Performance Assessment in Optimization

Anne Auger

RandOpt Team
INRIA and CMAP, Ecole Polytechnique
anne.auger@inria.fr

General Problem

Evaluate the performance of optimization algorithms

Compare the performance of different algorithms

understand strength and weaknesses of algorithms

help in design of new algorithms

General Problem (cont.)

Algorithms are in general too complicated to be evaluated theoretically on the wide range of problems/ difficulties one is interested to solve

need to do some benchmarking, i.e. evaluate empirically on test functions the performance of an optimizer

run the optimizer several times independently on a set of benchmark function

display some statistical measures of performance

Test functions

Many real world problems share common difficulties:

→ non separability (correlations between variables)

→ ill-conditioned (certain direction steeper than others),

→ ruggedness (noise, ...),

→ multi-modality

→ non-convexity

Ideally an optimizer should cope with all of them

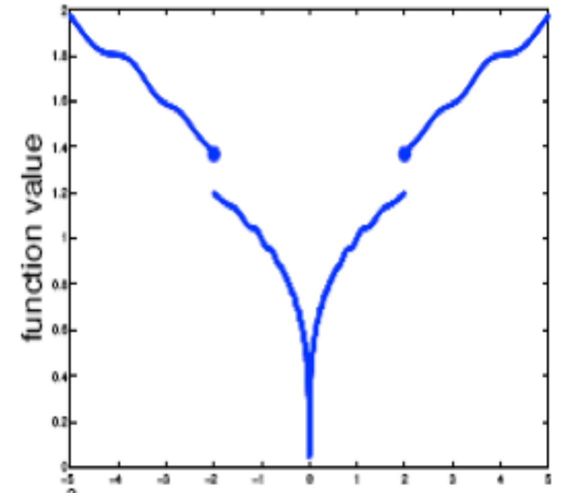
function testbed:

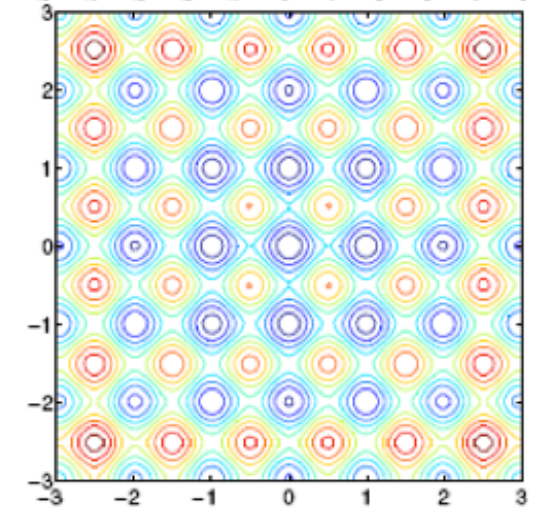
should "reflect reality": should model typical difficulties one is willing to solve

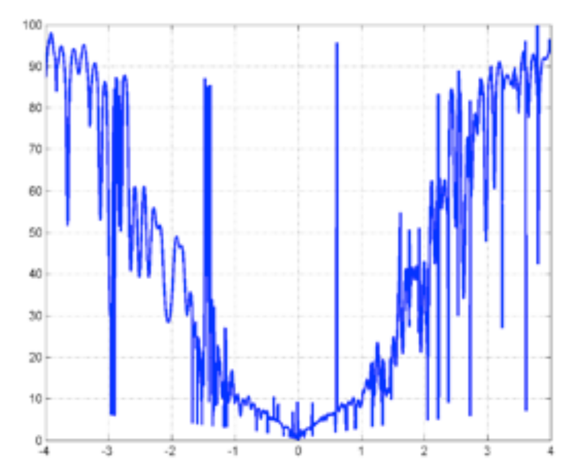
mainly non-convex and non-separable

scalable with the search space dimension

not too easy to solve, but yet comprehensible





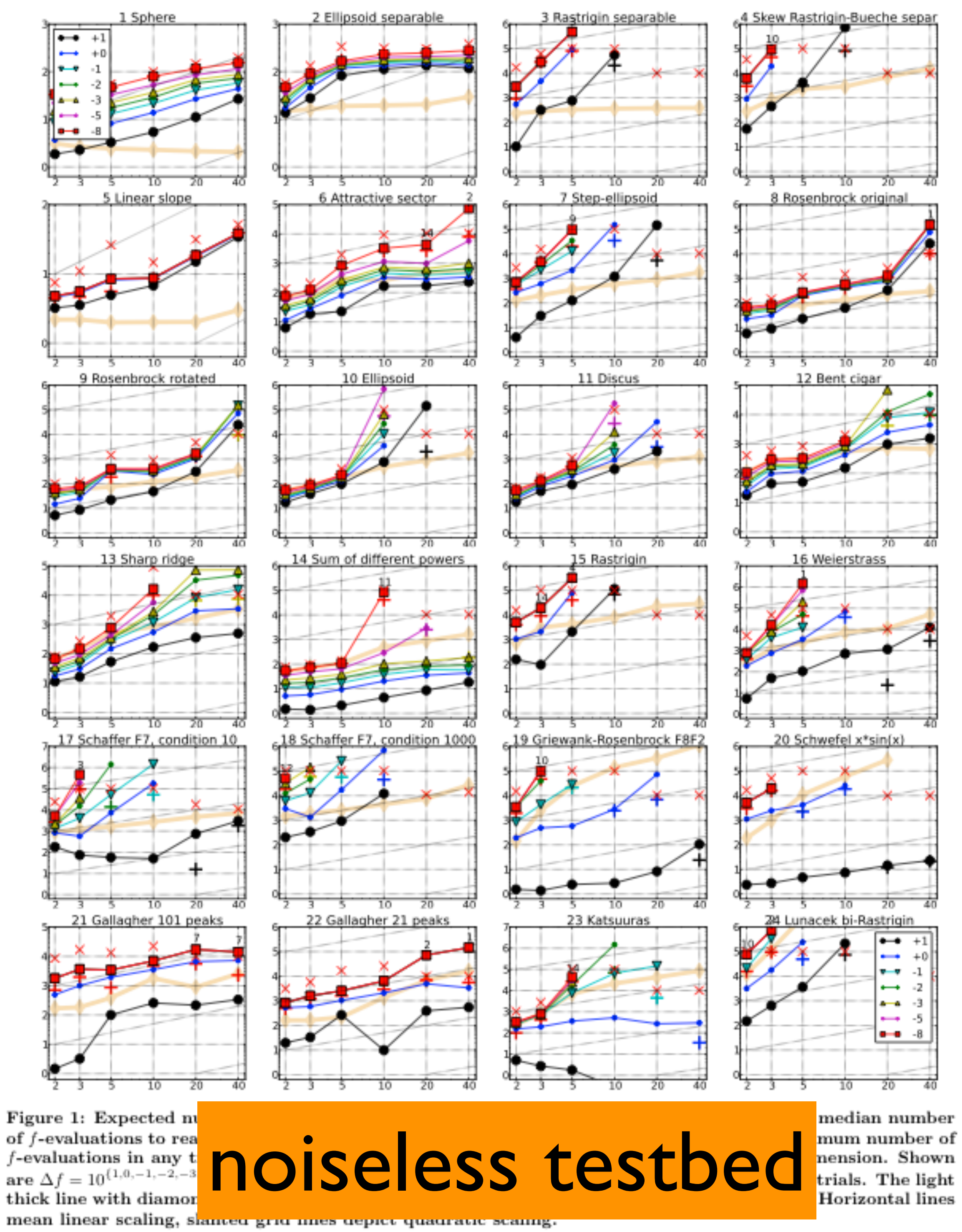


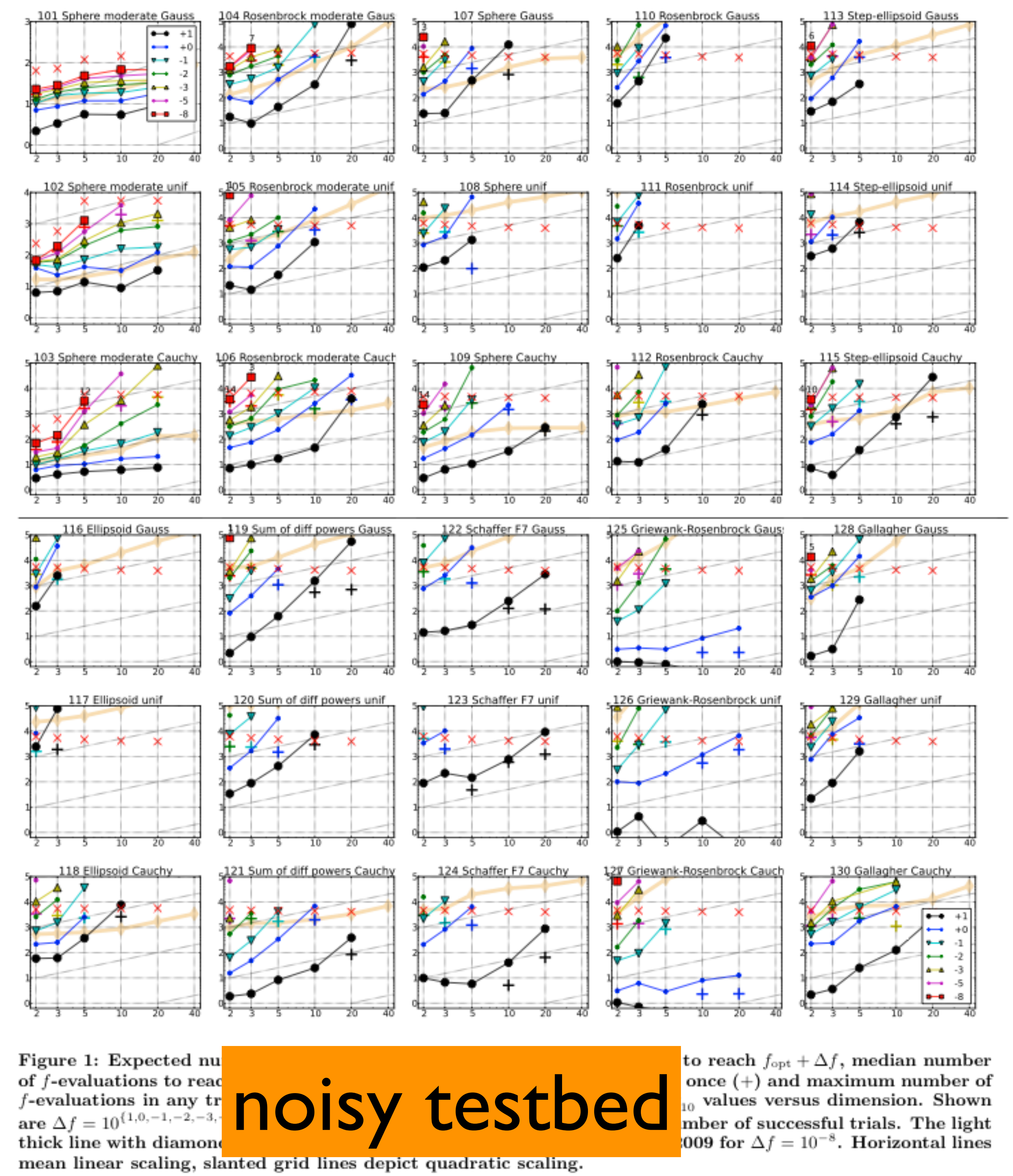
State-of-the-art Test Suite

Black-Box Optimization Benchmarking test suite

noiseless / noisy testbed

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





Performance measure

CPU time (to reach a given target)

drawbacks: depend on the implementation, on the language, on the machine

time is spent on code optimization instead of science

Testing heuristics, we have it all wrong, J.N. Hooker, 1995
Journal of Heuristics

Prefer "absolute" value: # of function evaluations to reach a given target

assumptions: internal cost of the algorithm negligible or measured independently

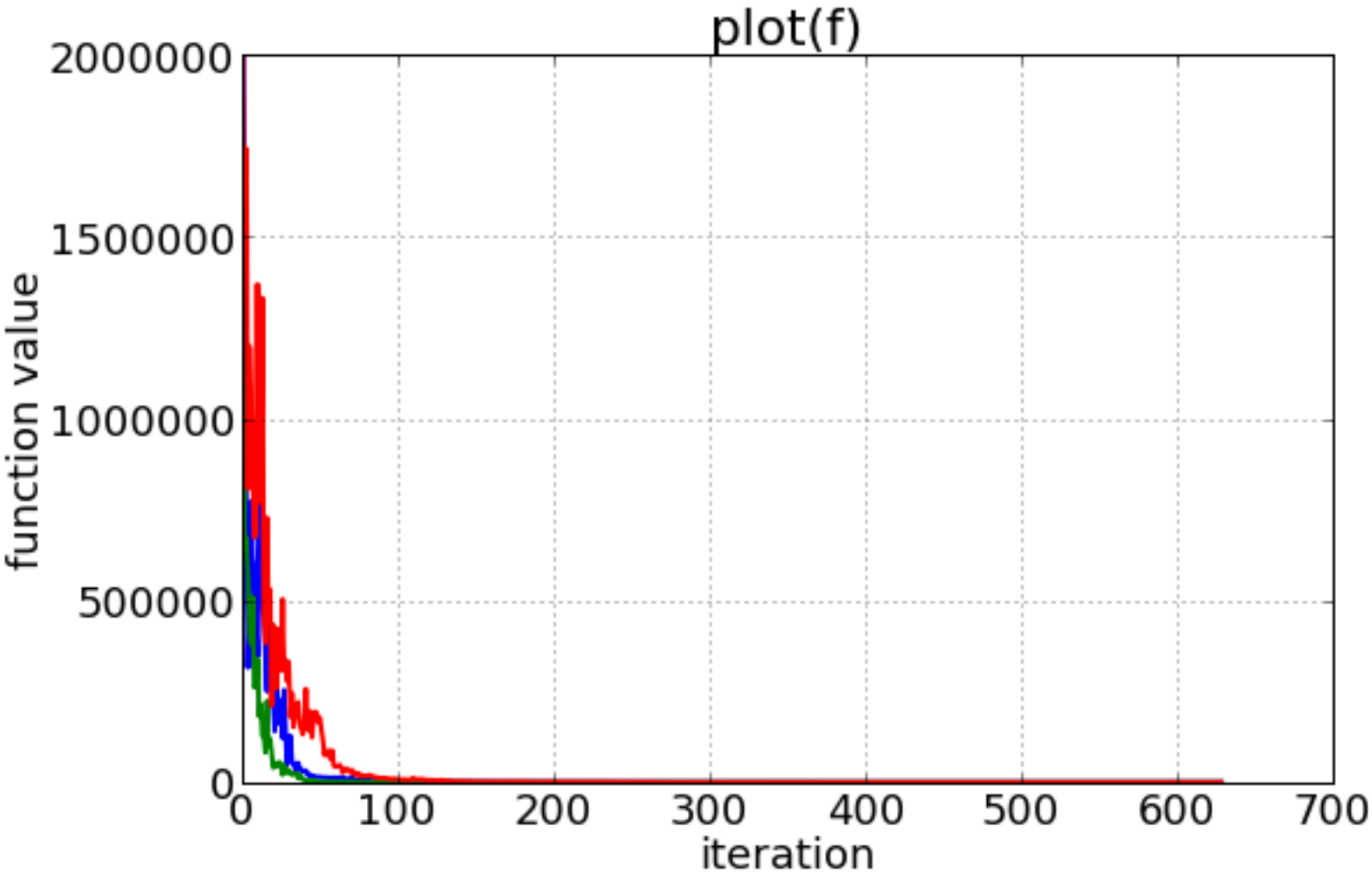
Performance measure

empirically

convergence graphs is all we have to start with

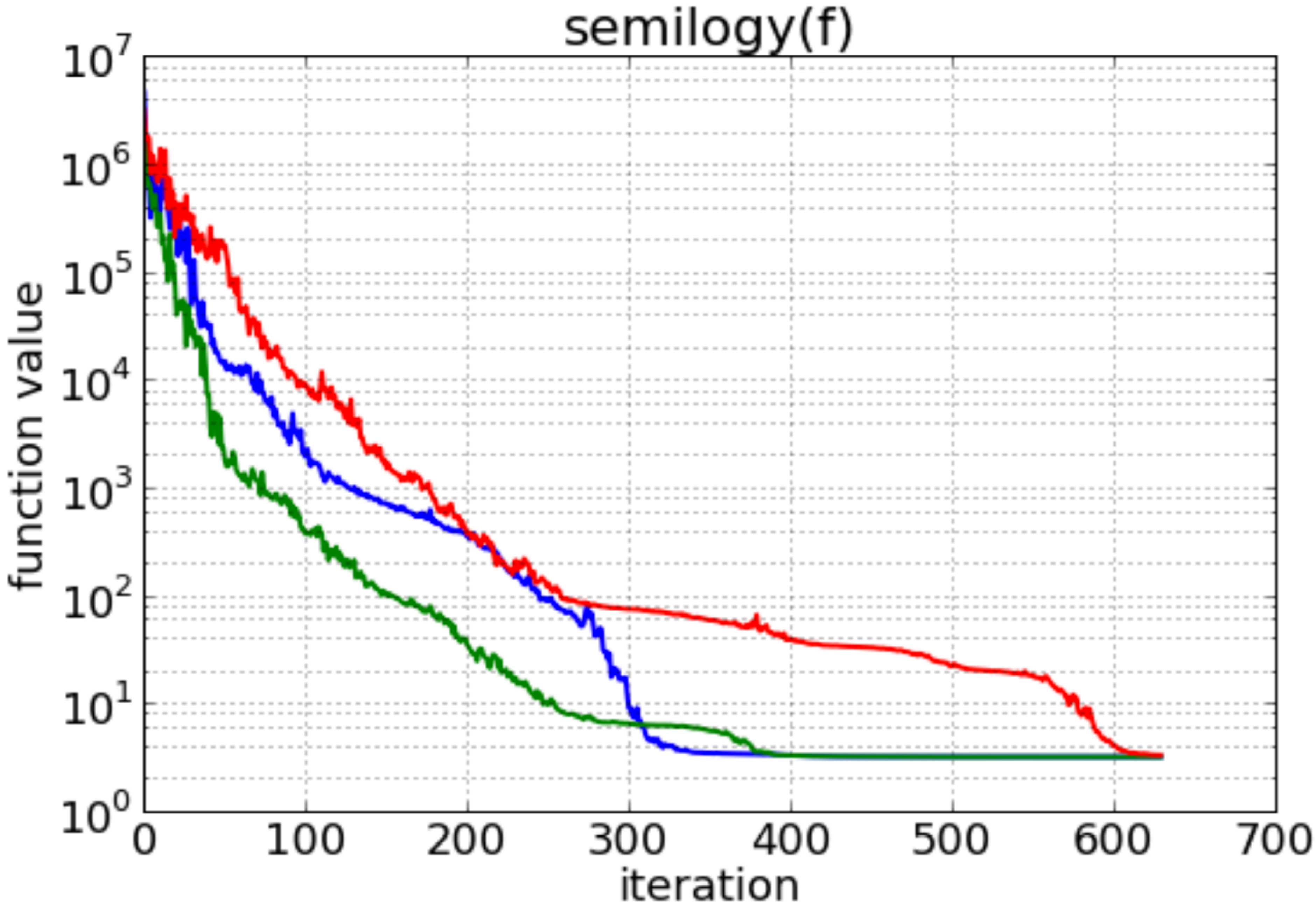
the right presentation cannot be overestimated: details are important!

Displaying 3 runs (three trials)



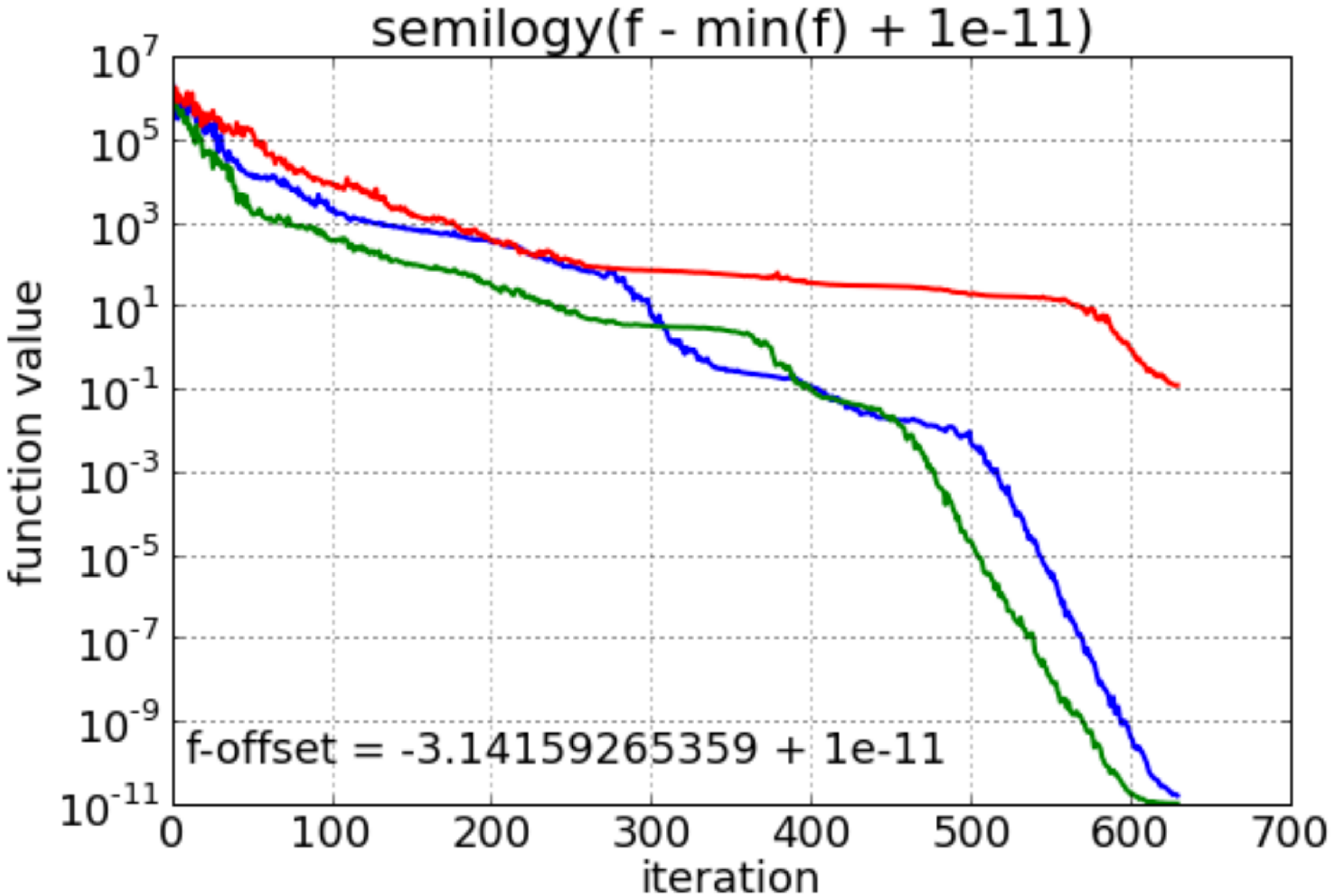
not like this (it's unfortunately a common picture)

Displaying 3 runs (three trials)



better like this (shown are the same data), caveat: fails with negative f-values

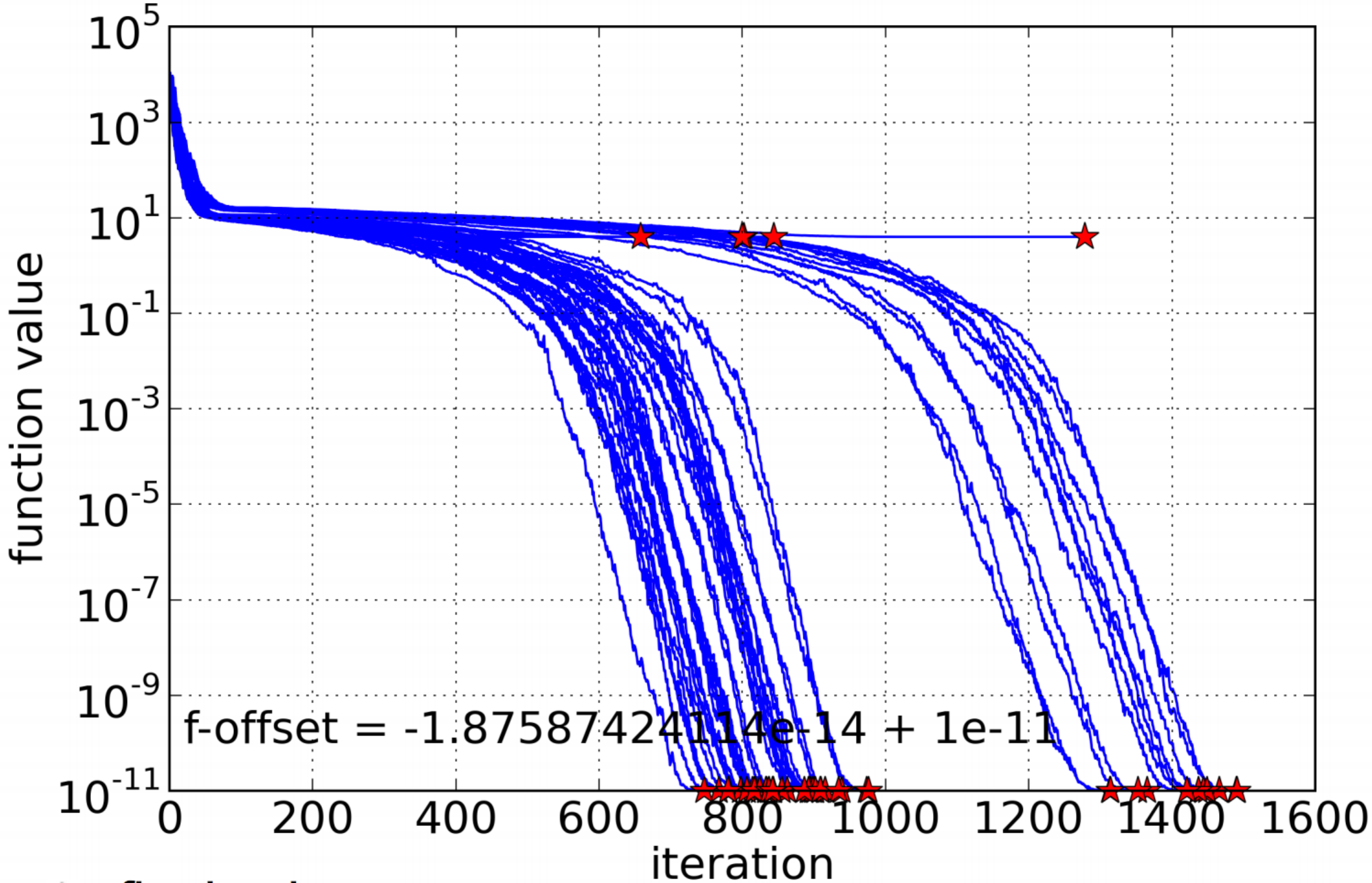
Displaying 3 runs (three trials)



even better like this: subtract minimum value over all runs

Displaying 51 runs

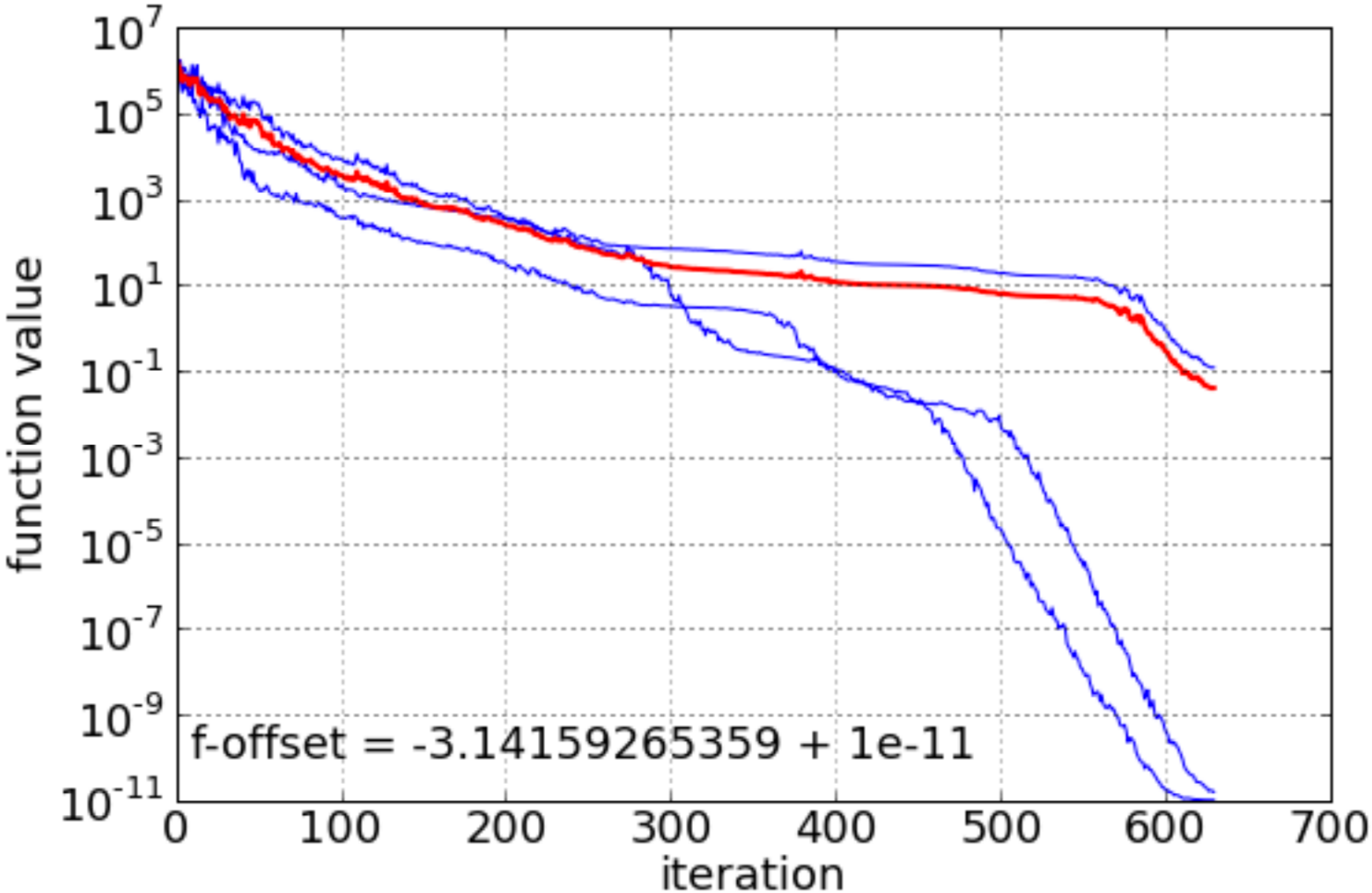
don't hesitate to display all data (the appendix is your friend)



★ : final value

observation: three different "modes", which would be difficult to represent or recover in single statistics

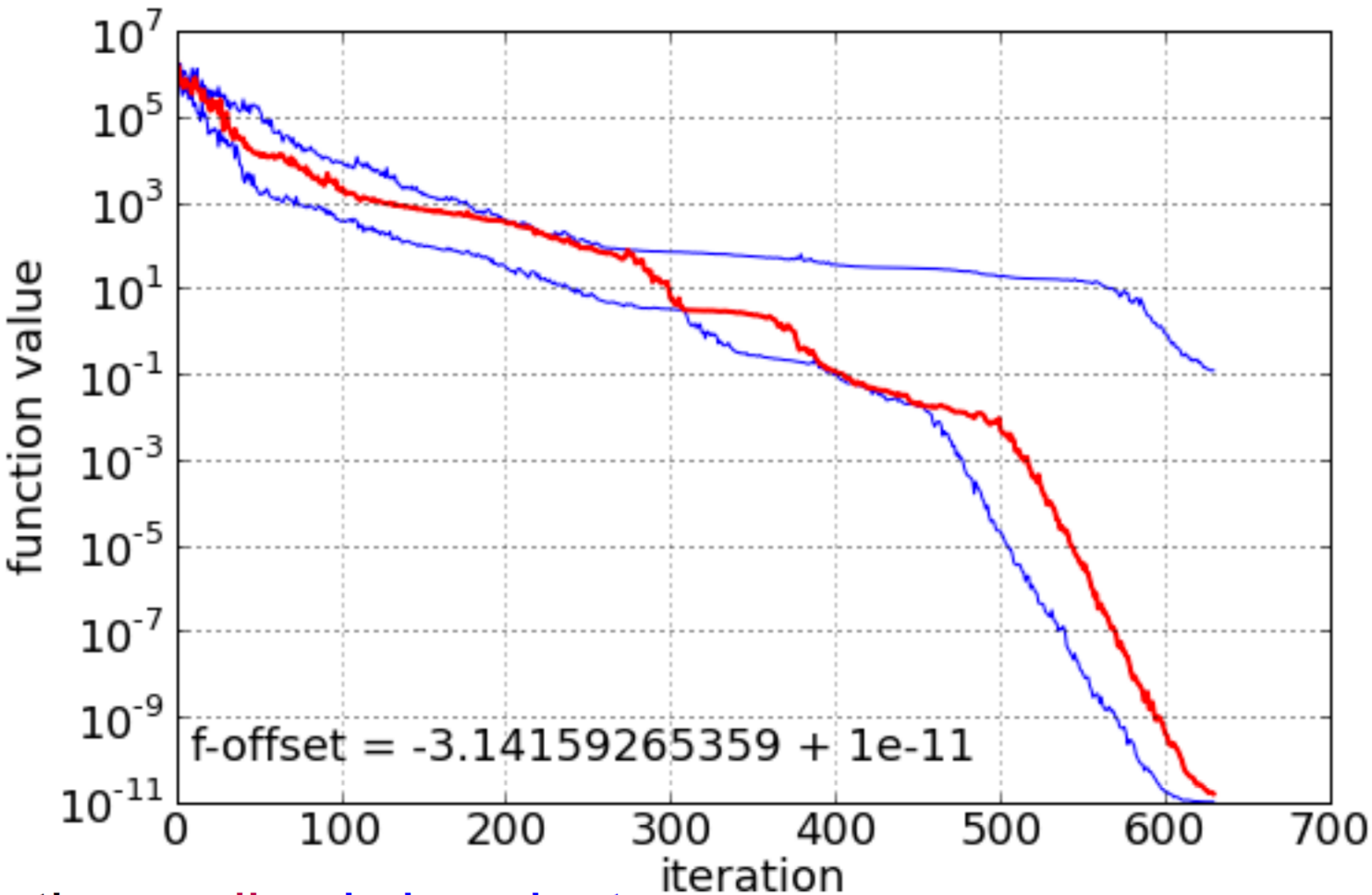
Which Statistics?



mean/average function value

- tends to emphasize large values

Which Statistics?



the median is invariant

- unique for uneven number of data
- independent of log-scale, offset...

  median(log(data))=log(median(data))
- same when taken over x- or y-direction

On performance measures
Requirements

“Algorithm A is 10/100 times faster than Algorithm B to solve this type of problems”

On performance measures
Requirements

"Algorithm A is 10/100 times faster than Algorithm B to solve this type of problems"

quantitative measures

On performance measures
Requirements

"Algorithm A is 10/100 times faster than Algorithm B to solve this type of problems"

quantitative measures

As opposed to

F.	EFWA *vs* EFWA-NG		
	EFWA	EFWA-NG	p-value
f_1	-1.3999E+03	-1.3999E+03	**2.316E-03**
f_2	6.8926E+05	6.5258E+05	4.256E-01
f_3	7.7586E+07	6.4974E+07	8.956E-01
f_4	-1.0989E+03	-1.0989E+03	7.858E-01
f_5	-9.9992E+02	-9.9992E+02	**4.290E-02**
f_6	-8.5073E+02	-8.4462E+02	1.654E-01
[illegible]	[illegible]	[illegible]	[illegible]

displayed: mean f-value after 3.10^5 f-evals (51 runs)
bold: statistically significant
concluded: "EFWA significantly better than EFWA-NG"

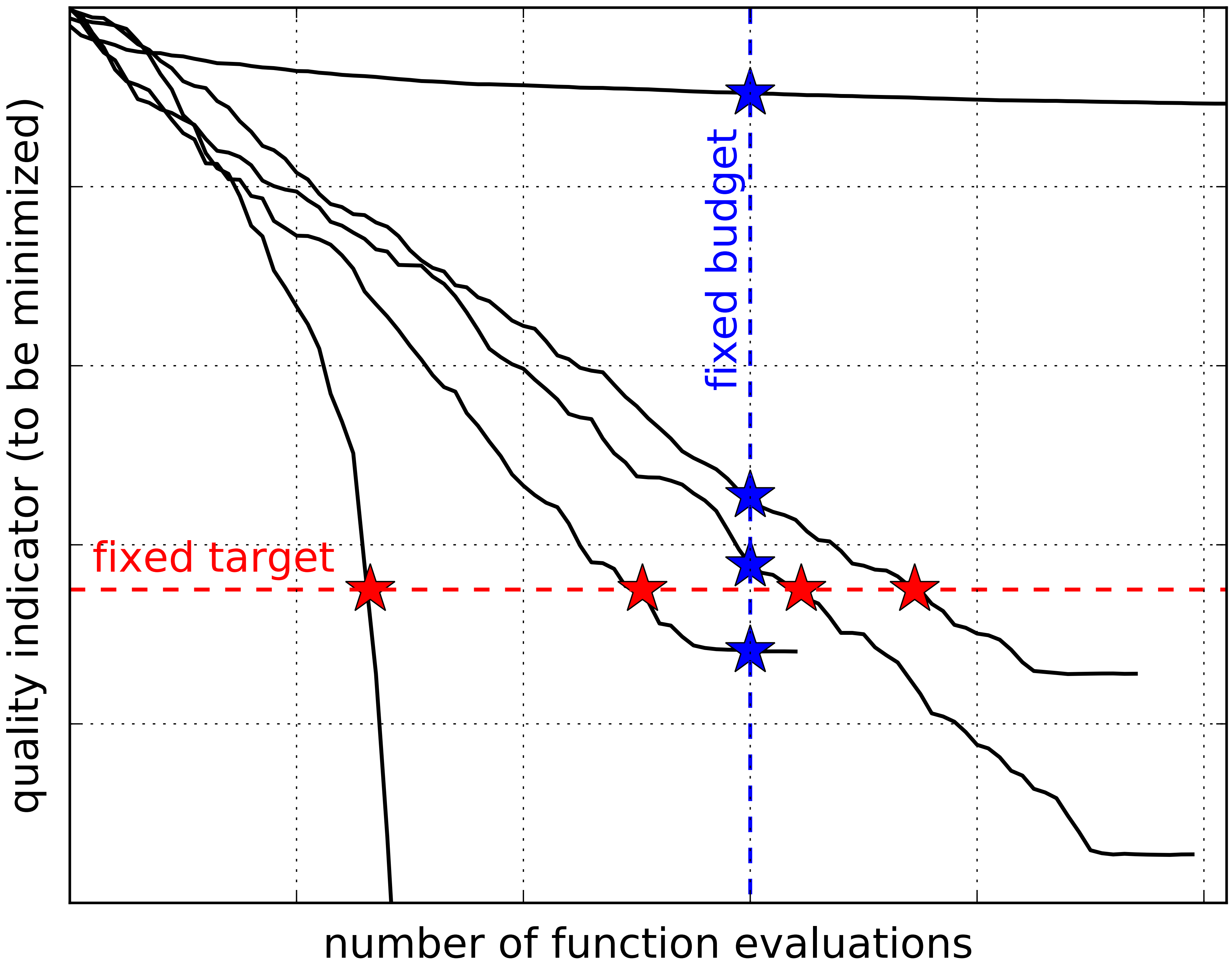
Source: Dynamic search in fireworks algorithm, Shaoqiu Zheng, Andreas Janecek, Junzhi Li and Ying Tan CEC 2014

On performance measures
Requirements

a performance measure should be

- quantitative, with a ratio scale
- well-interpretable with a meaning
- relevant in the “real world”
- simple

Fixed Cost versus Fixed Budget
Collecting Data



Fixed Cost versus Fixed Budget
Collecting Data

Collect for a given target (several target), the number of function evaluations needed to reach a target

Repeat several times:

- if algorithms are stochastic, never draw a conclusion from a single run

- if deterministic algorithm, repeat by changing (randomly) the initial conditions

Displaying Performance

ECDF

Average RunTime (ART)

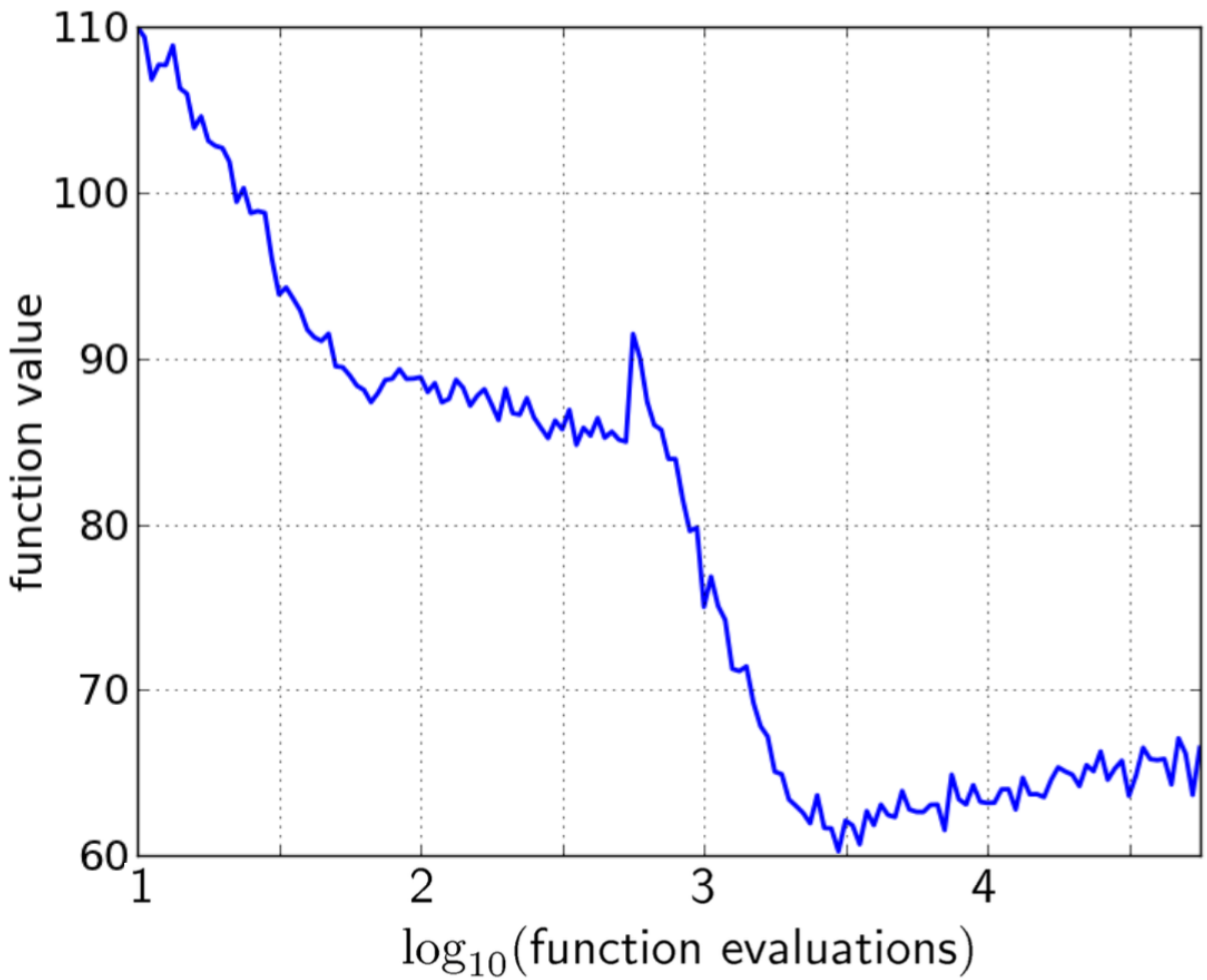
Displaying Performance
ECDF

Empirical Cumulative Distribution (ECDF) of Runtime

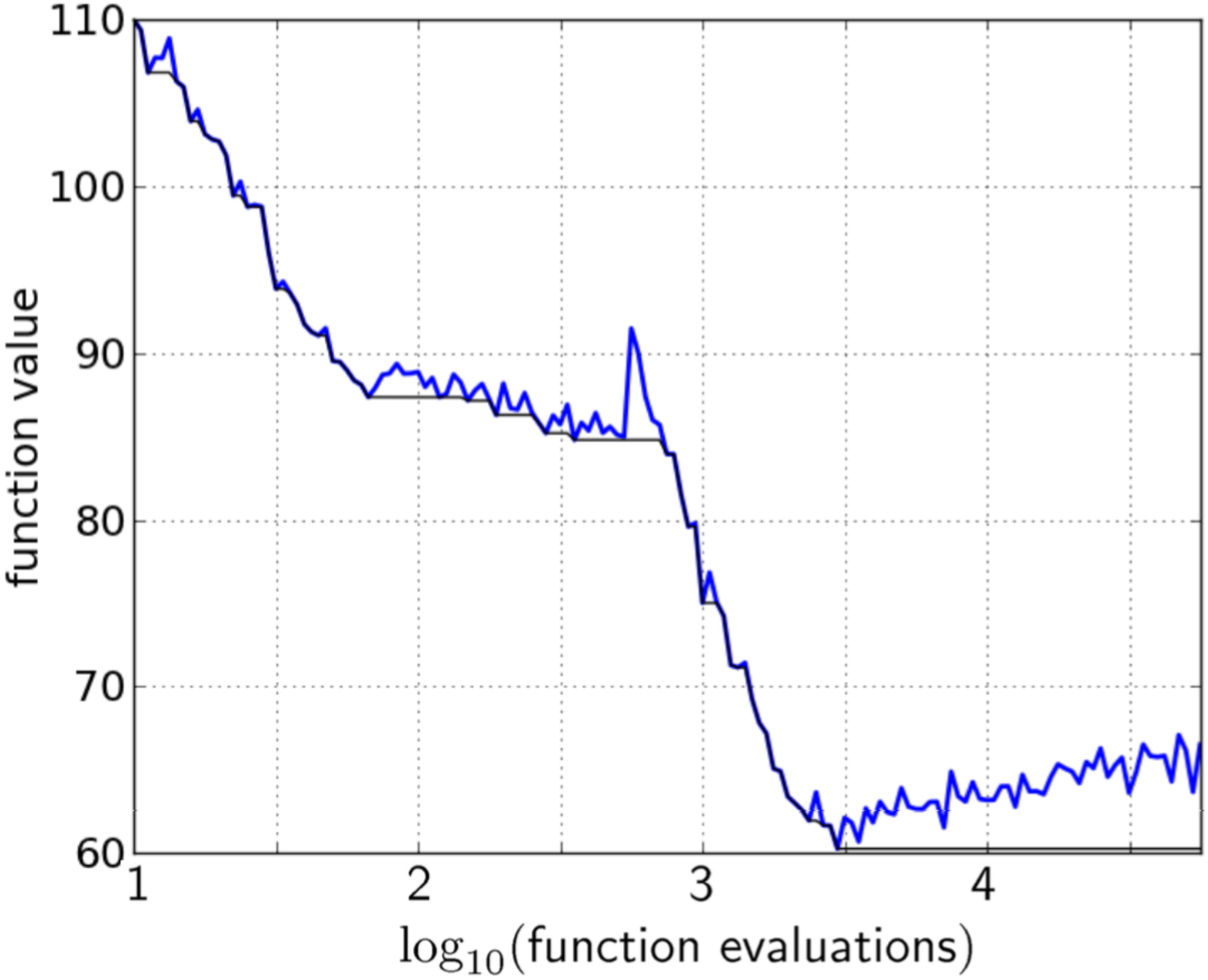
also known as data profile

[Moré, Wild, Benchmarking Derivative-Free Optimization Algorithms, SIOPT 2009]

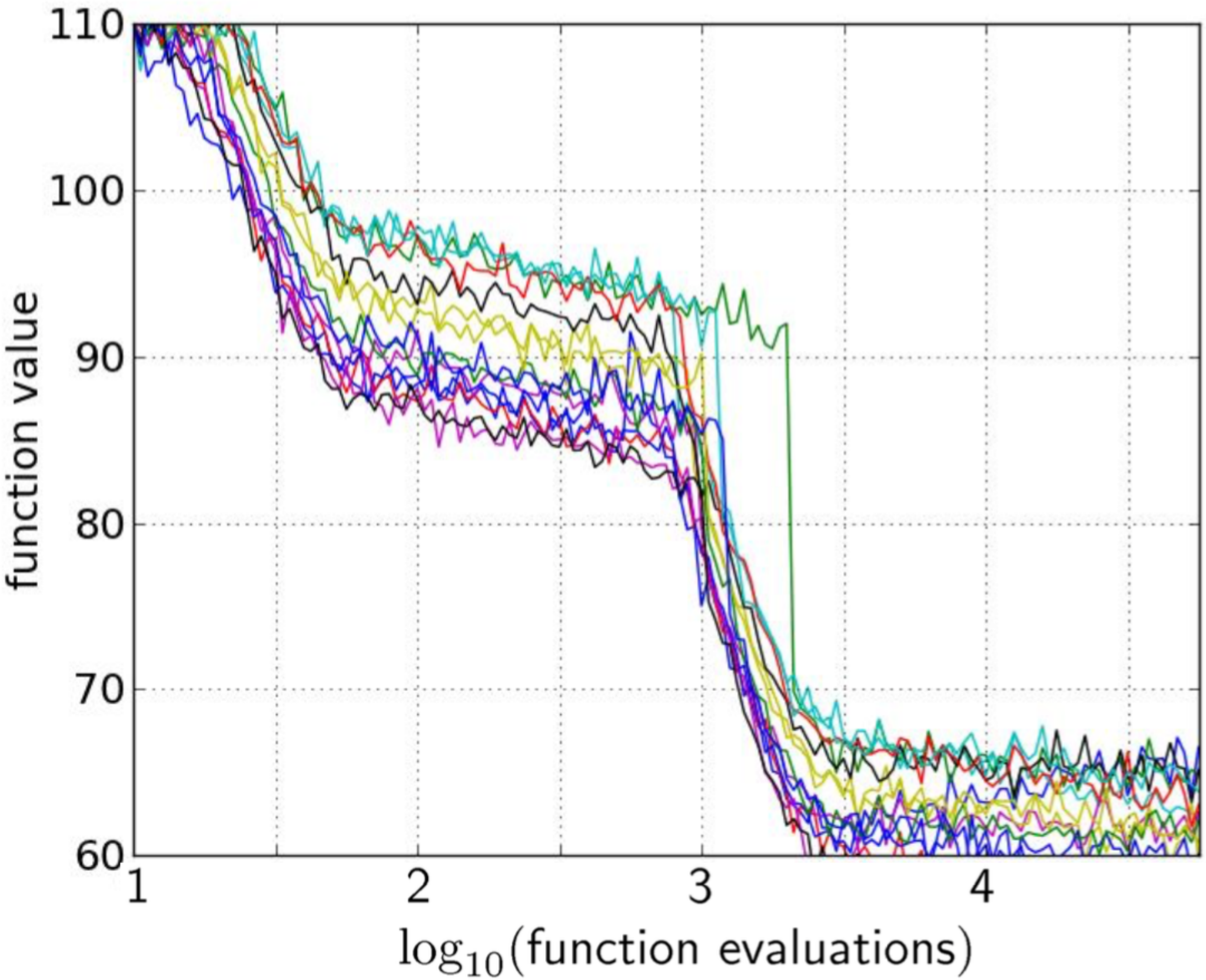
A Convergence Graph



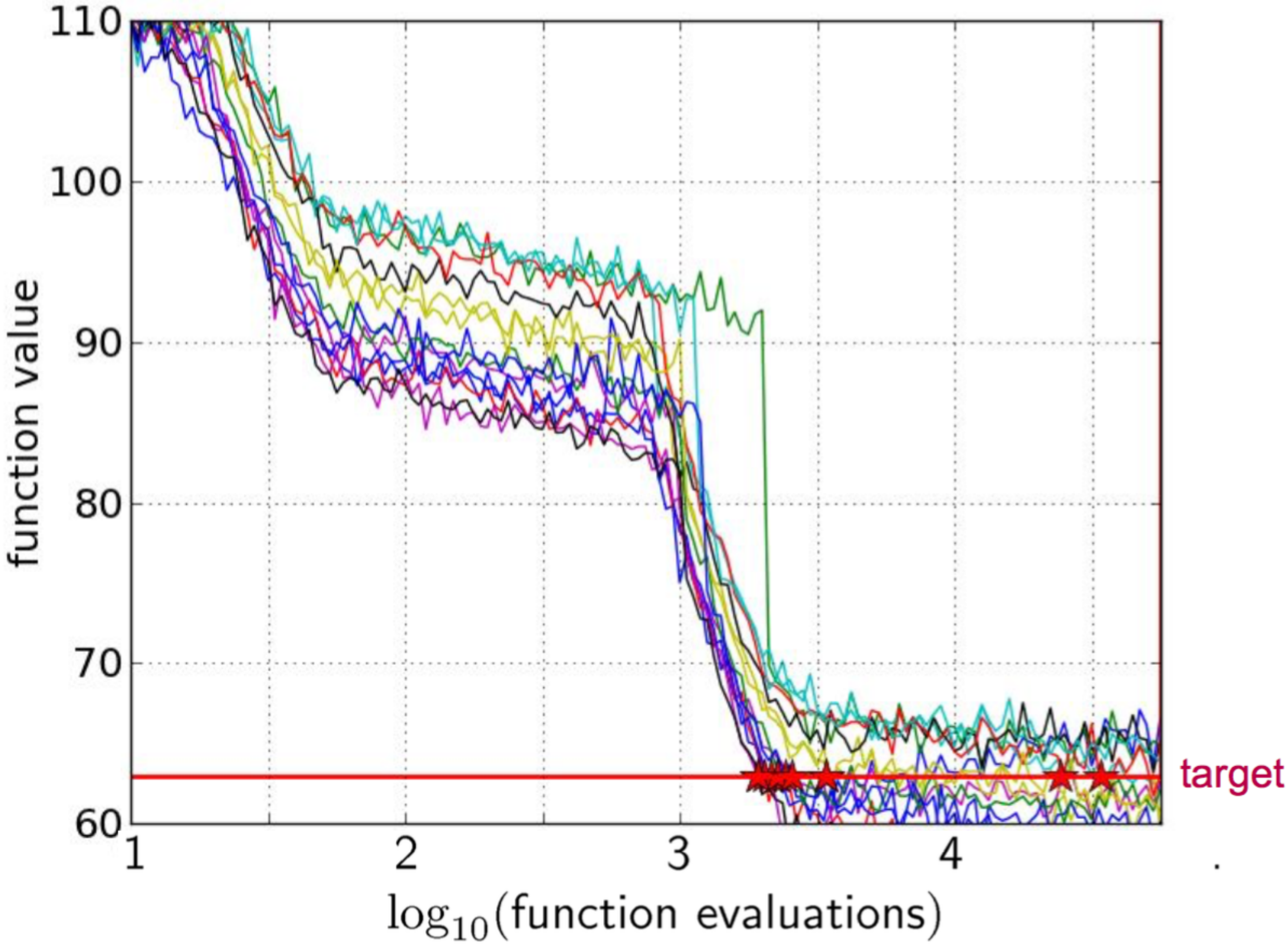
First Hitting Time is Monotonous



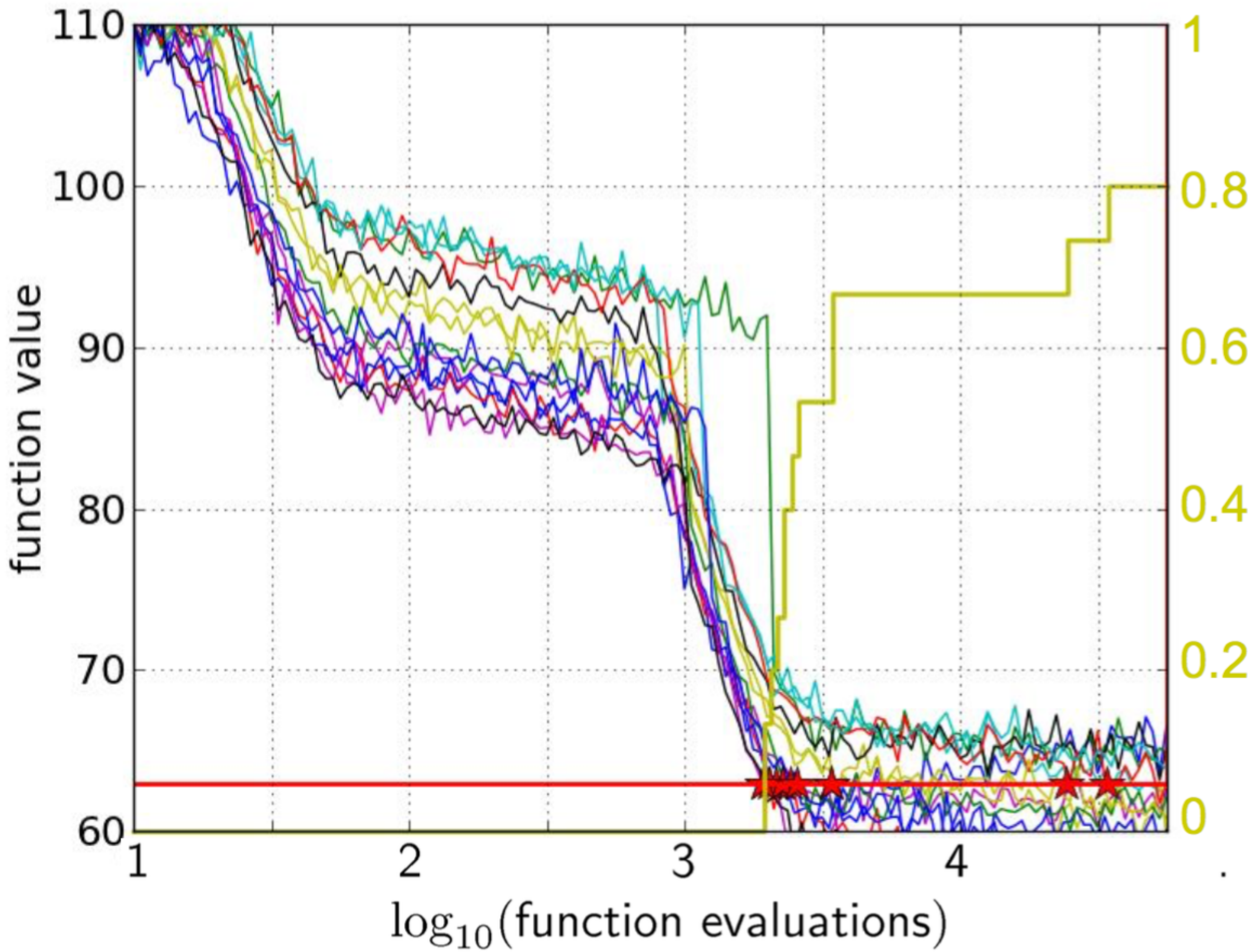
15 Runs


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



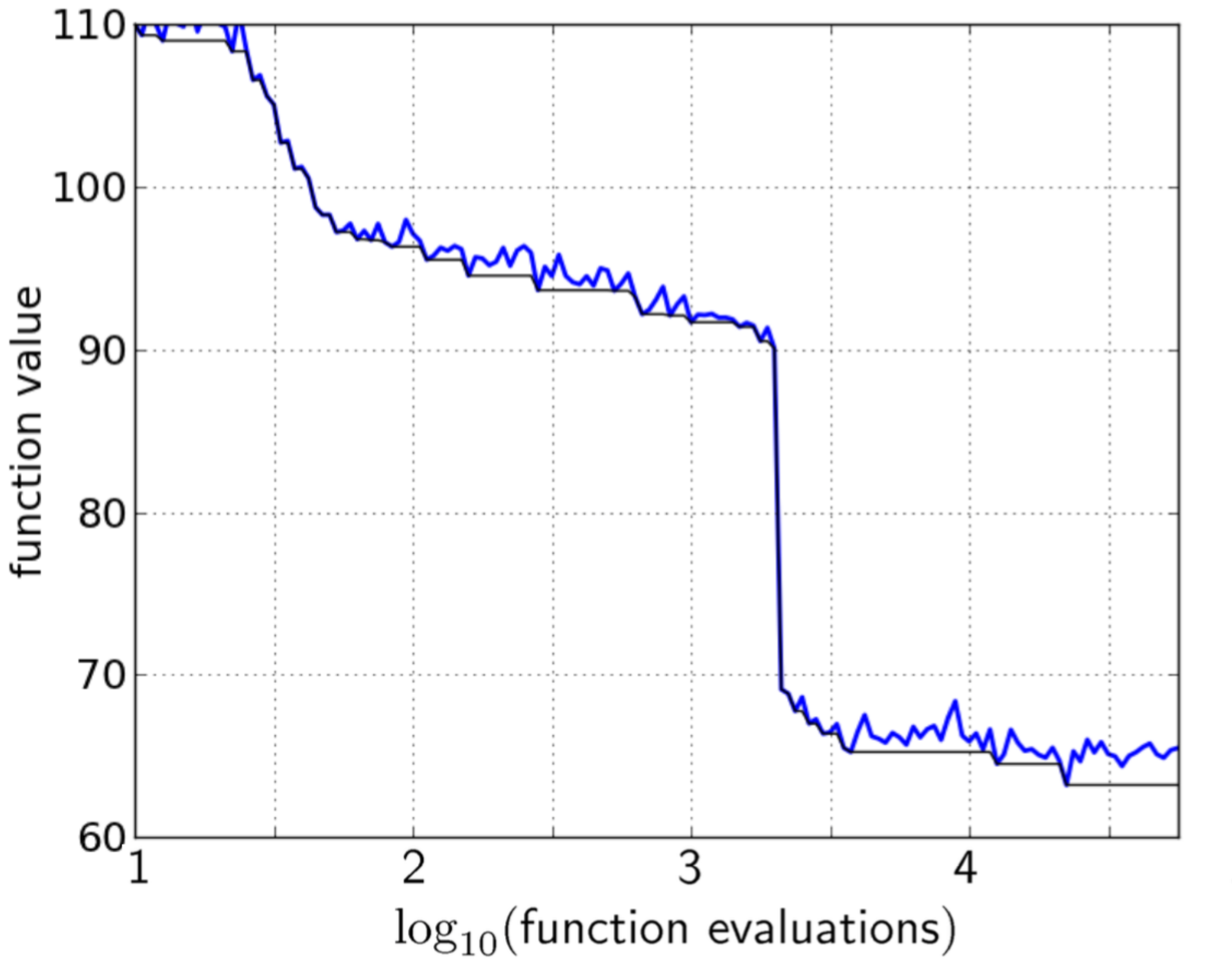
the ECDF of run lengths to reach the target

- 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)

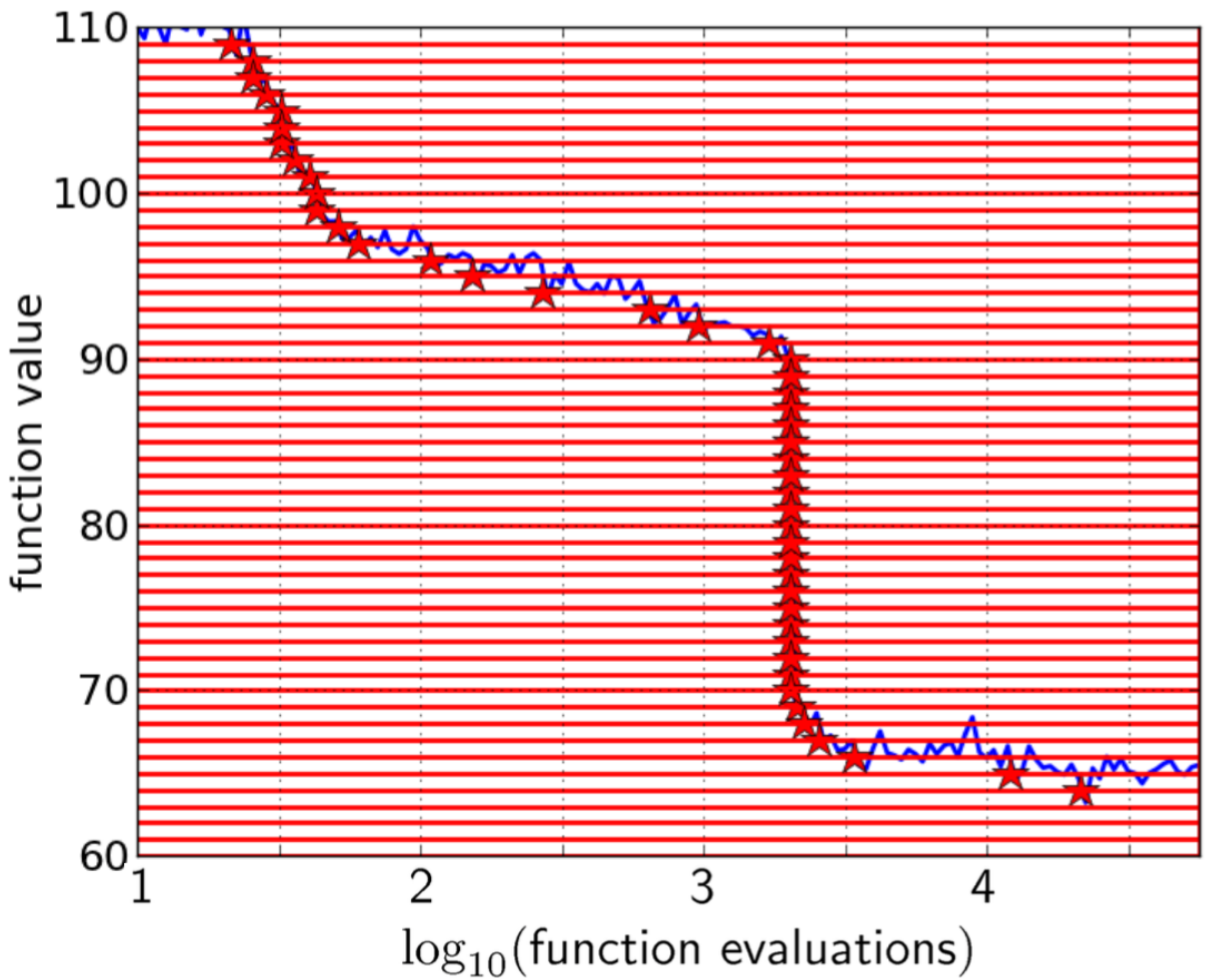
e.g. 60% of the runs need between 2000 and 4000 evaluations
80% of the runs reached the target

Reconstructing a single run via ECDF…

Reconstructing A Single Run

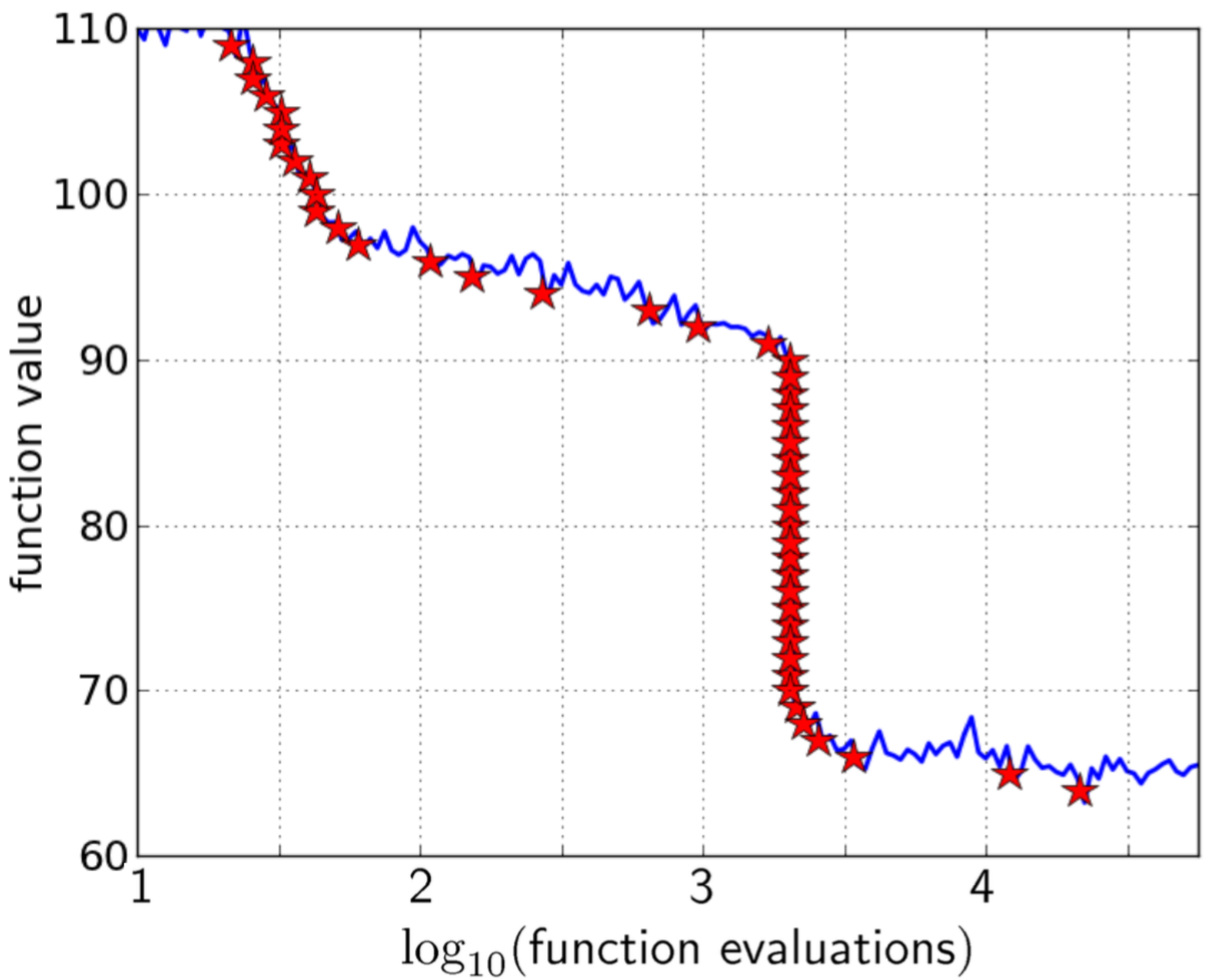


Reconstructing A Single Run

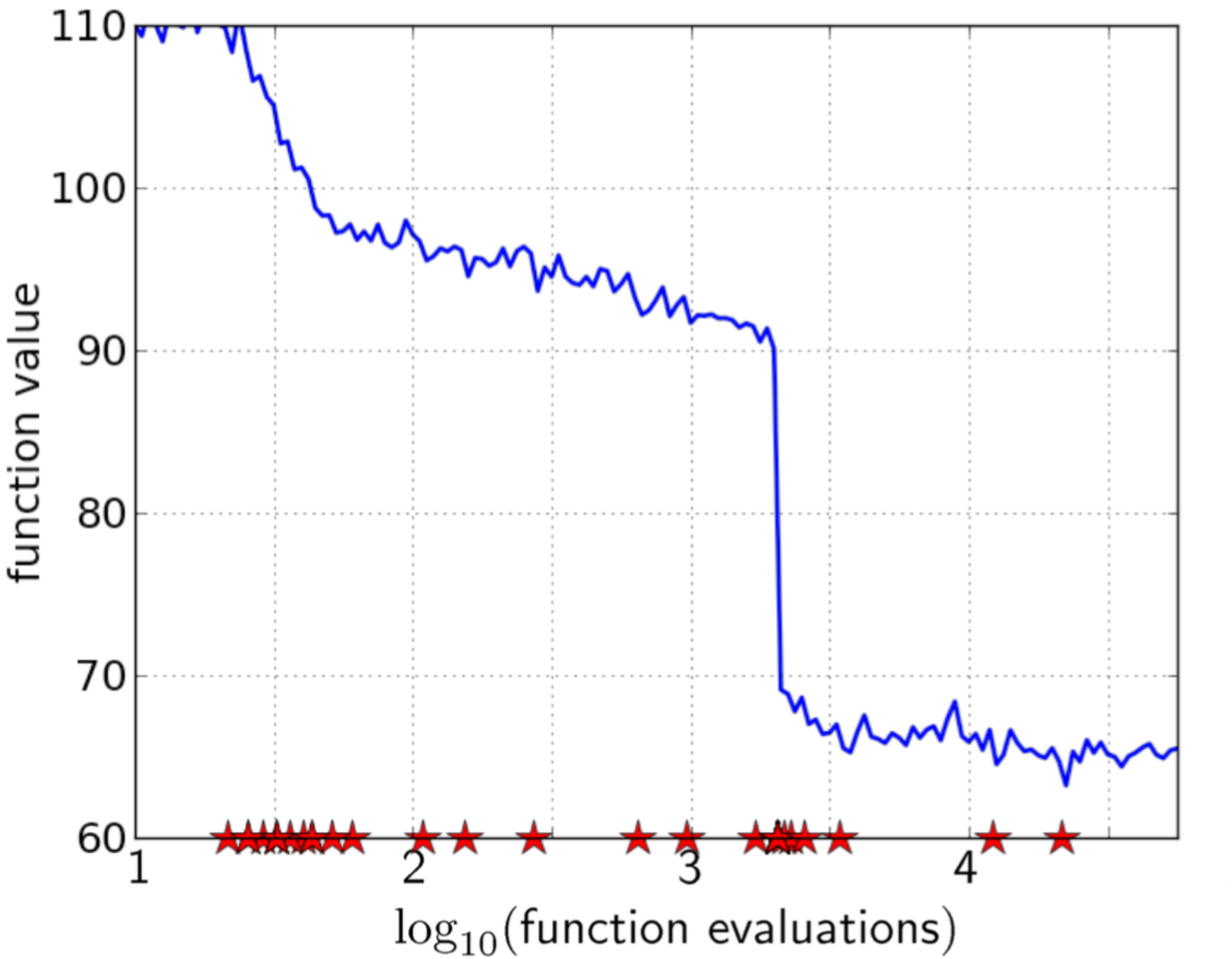


50 equally spaced targets

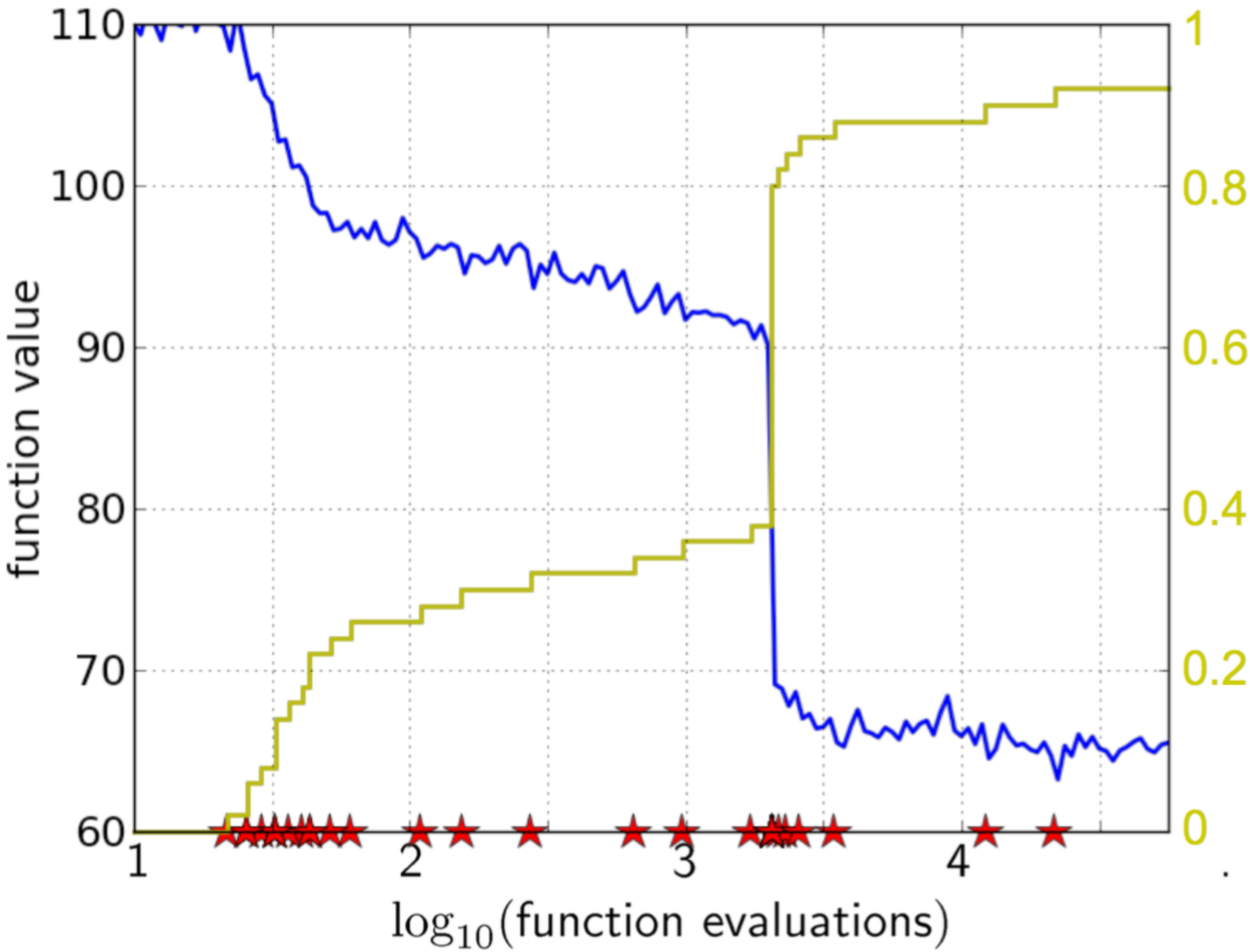
Reconstructing A Single Run



Reconstructing A Single Run

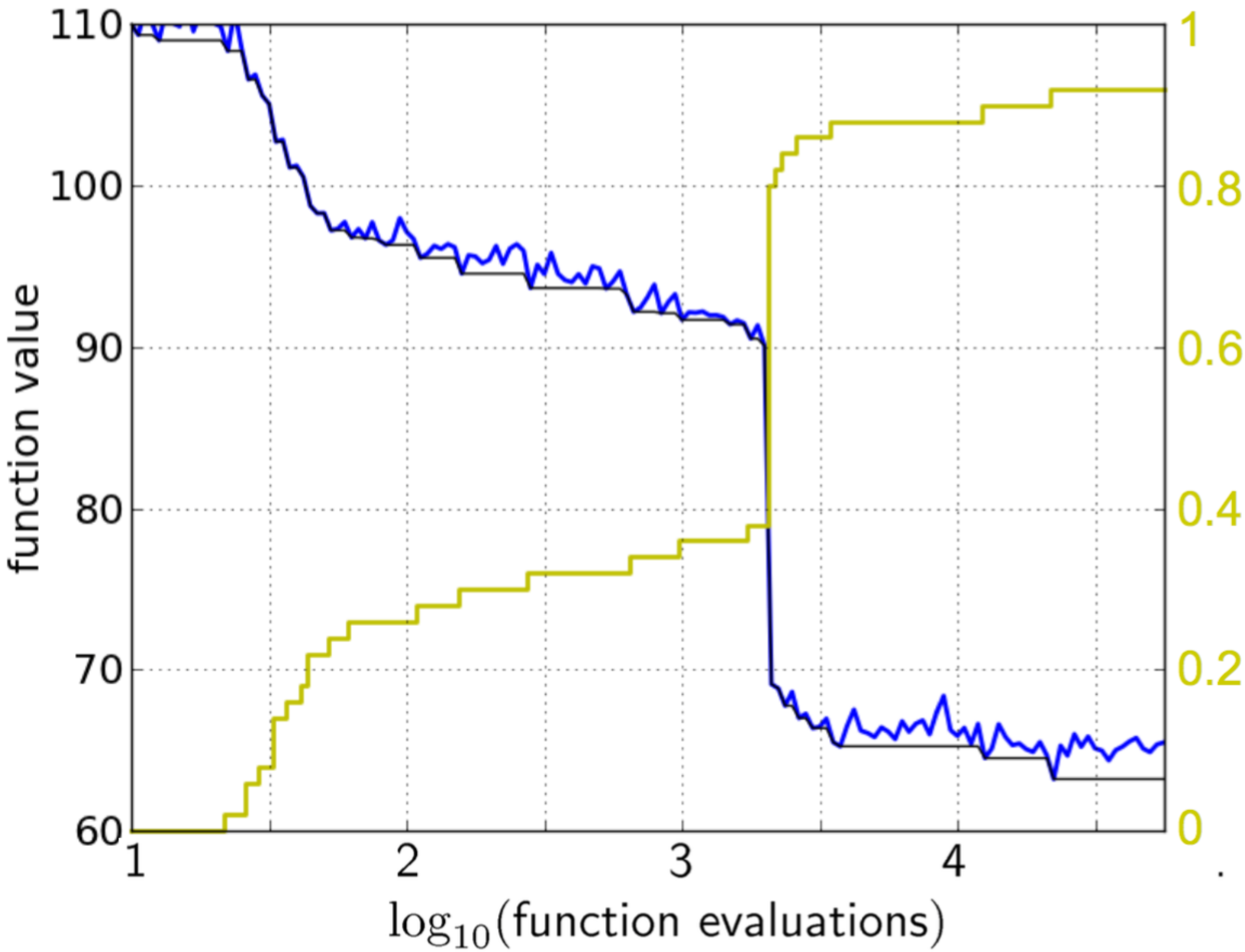


Reconstructing A Single Run



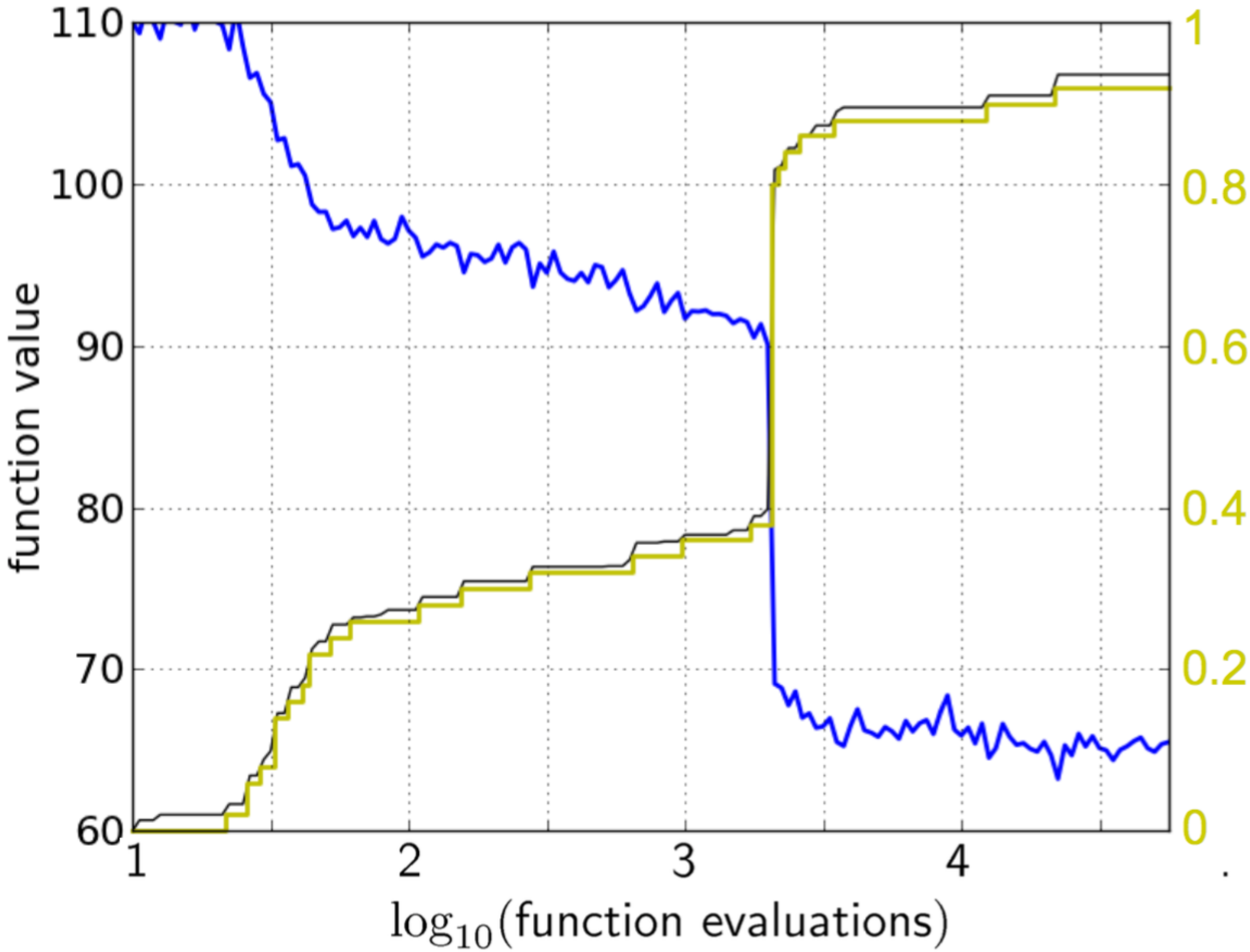
the empirical CDF makes a step for each star, is monotonous and displays for each budget the fraction of targets achieved within the budget

Reconstructing A Single Run



the ECDF recovers
the monotonous
graph,
discretised and
flipped

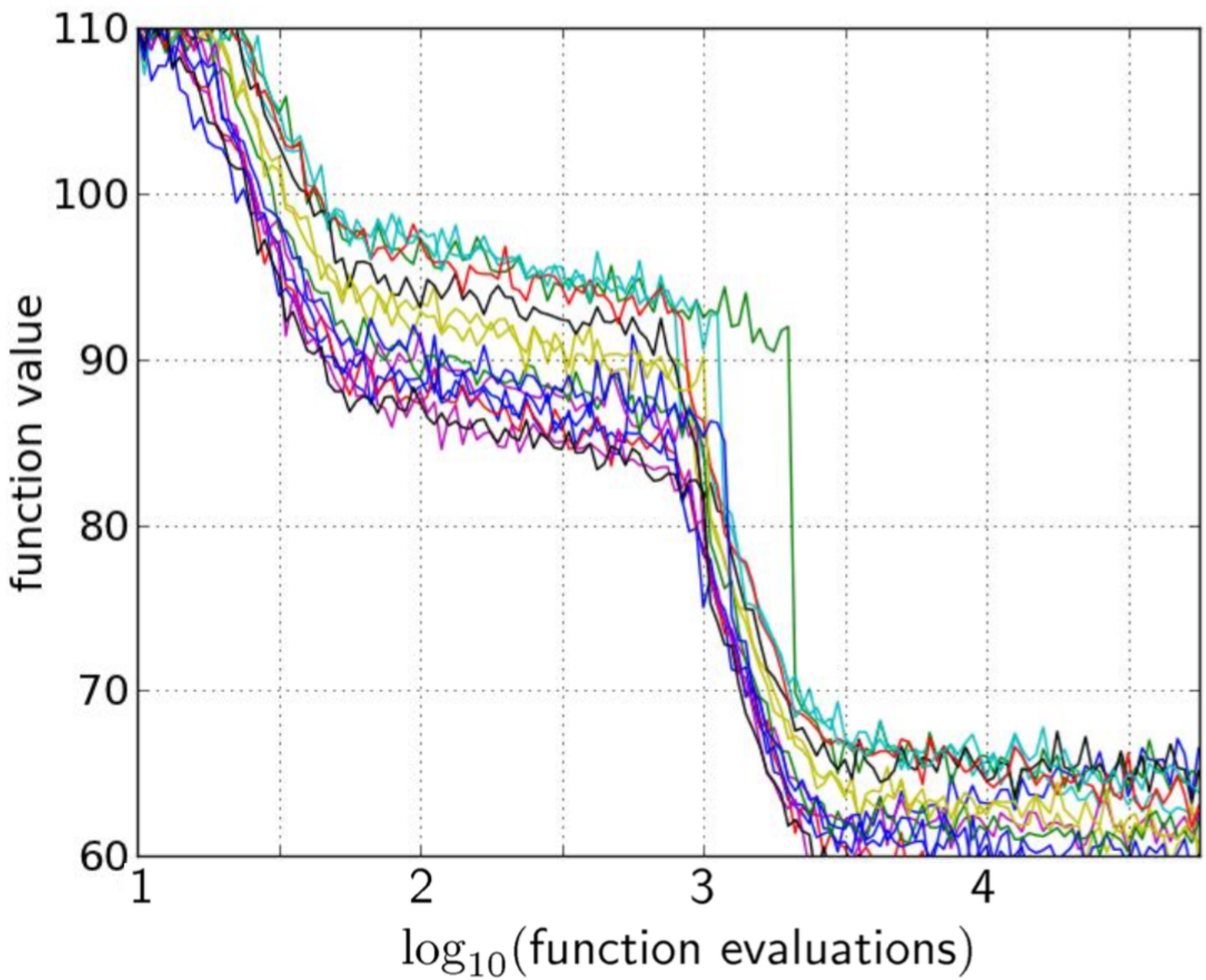
Reconstructing A Single Run



the ECDF recovers the monotonous graph, discretised and flipped

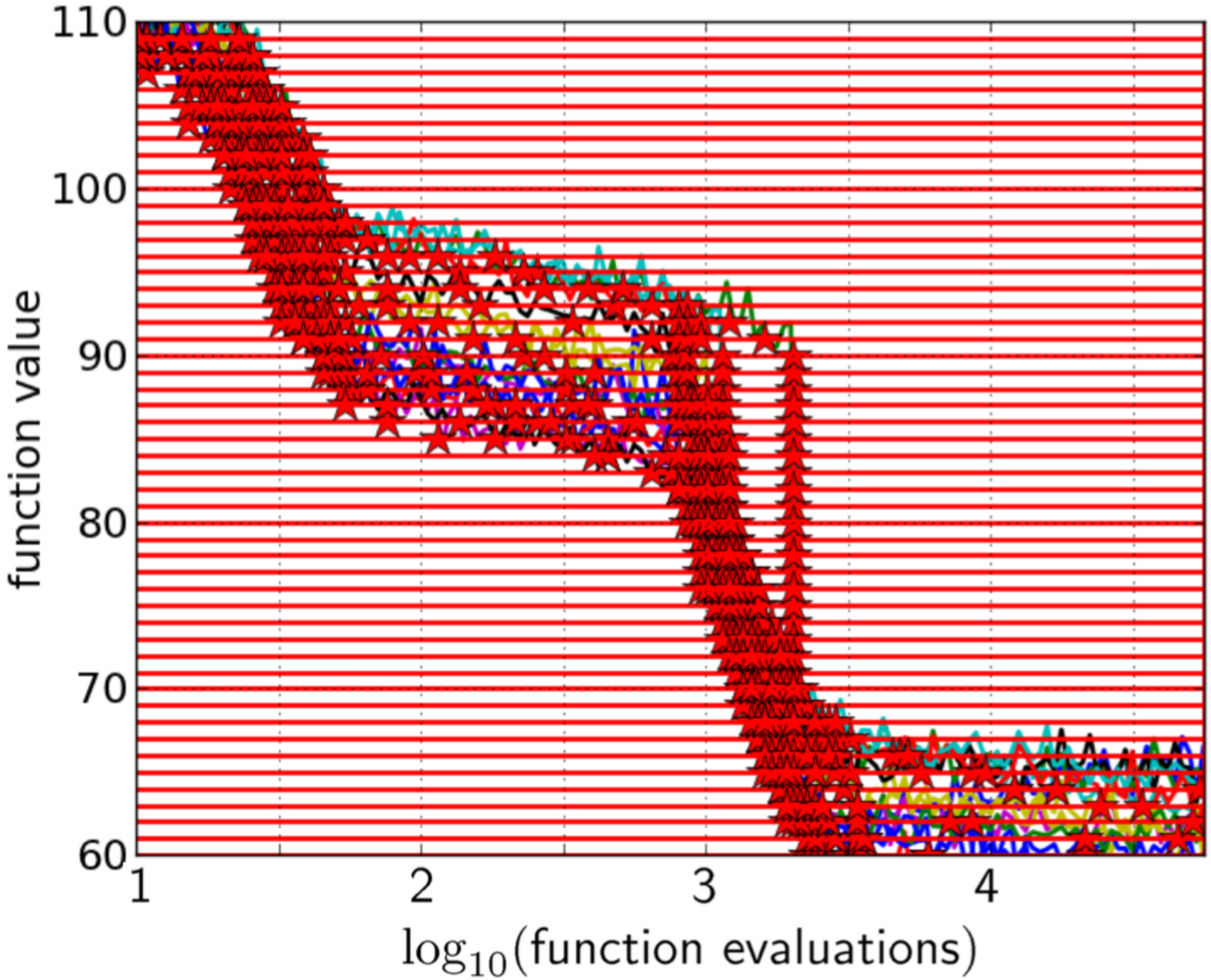
Aggregation …

Aggregation



15 runs

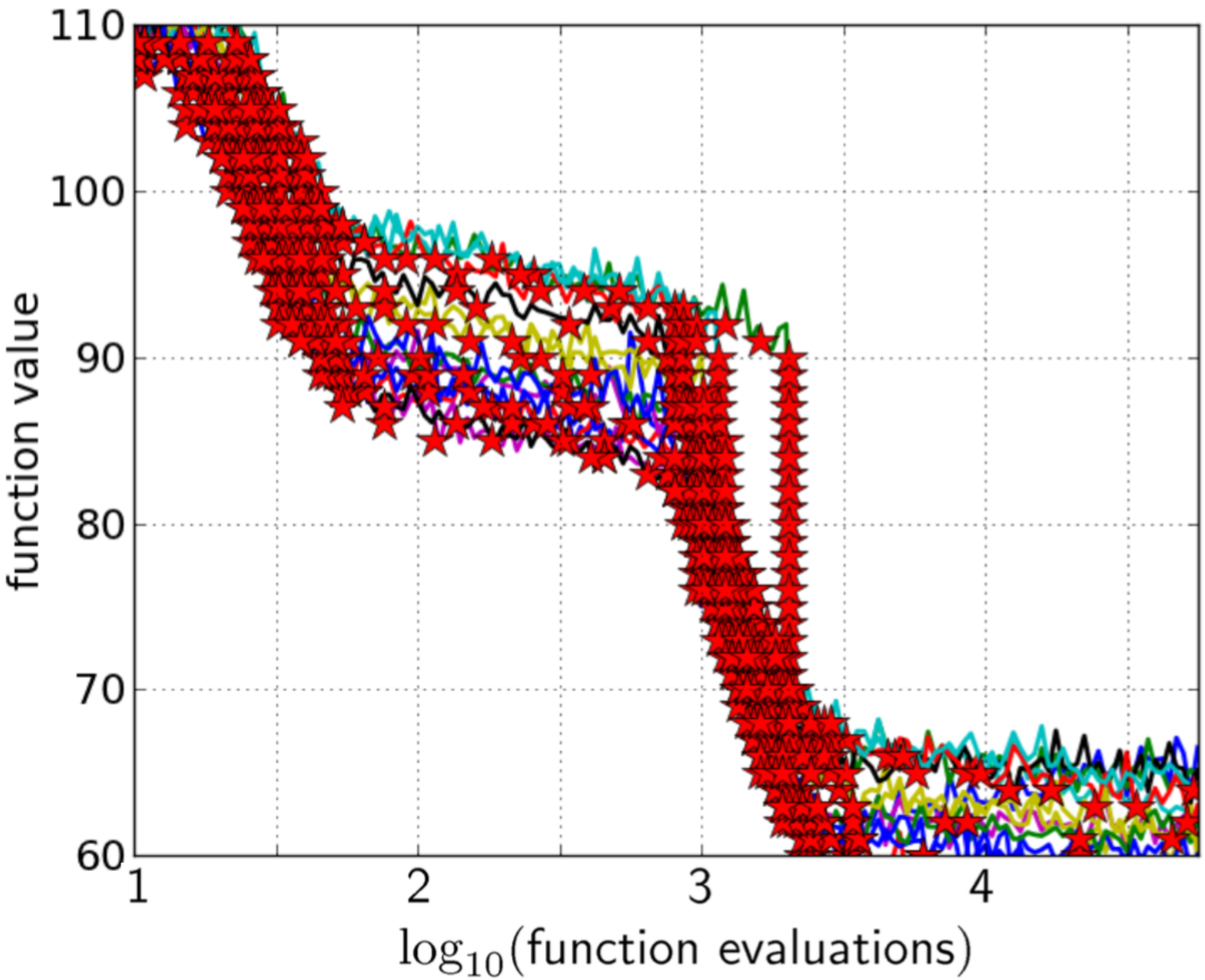
Aggregation



15 runs

50 targets

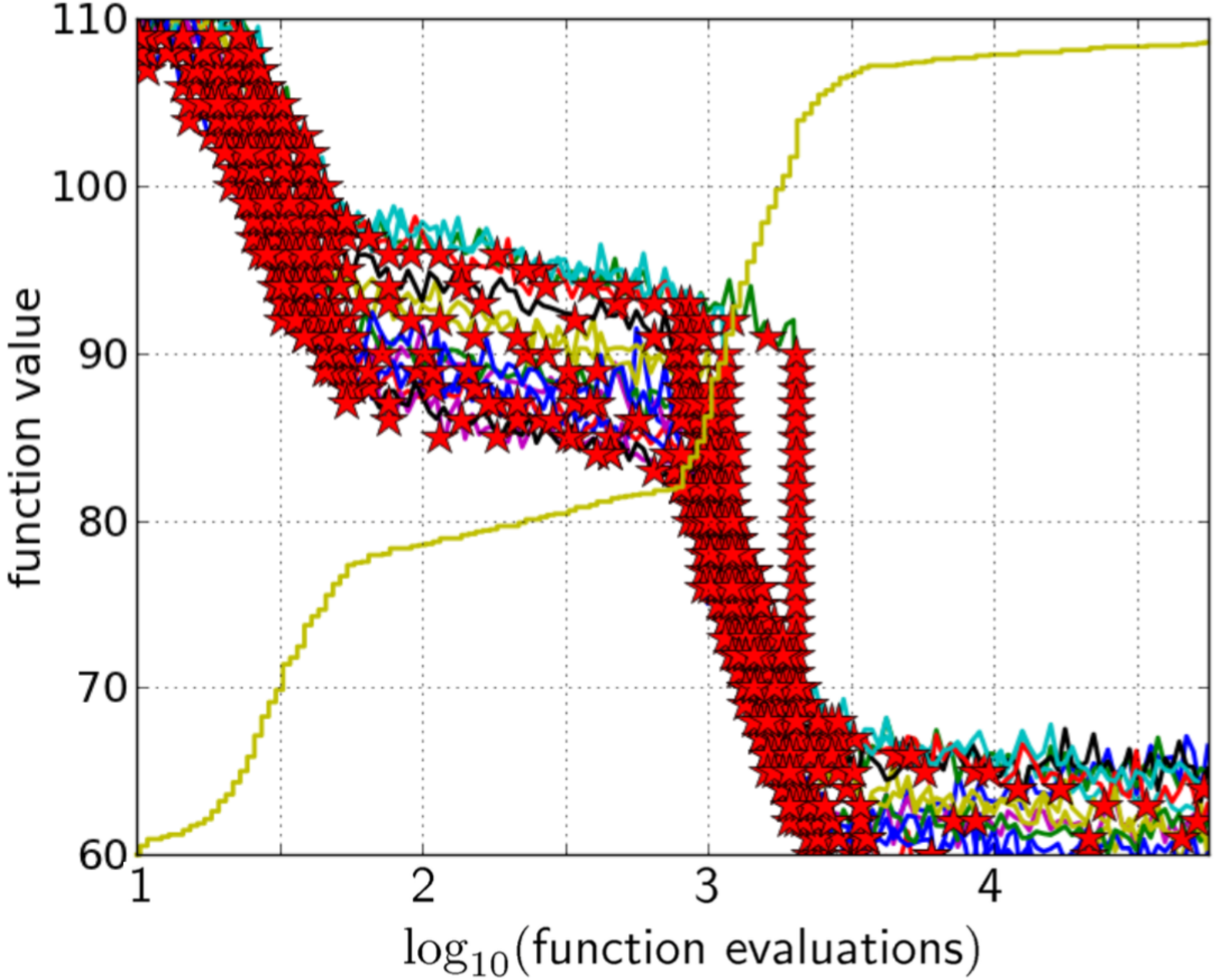
Aggregation



15 runs

50 targets

Aggregation

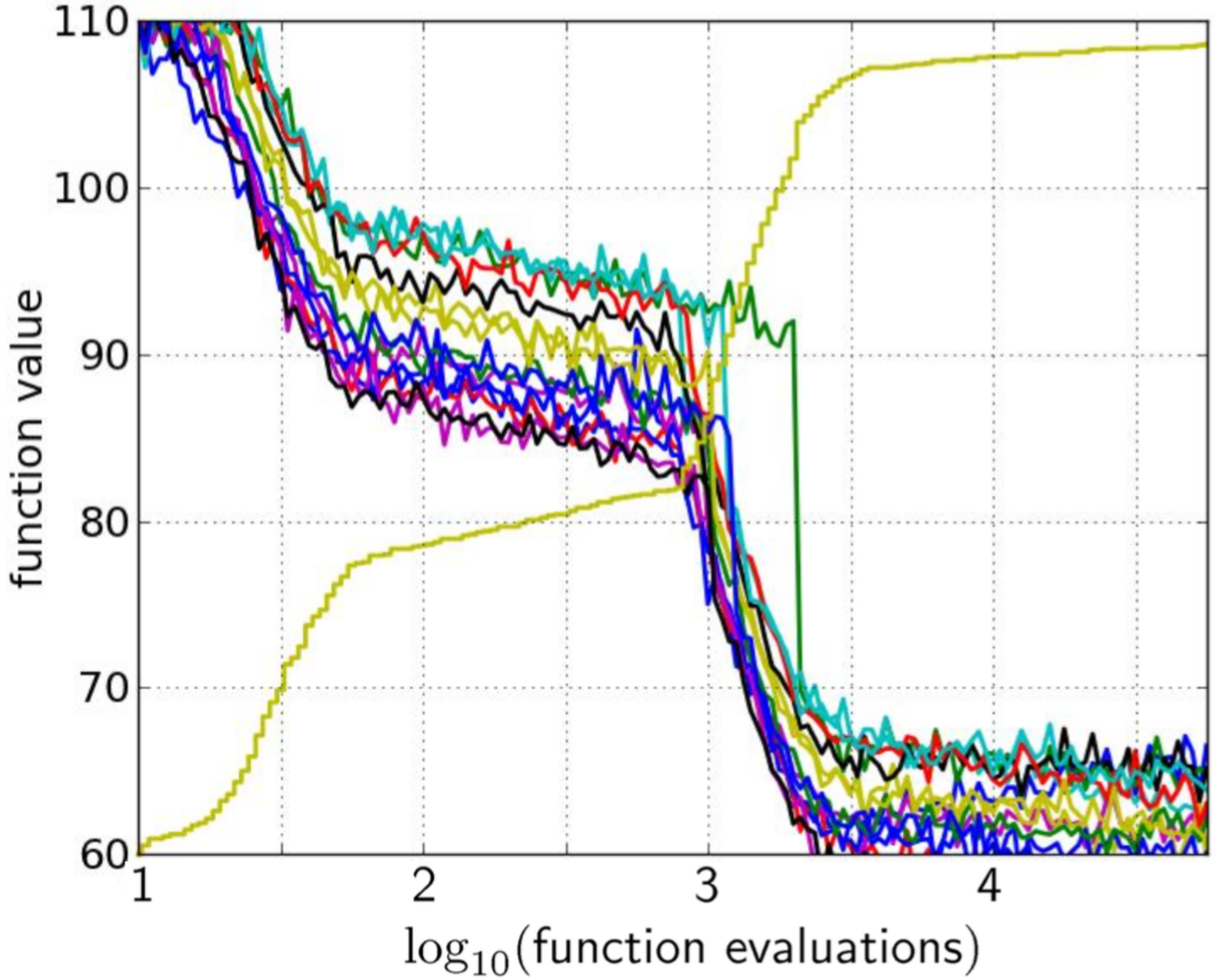


15 runs

50 targets

ECDF with 750 steps

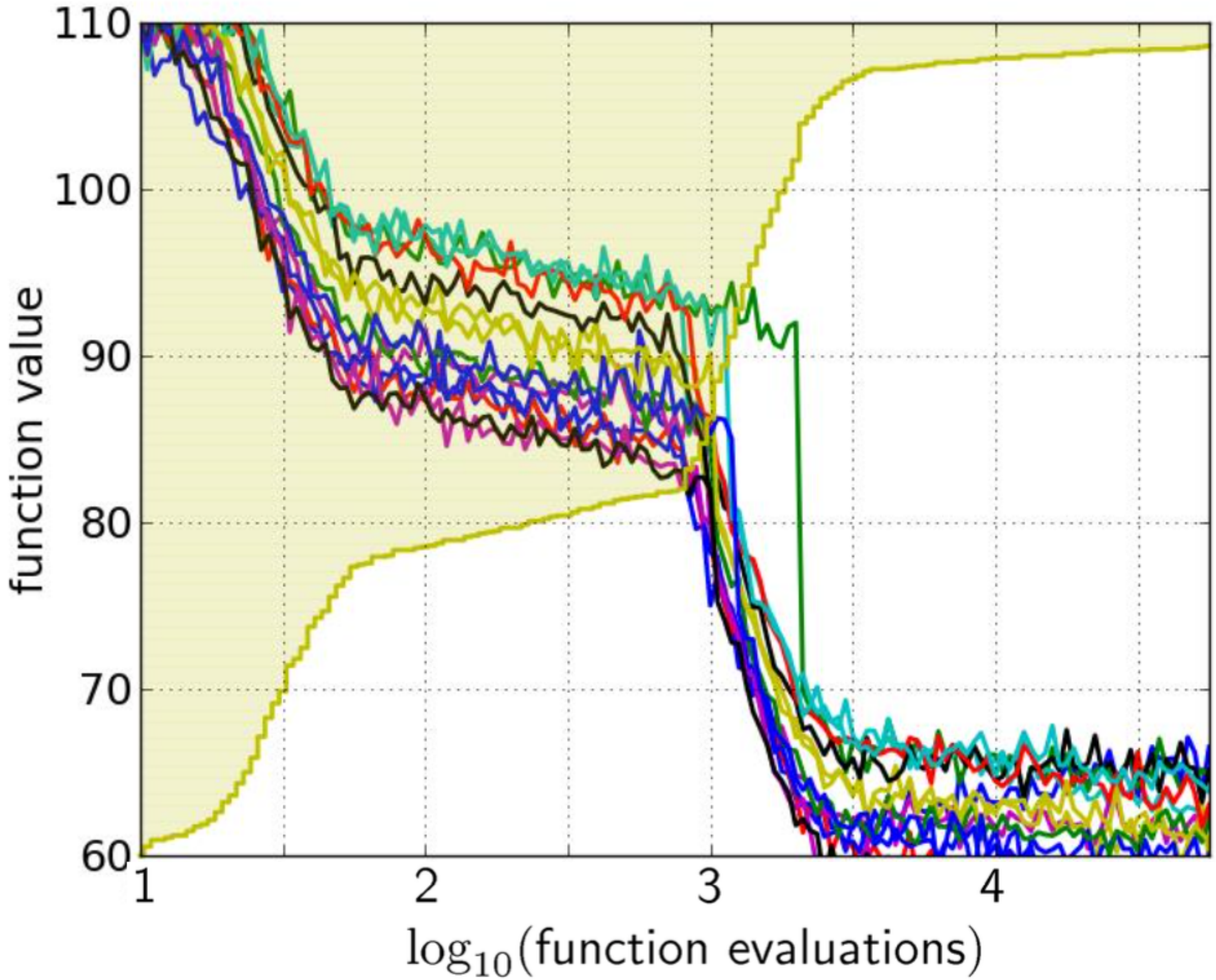
Aggregation



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

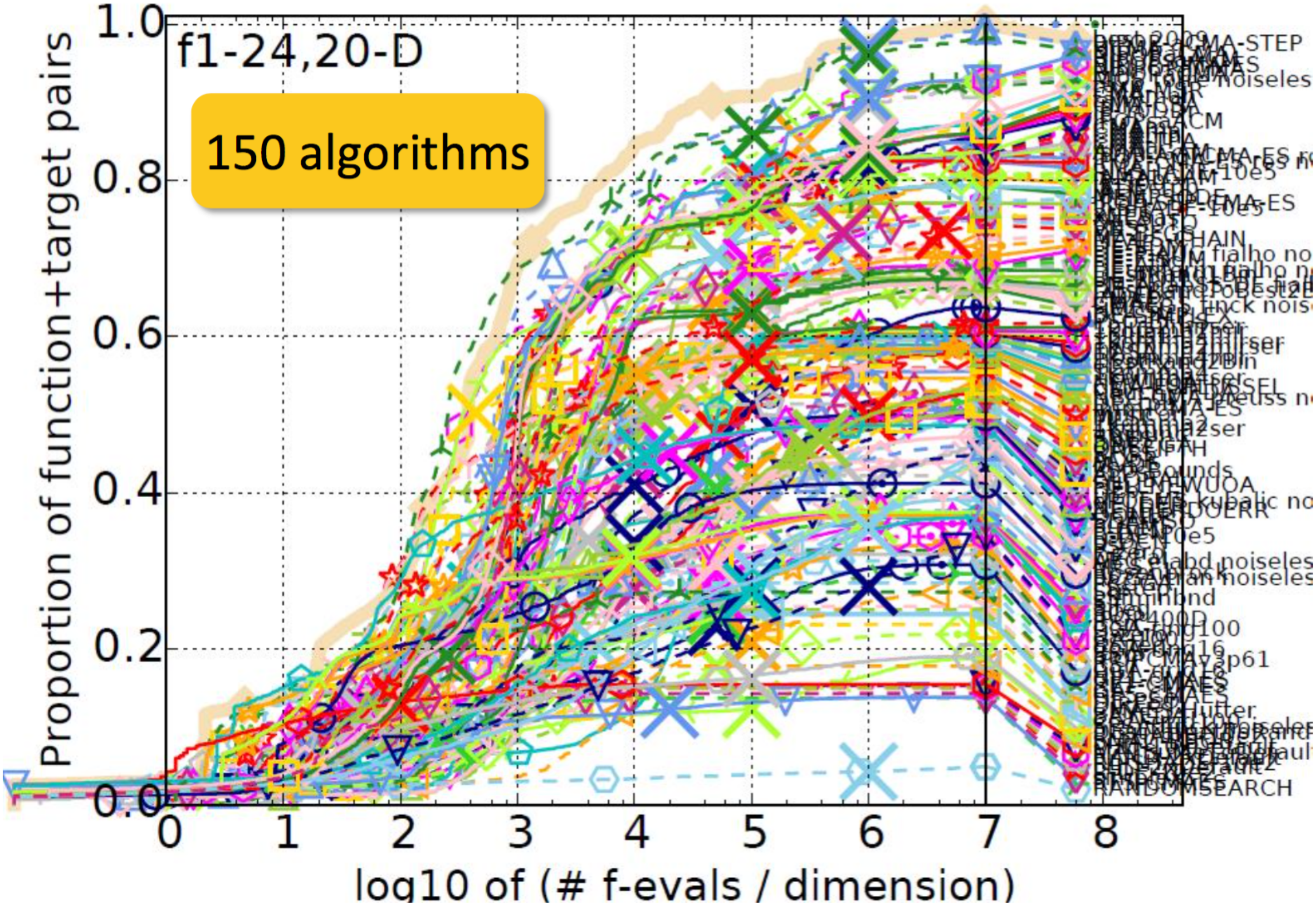
Aggregation

over runs

over test functions

over targets

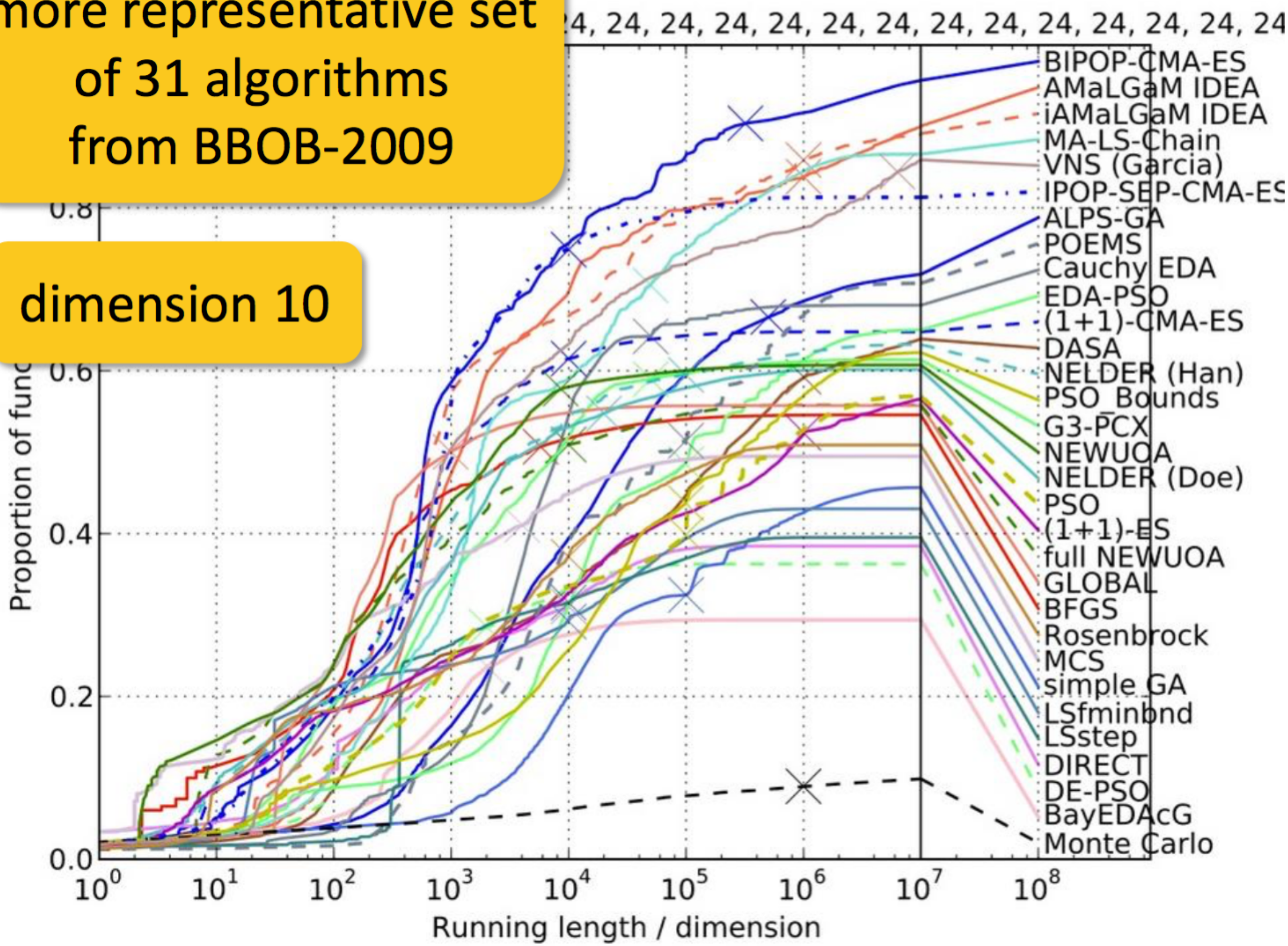
not over dimension



f1-24,20-D
150 algorithms
Proportion of function+target pairs
log10 of (# f-evals / dimension)
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
5
6
7
8
RANDOMSEARCH

more representative set of 31 algorithms from BBOB-2009

dimension 10



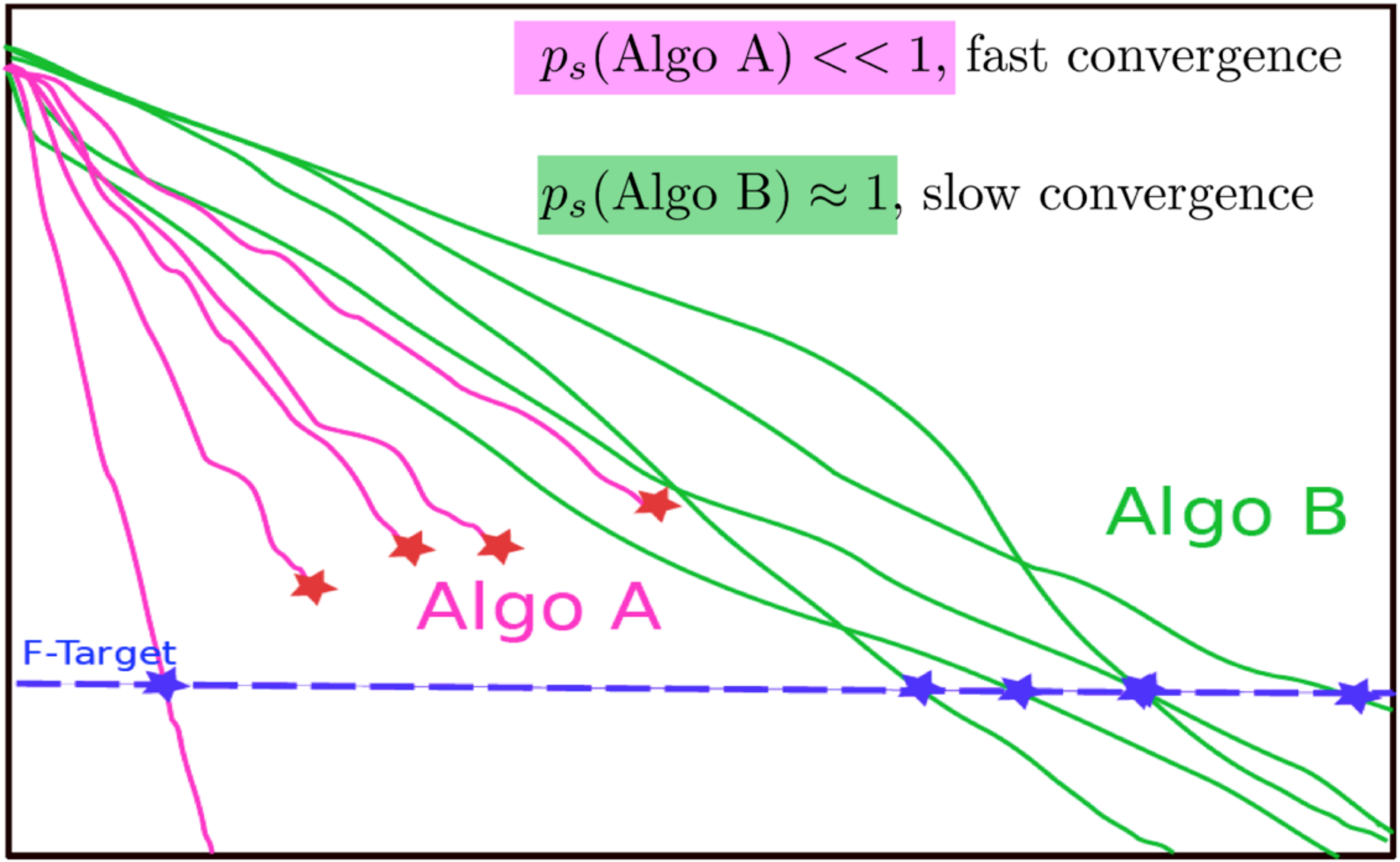
Displaying Performance

ECDF

Average RunTime (ART)

Which performance measure ?

to compare the two following scenario?



Which performance measure ?

Algo Restart A:

$$p_s(\text{Algo Restart A}) = 1$$

Algo Restart B:

$$p_s(\text{Algo Restart B}) = 1$$

Expected Running Time (restart algo)

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

Estimator for ERT

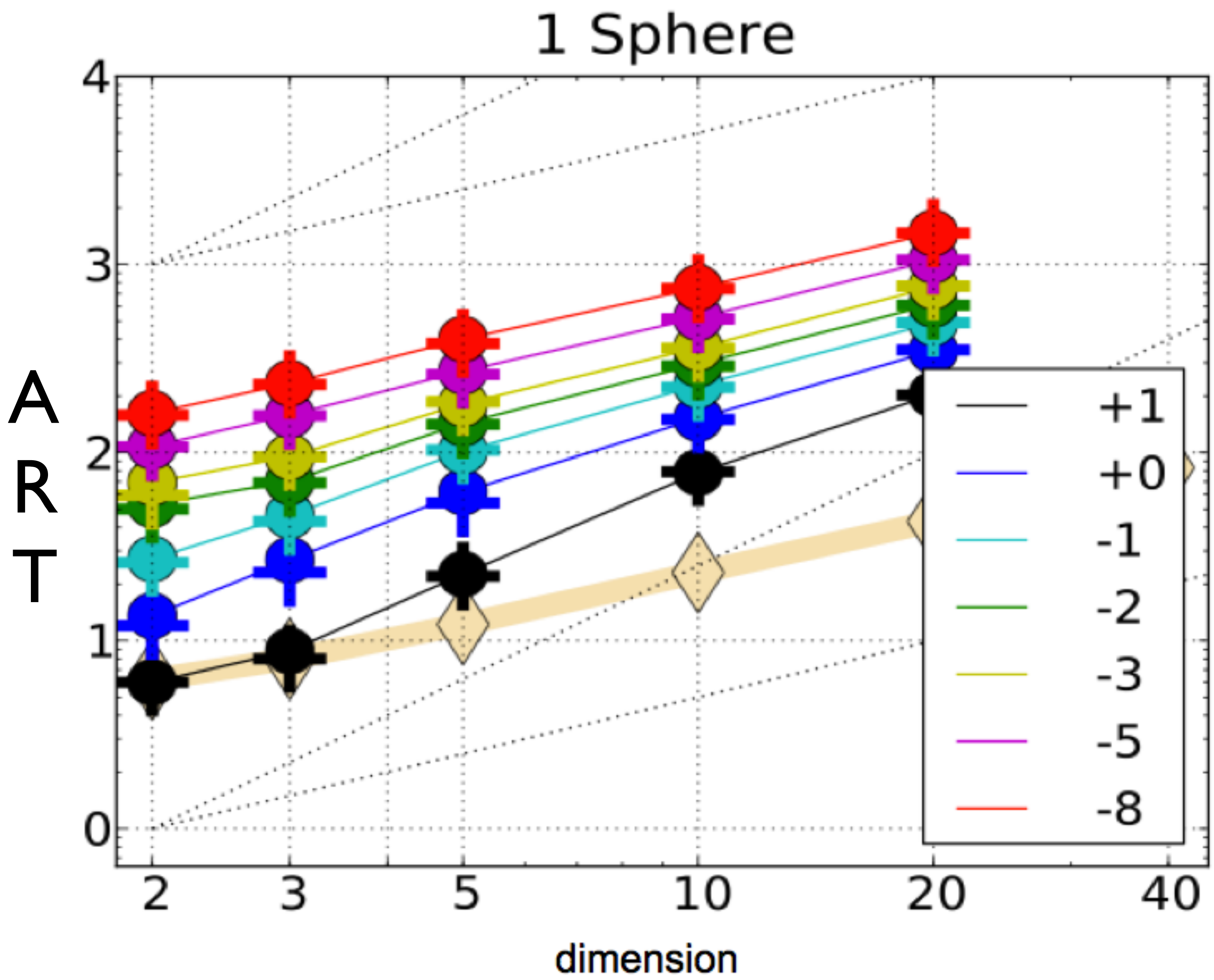
$$\widehat{p_s} = \frac{\#\mathrm{succ}}{\#Runs}$$

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

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

$$\mathrm{ART} = \frac{\#\mathrm{Evals}}{\#\mathrm{success}}$$

Example: scaling behavior



ART on f1 of a variant of CMA-ES – linear scaling

Automatizing the benchmarking COCO platform

COCO platform - COmparing Continuous Optimizers

https://github.com/numbbo/coco