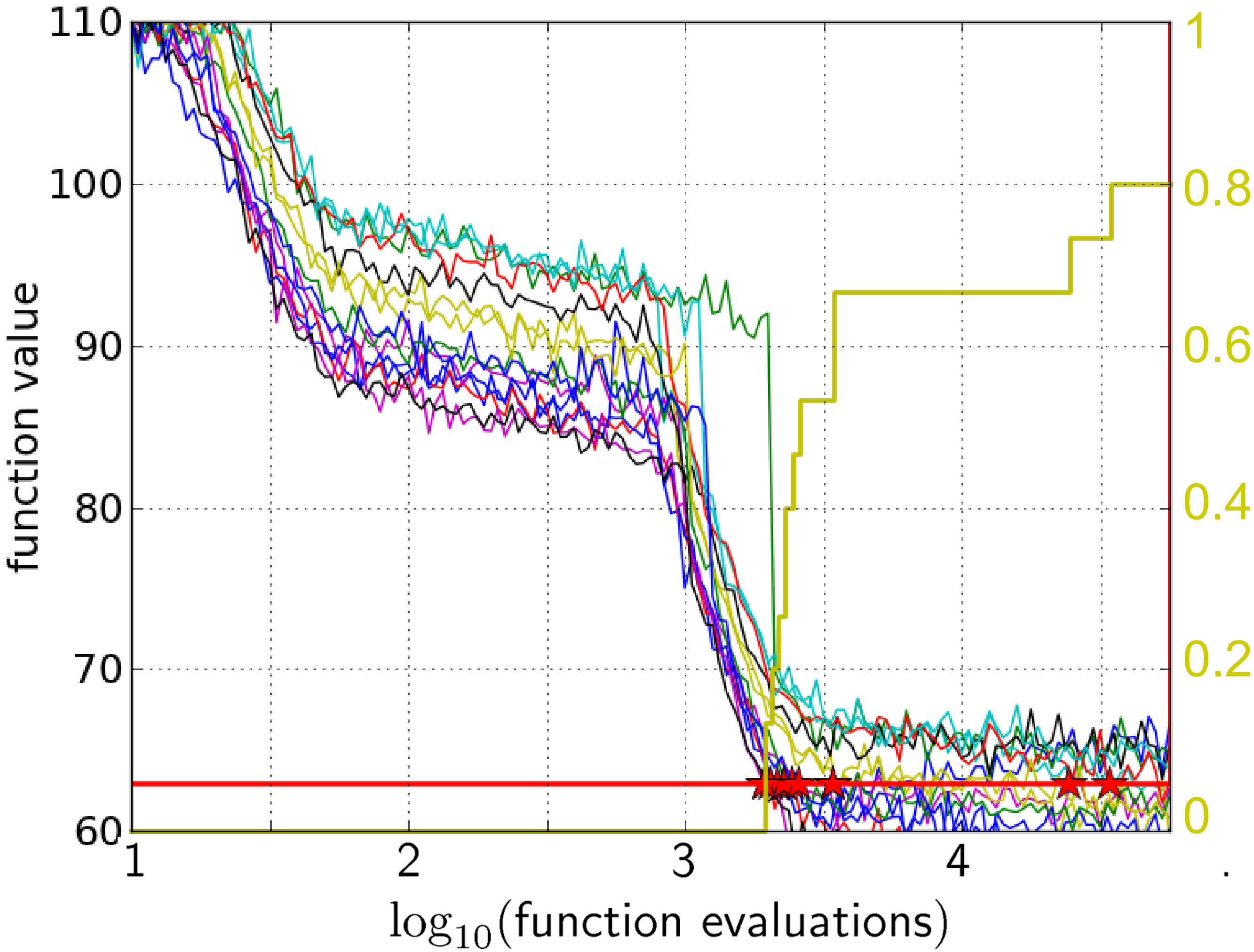
Convergence Graph of 15 Runs



15 Runs ≤ 15 Runtime Data Points



Empirical Cumulative Distribution

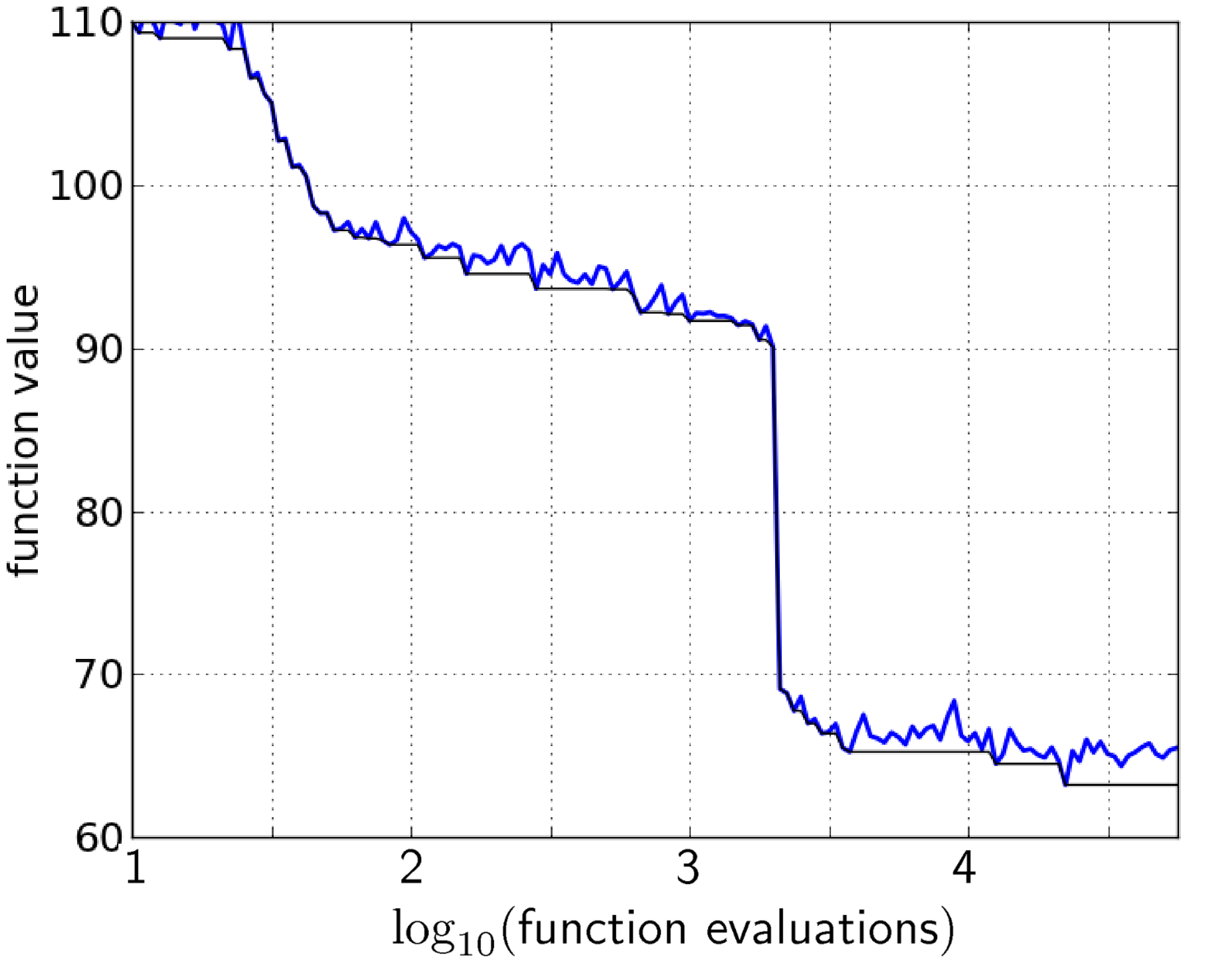


the ECDF of run lengths to reach the target

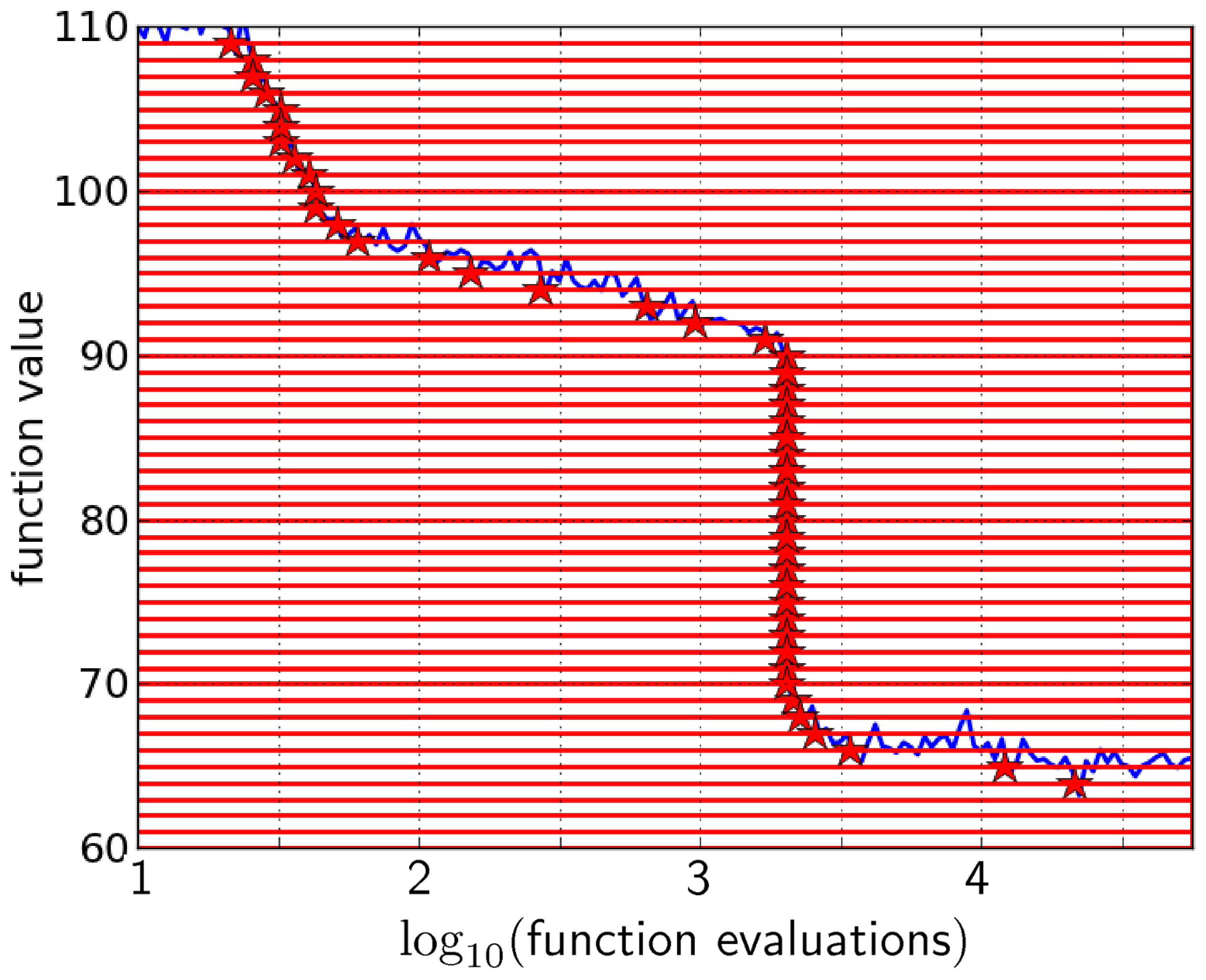
- has for each data point a vertical step of constant size
- displays for each x-value (budget) the count of observations to the left (first hitting times)

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

Reconstructing A Single Run

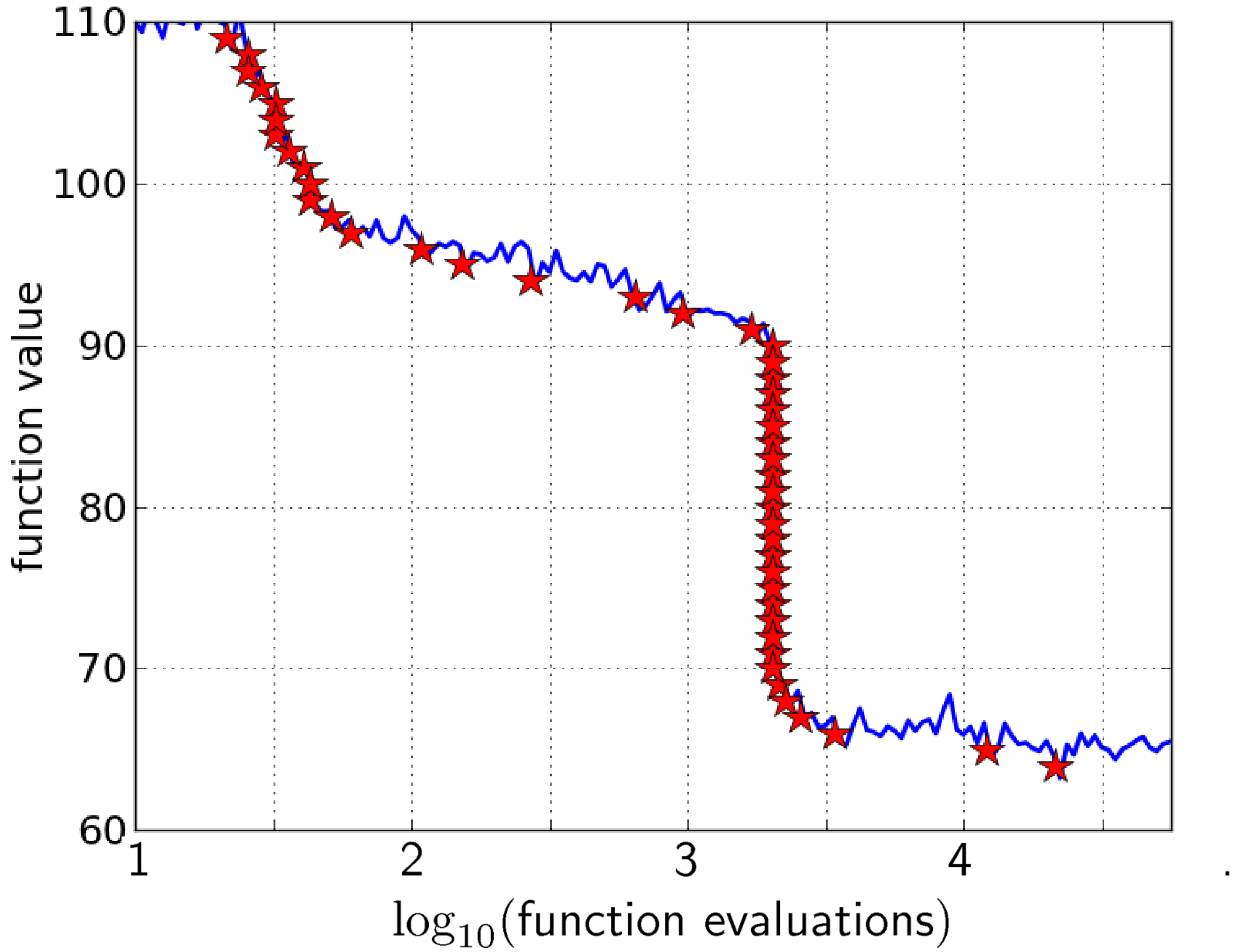


Reconstructing A Single Run

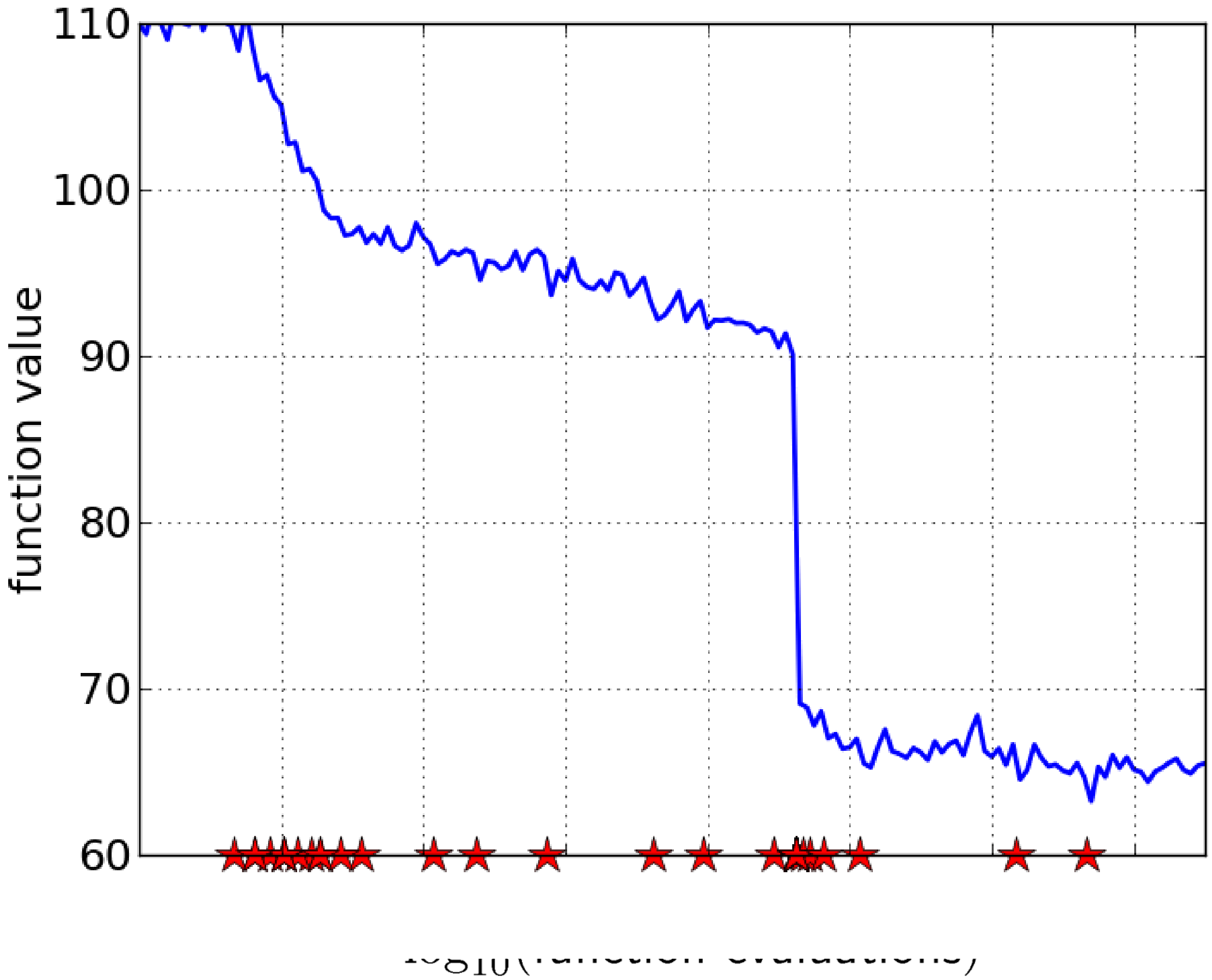


50 equally spaced targets

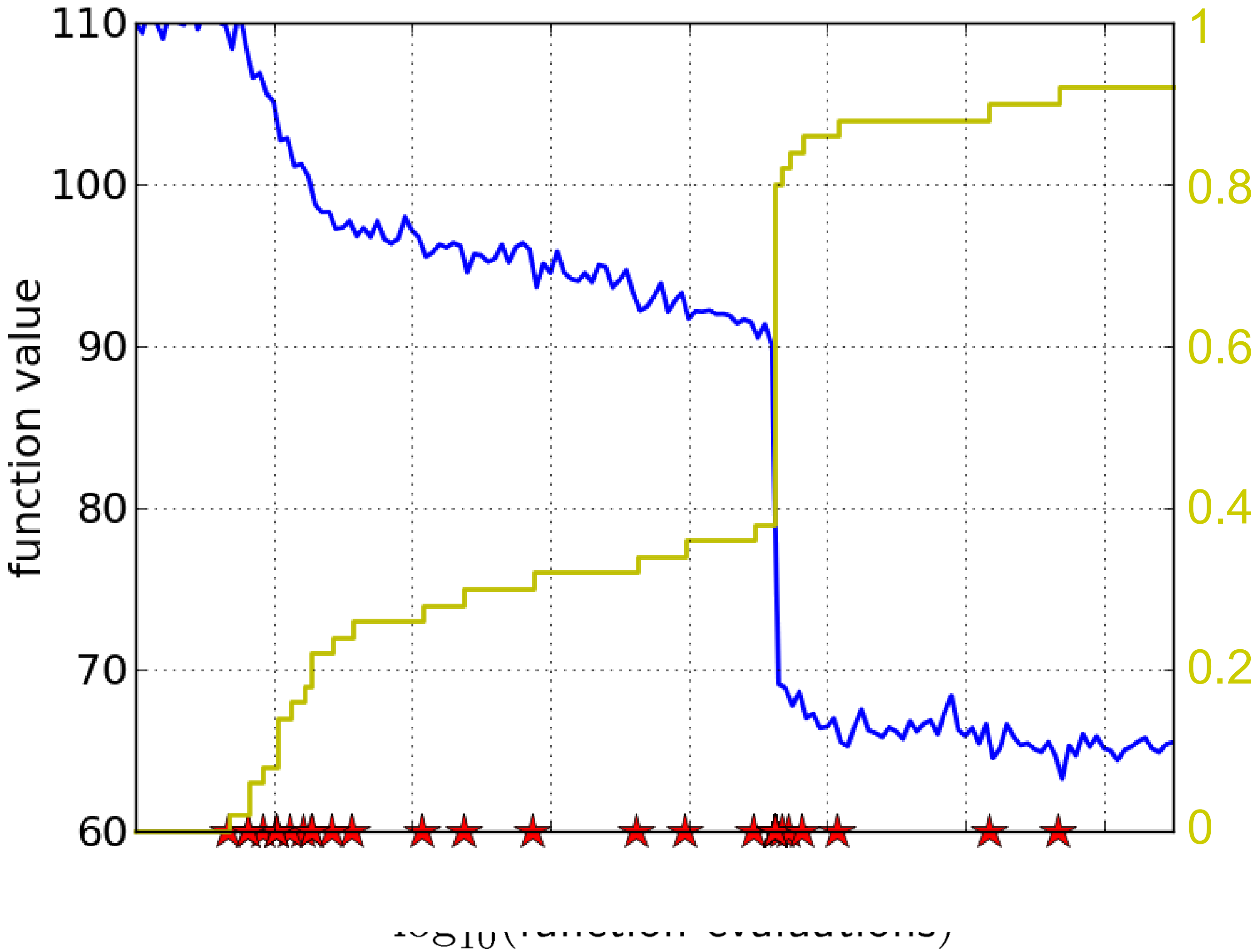
Reconstructing A Single Run



Reconstructing A Single Run

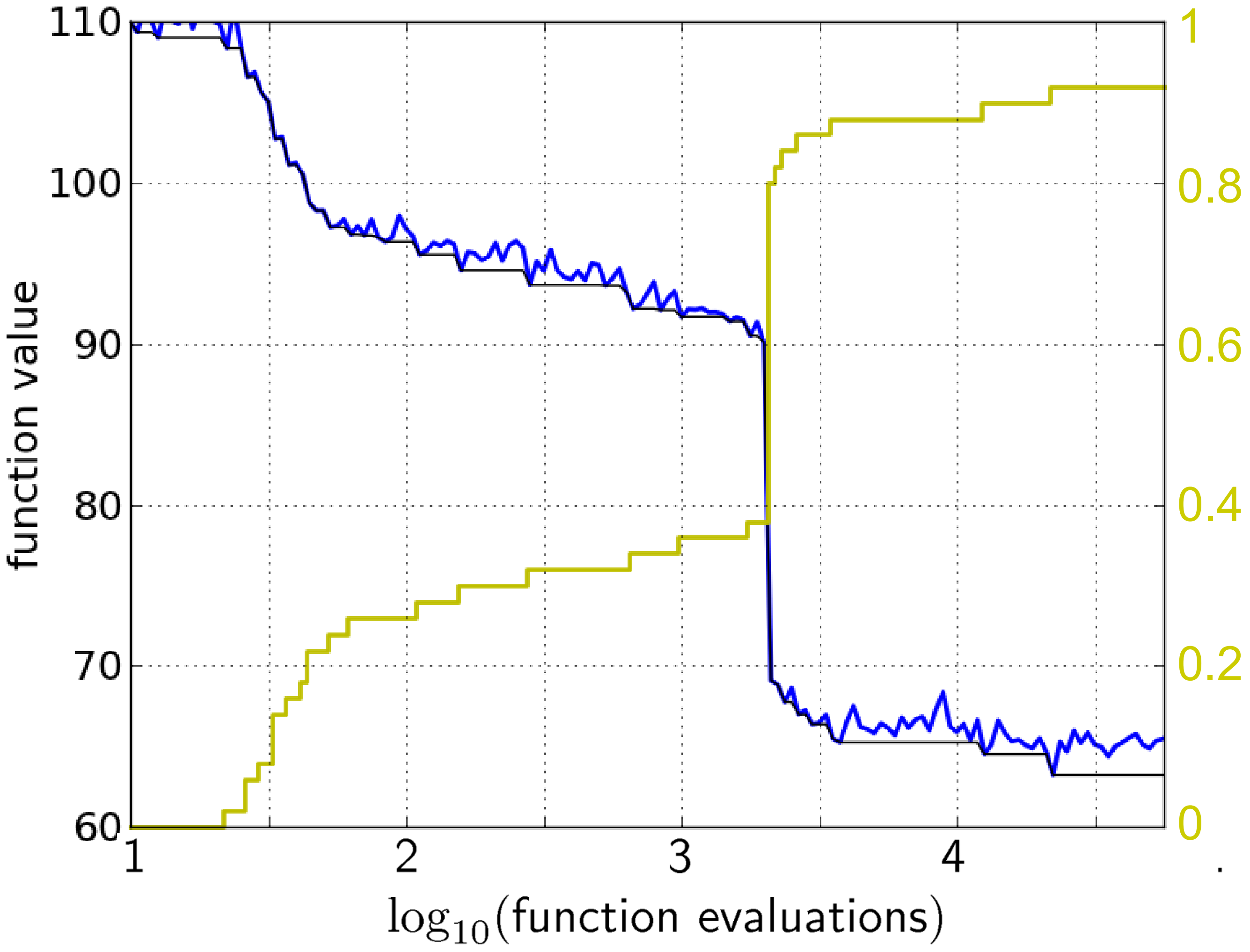


Reconstructing A Single Run



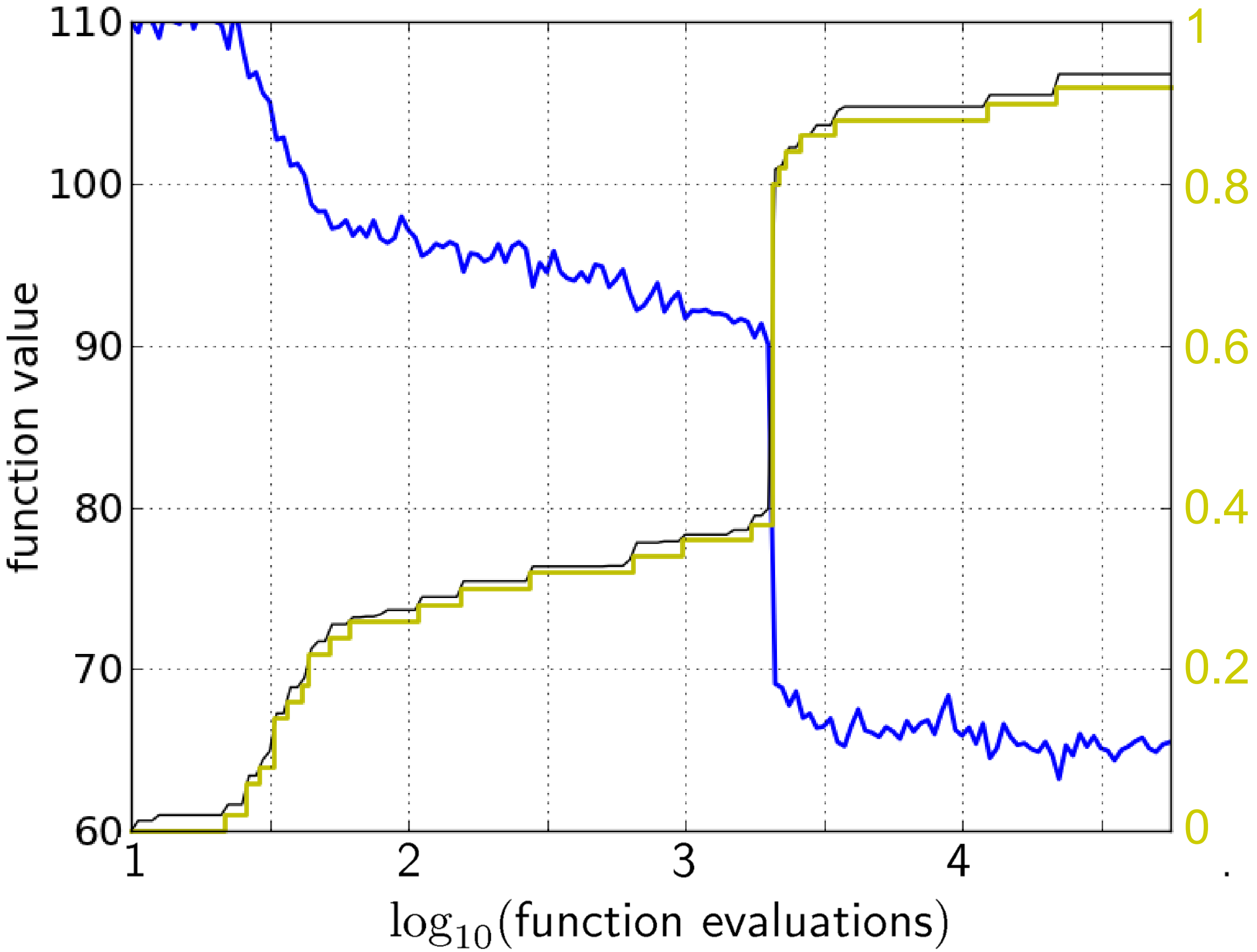
ıe empirical CDF makes a step for each star, is monotonous and displays for each budget the fraction of targets achieved within the budget

Reconstructing A Single Run



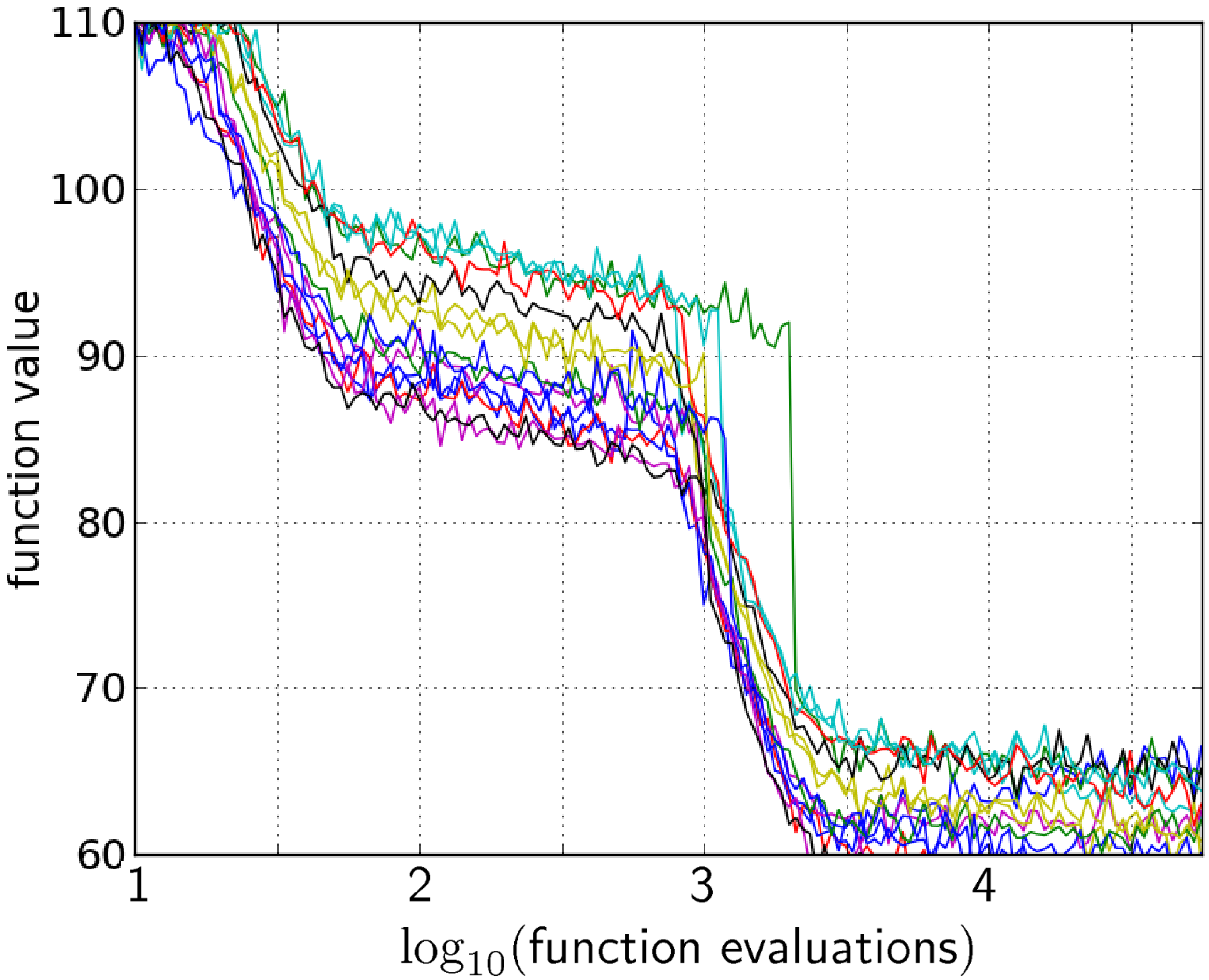
the ECDF recovers the monotonous graph, discretised and flipped

Reconstructing A Single Run



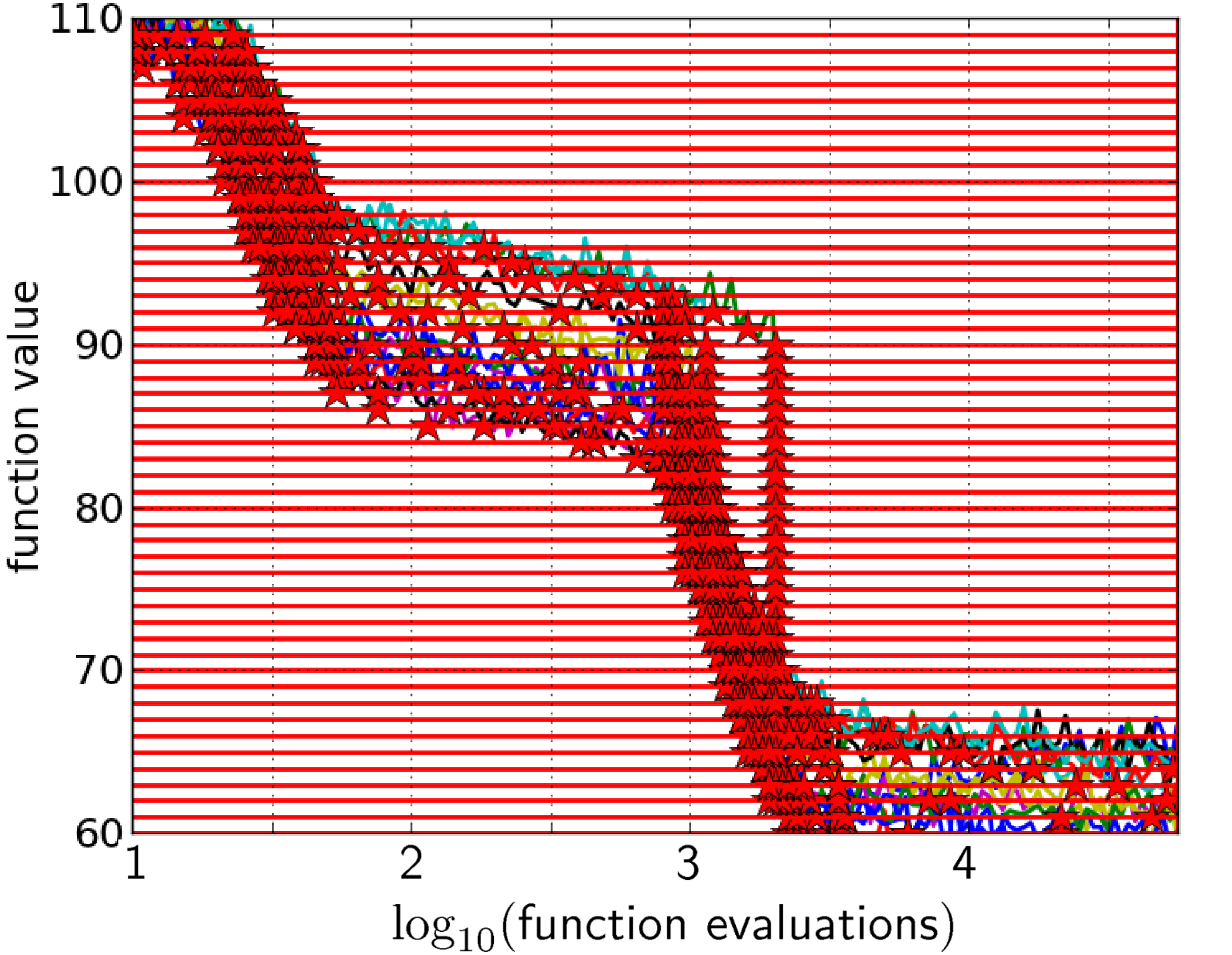
the ECDF recovers the monotonous graph, discretised and flipped

Aggregation



15 runs

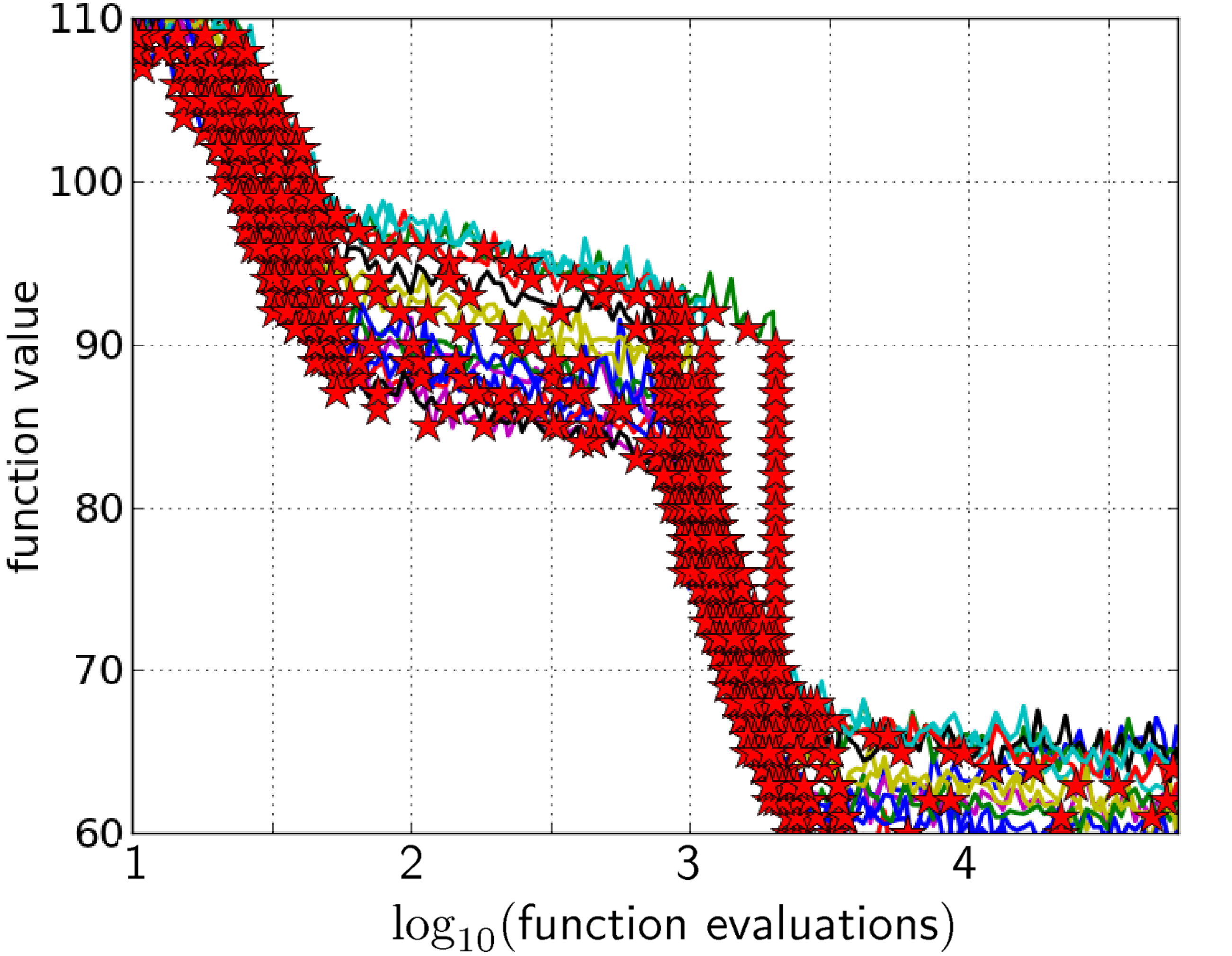
Aggregation



15 runs

50 targets

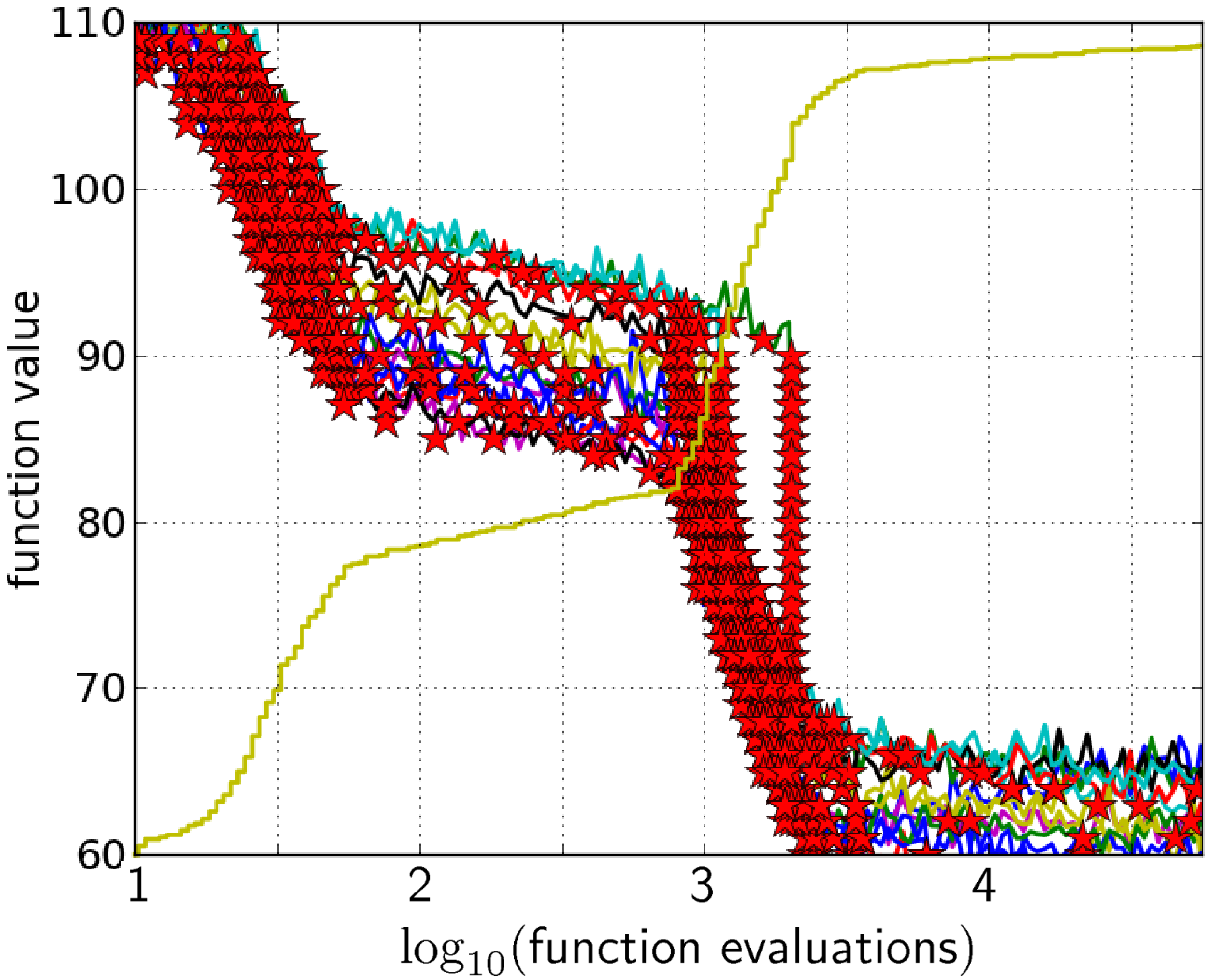
Aggregation



15 runs

50 targets

Aggregation

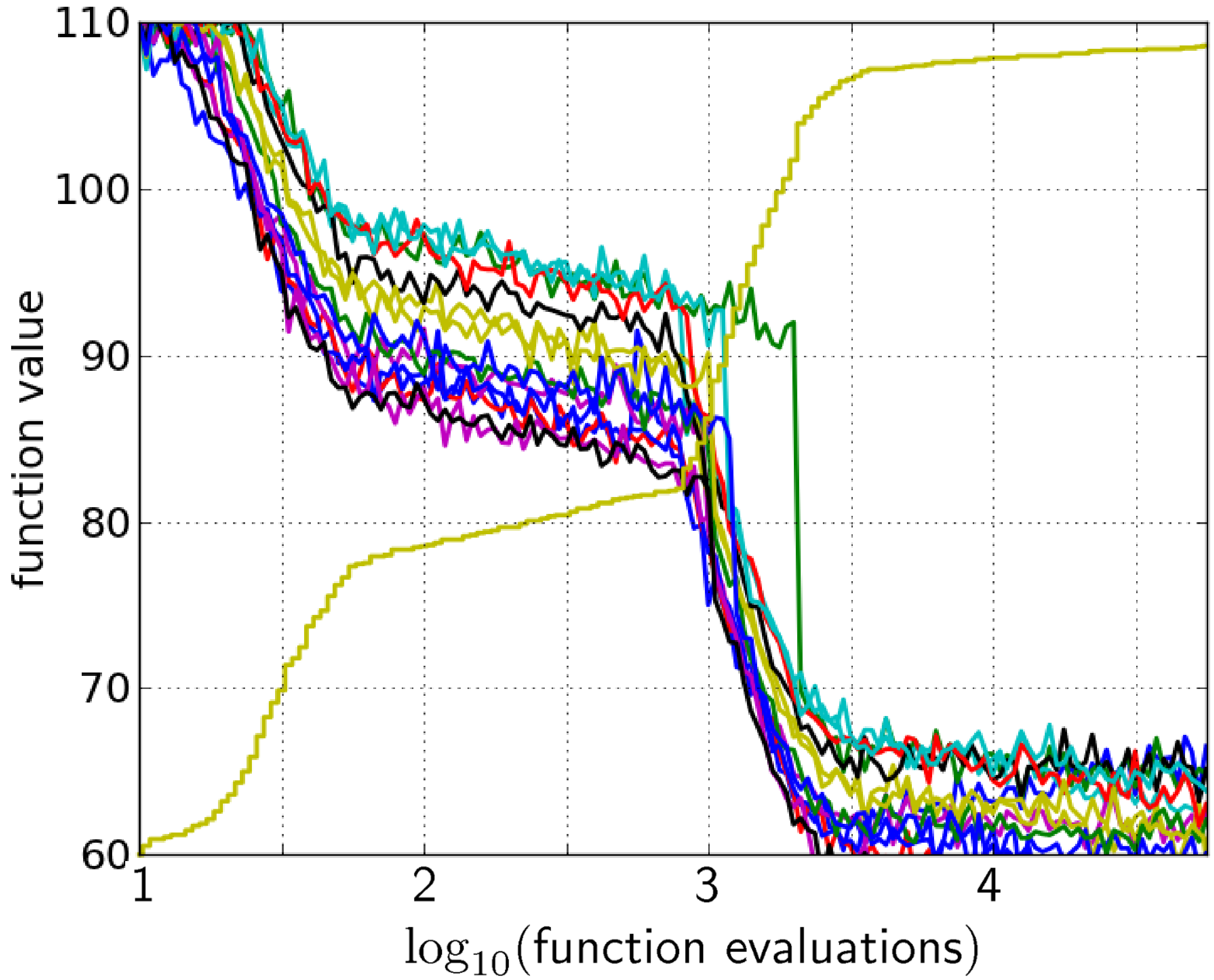


15 runs

50 targets

ECDF with 750 steps

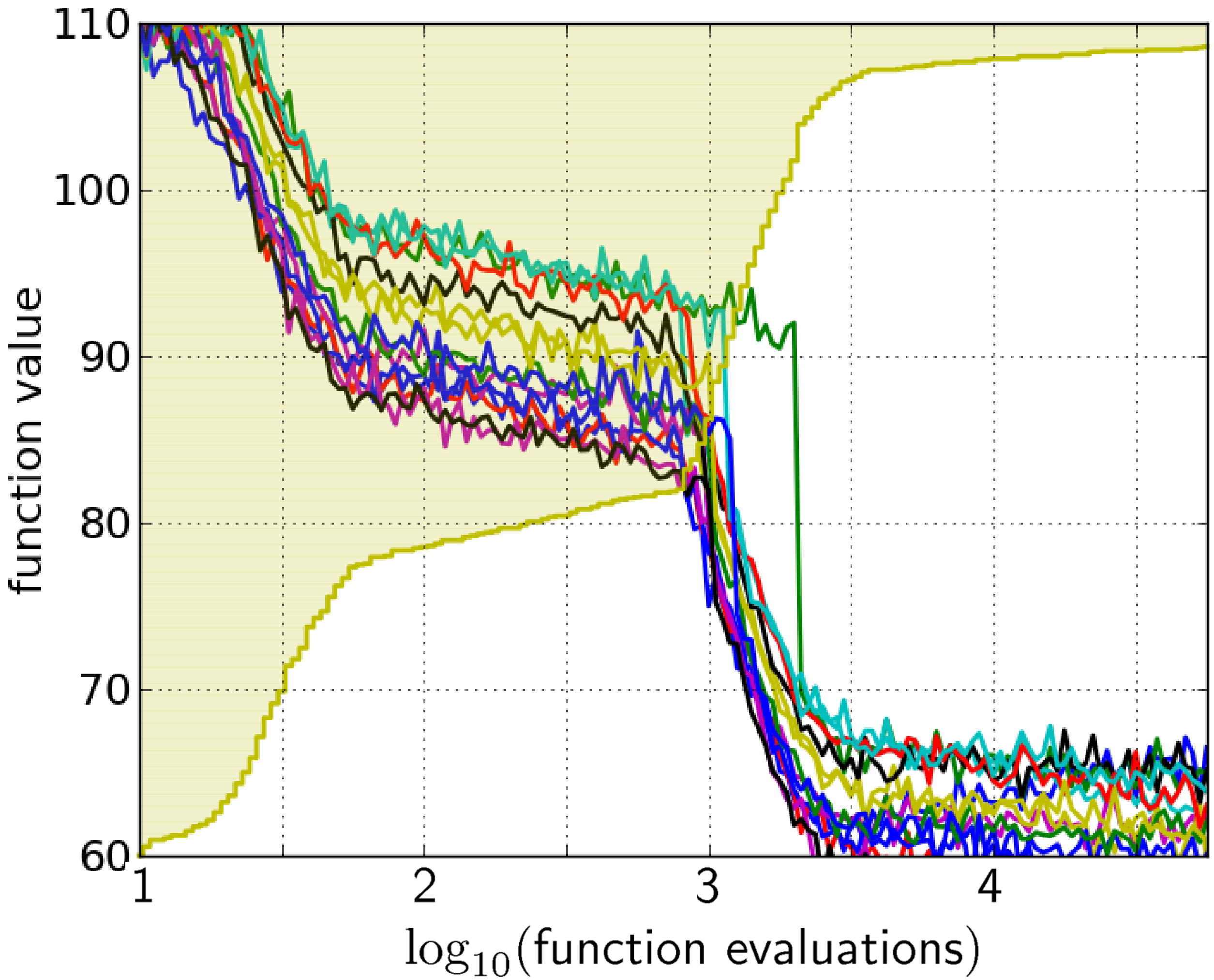
Aggregation



50 targets from 15 runs

...integrated in a single graph

Interpretation



50 targets from 15 runs integrated in a single graph

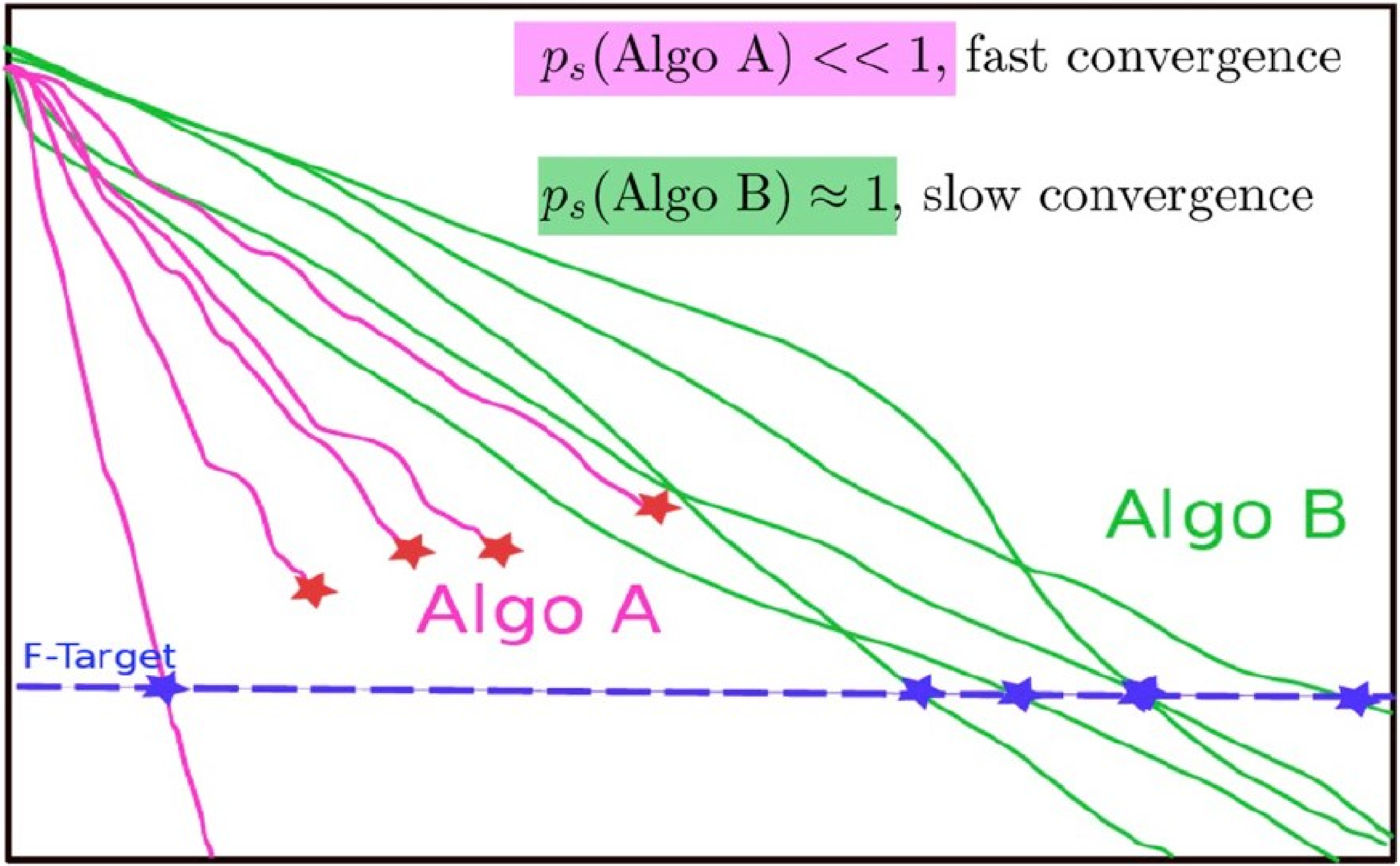
area over the ECDF curve

=

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:

RT_A^r

p_s(Algo Restart A) = 1

- Algo Restart B:

RT_B^r

p_s(Algo Restart B) = 1

Fixed-target: Measuring Runtime

- Expected running time of the restarted algorithm:

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

- Estimator average running time (aRT):

$$\widehat{p_s} = \frac{\#\text{successes}}{\#\text{runs}}$$

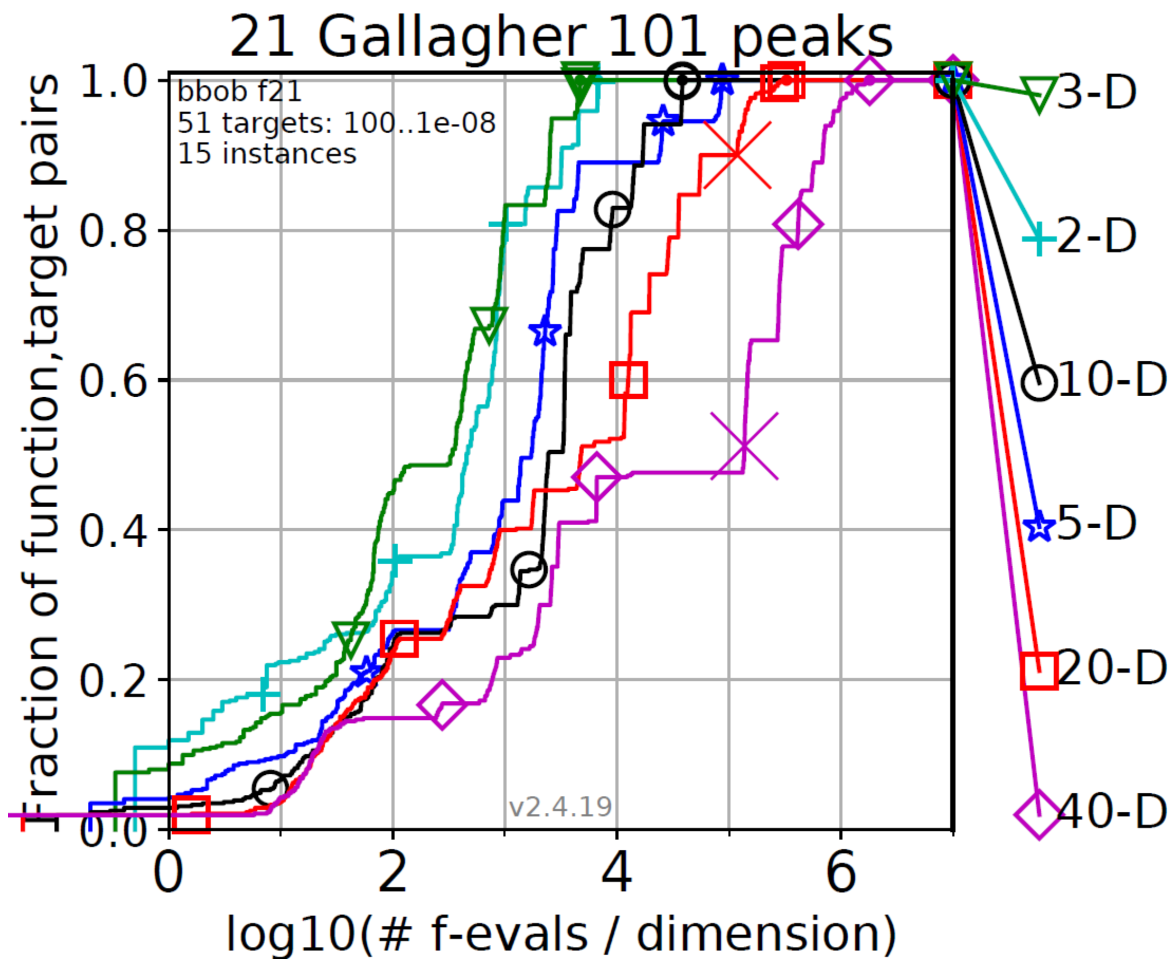
$$\widehat{RT_{unsucc}} = \text{Average evals of unsuccessful runs}$$

$$\widehat{RT_{succ}} = \text{Average evals of successful runs}$$

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

ECDFs with Simulated Restarts

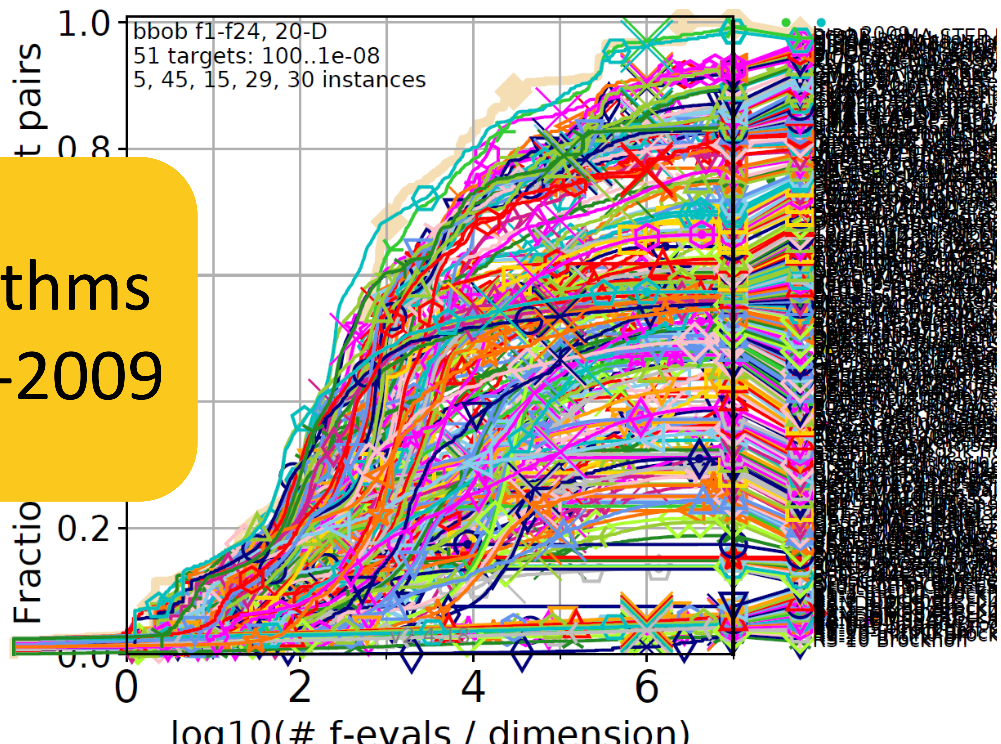
What we typically plot are ECDFs of the simulated restarted algorithms (exception: multiobjective case)



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

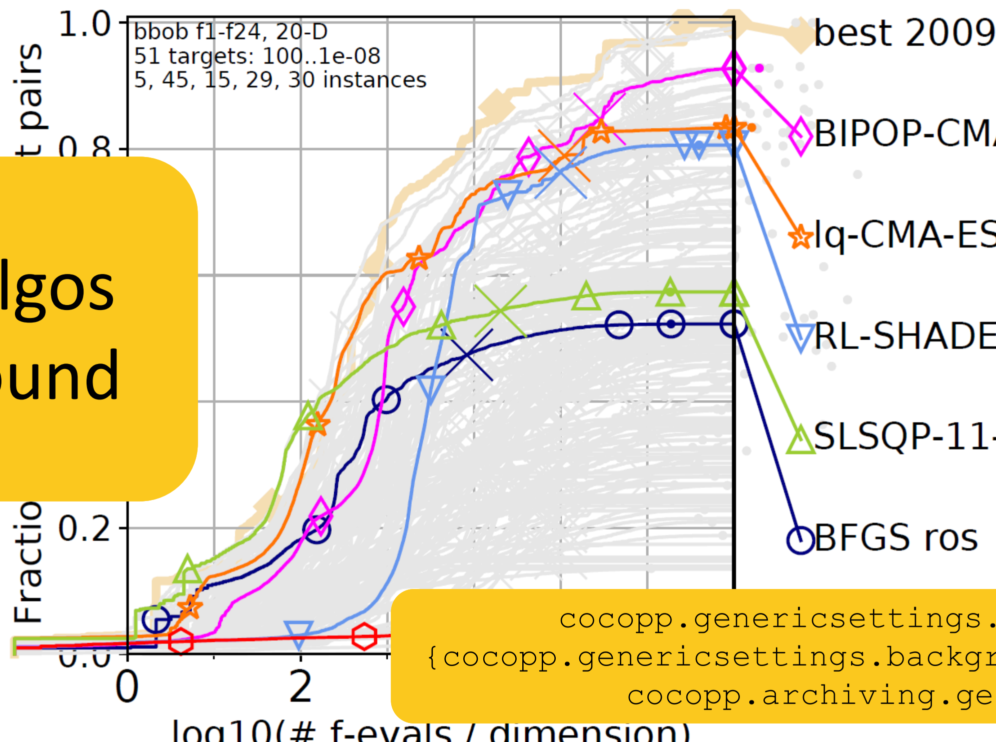


200+ algorithms since BBOB-2009

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

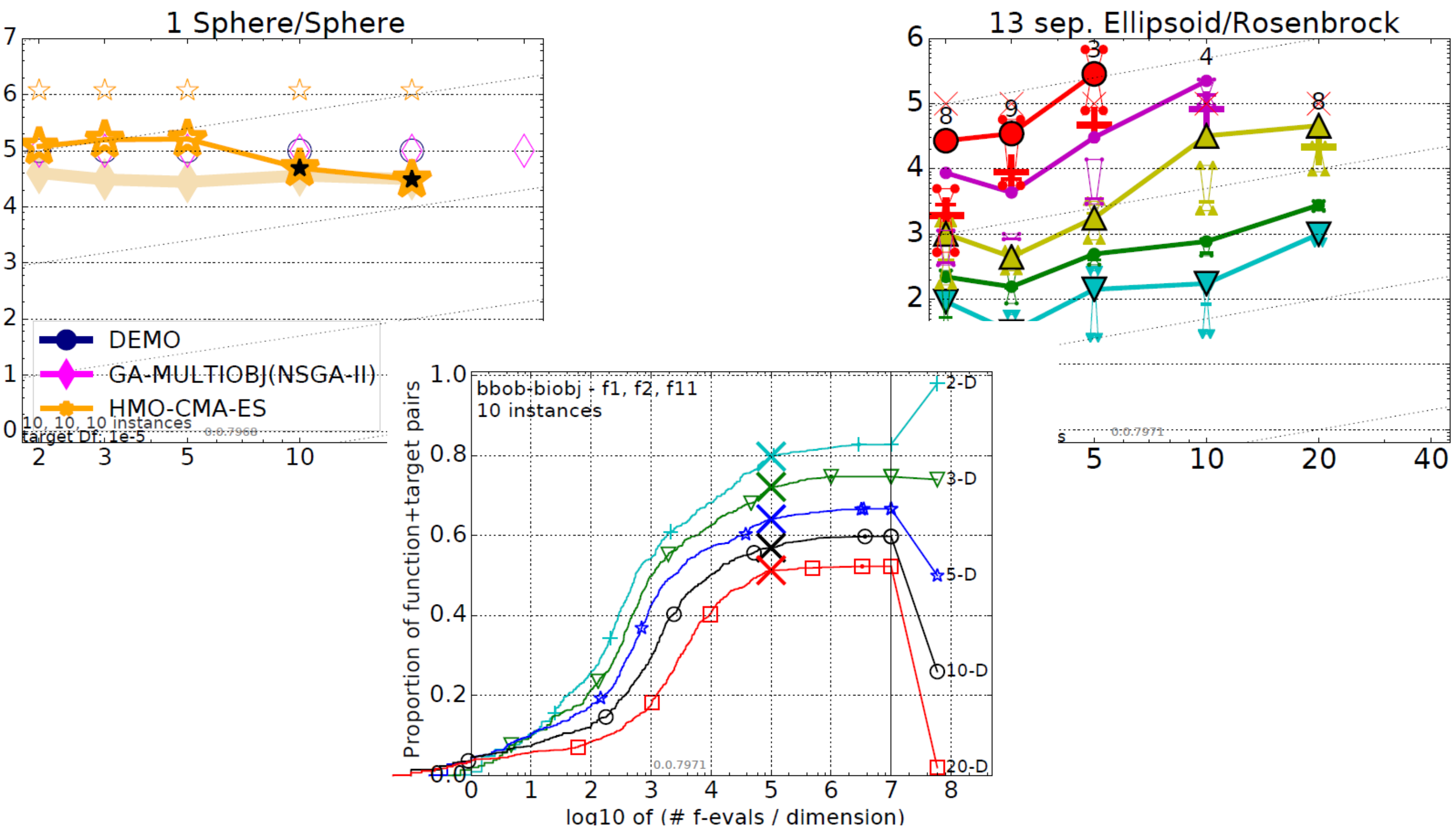


all but 6 algos in background

```
cocopp.genericsettings.background =
{cocopp.genericsettings.background_default_style:
cocopp.archiving.get('bbob')}
```

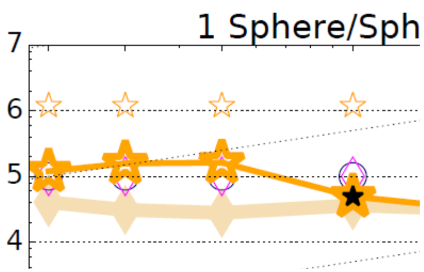
More Automated Plots...

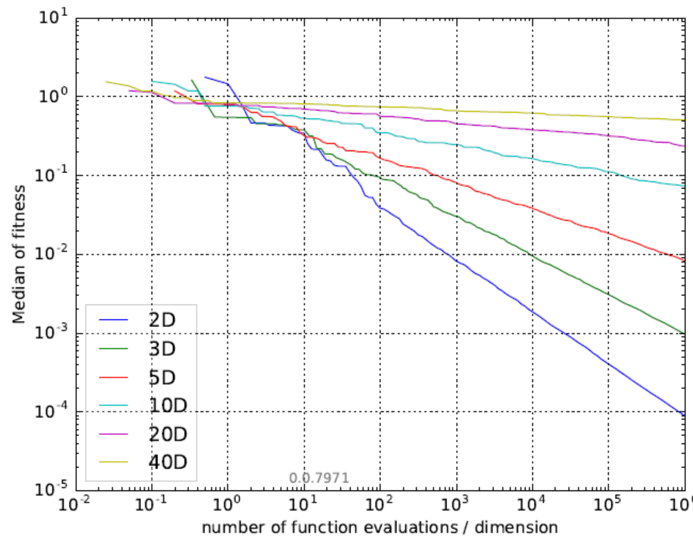
...but no time to explain them here ☹

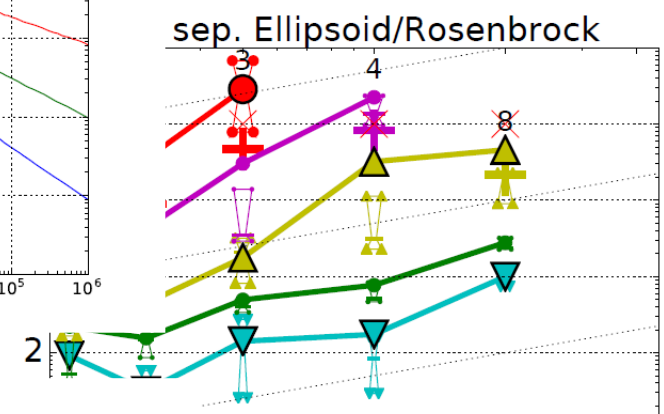


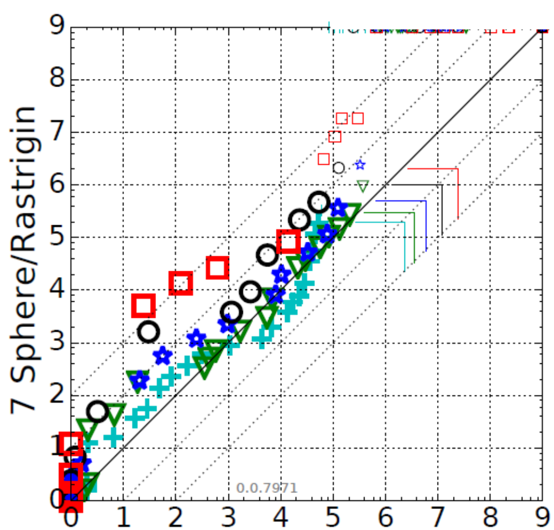
More Automated Plots...

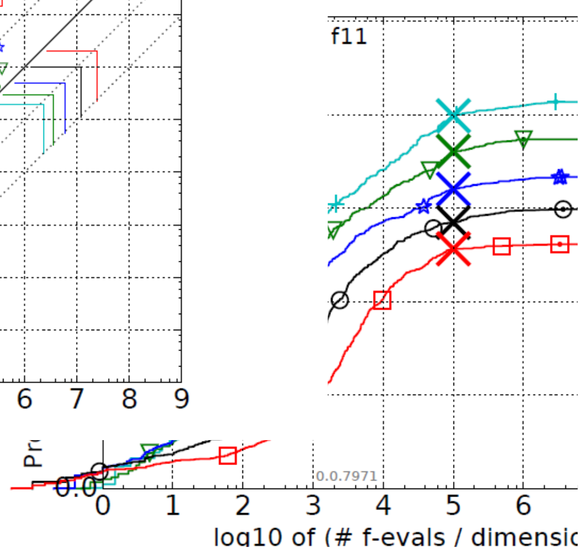
...but no time t

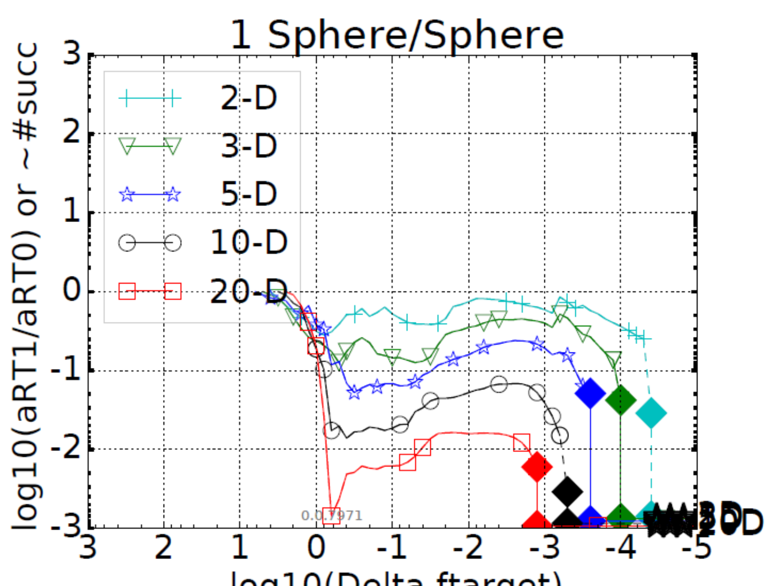












and now?

BBOB-2021

Session Saturday 10th of July, 2021	
11:00 - 11:30	The BBOBies: **A Short Introduction to COCO and BBOB**
11:30 - 11:55	Michał Okulewicz, Mateusz Zaborski: **Benchmarking SHADE algorithm enhanced with model based optimization on the BBOB noiseless testbed**
11:55 - 12:20	Dimo Brockhoff, Baptiste Plaquevent-Jourdain, Anne Auger, and Nikolaus Hansen: **DMS and MultiGLODS: Black-Box Optimization Benchmarking of Two Direct Search Methods on the Biobjective bbob-biobj Test Suite**
12:20 – 12:50	**Open Discussion**