Performance Evaluation of Anytime Black Box Optimizers

Black-Box Optimization (Search)

Minimize an objective function (also: cost, loss, error, or fitness function)

$$f : \mathcal{X} \subset \mathbb{R}^n \to \mathbb{R}, \quad x \mapsto f(x)$$

in a black-box scenario (direct search, no gradients)

$$x \longrightarrow \quad \blacksquare \quad \longrightarrow f(x)$$

where the black box can be

- non-linear, non-convex, discontinuous, dynamic, stochastic
- from milli-seconds to hours to evaluate

Objective:

- convergence to a global essential infimum of f as fast as possible
- (informally, time-finite) find $x \in \mathcal{X}$ with small $f(x)$ value using as few back-box calls (function evaluations) as possible

Why Do We Need to Measure Performance?

- putting algorithms to a *standardized* test
  - simplify judgement
  - simplify comparison
  - regression test/quality check under algorithm changes

- algorithm selection

- understanding of algorithms

How do we measure performance?

Birds View

We can measure performance on

- real world problems
  - expensive
  - comparison is typically limited to certain domains
  - experts have limited interest to publish

- "artificial" benchmark functions
  - cheap
  - data acquisition is comparatively easy
  - problem of representativity

- caveat: parameter of algorithms

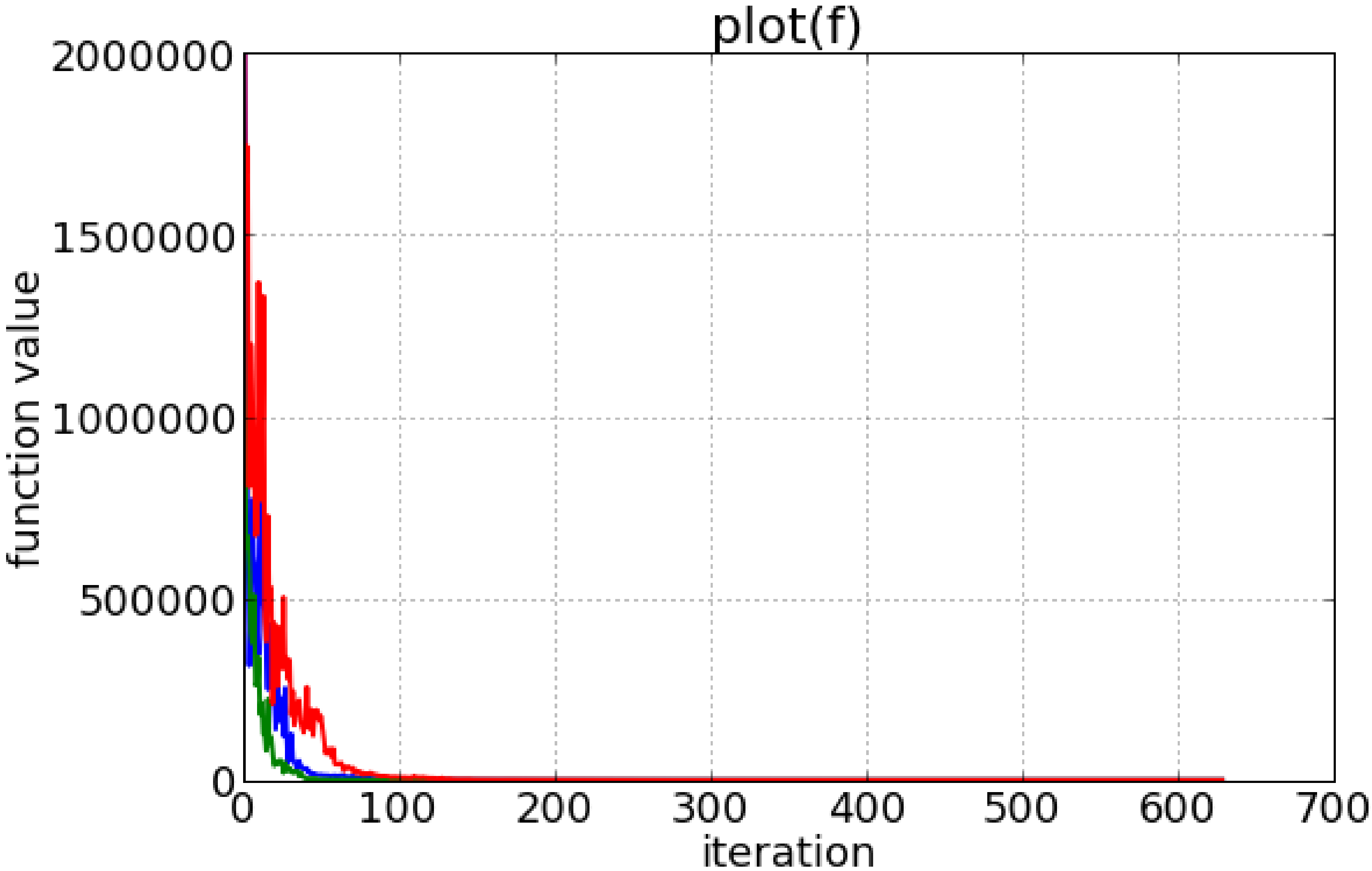
Measuring Performance

...empirically...

- convergence graphs is all we have to start with
- the right presentation cannot be overestimated

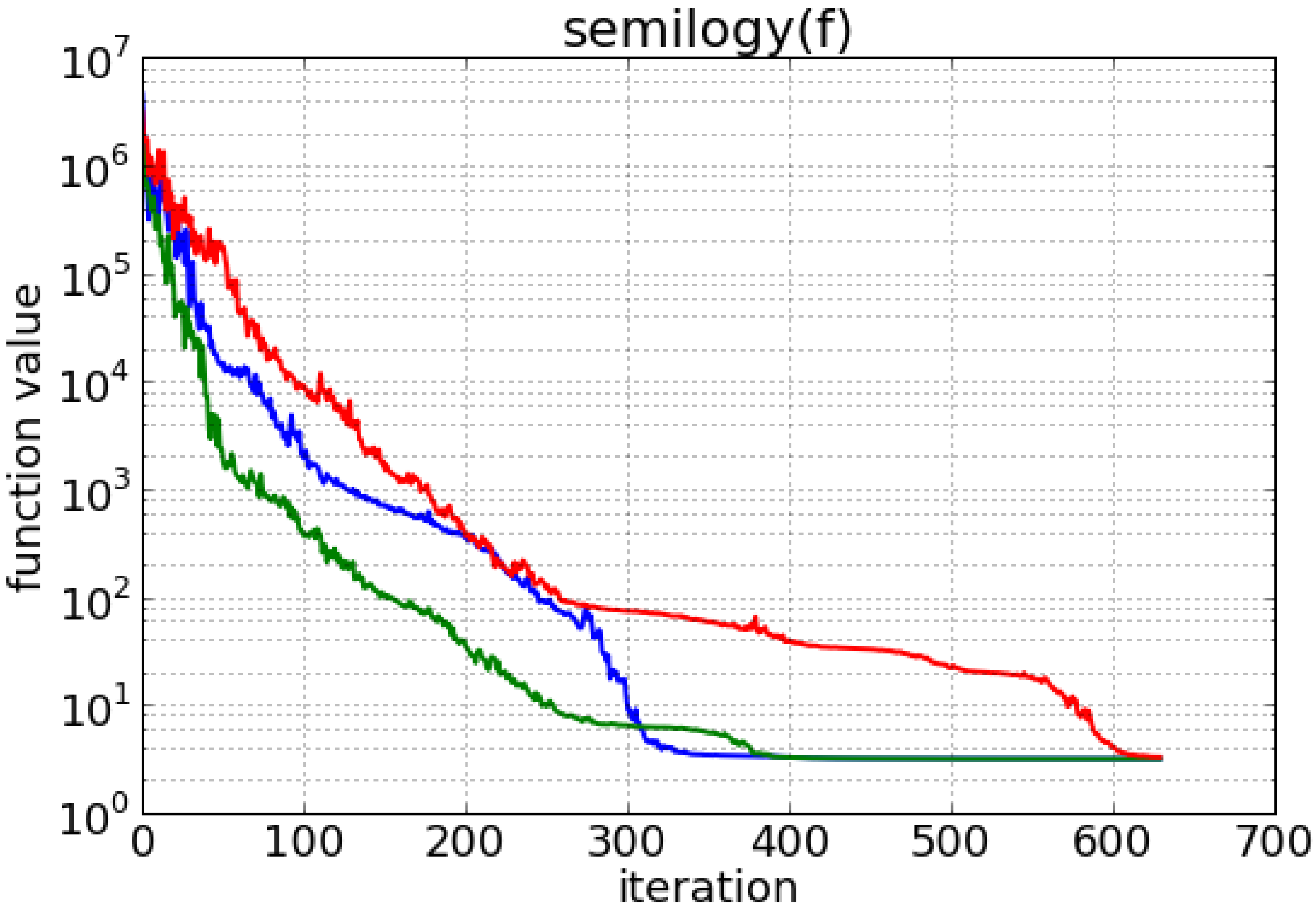
the details are important

Displaying Three Runs (three trials)



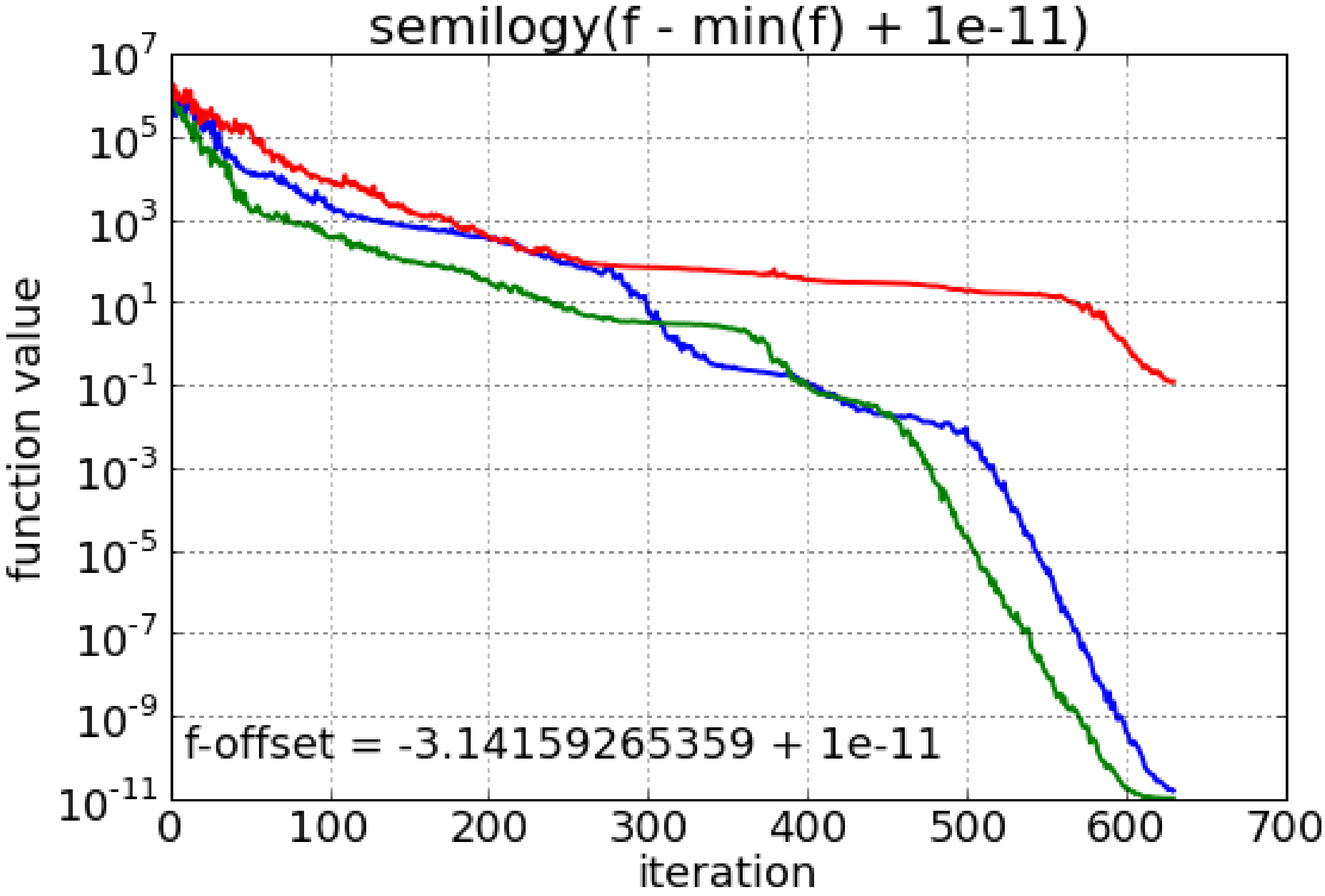
not like this (it's unfortunately a common picture)

Displaying Three Runs (three trials)



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

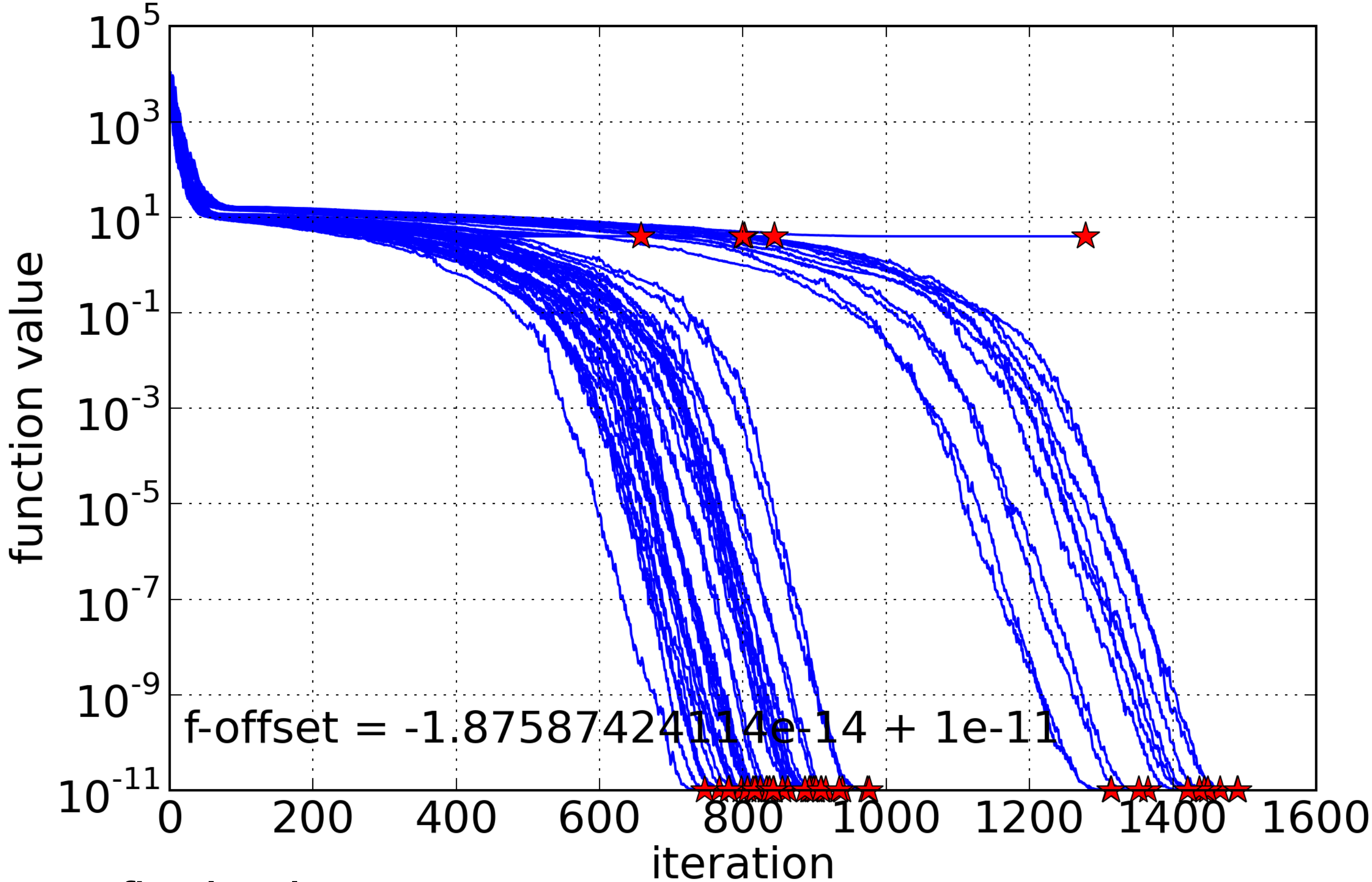
Displaying Three Runs (three trials)



even better like this: subtract minimum value over all runs

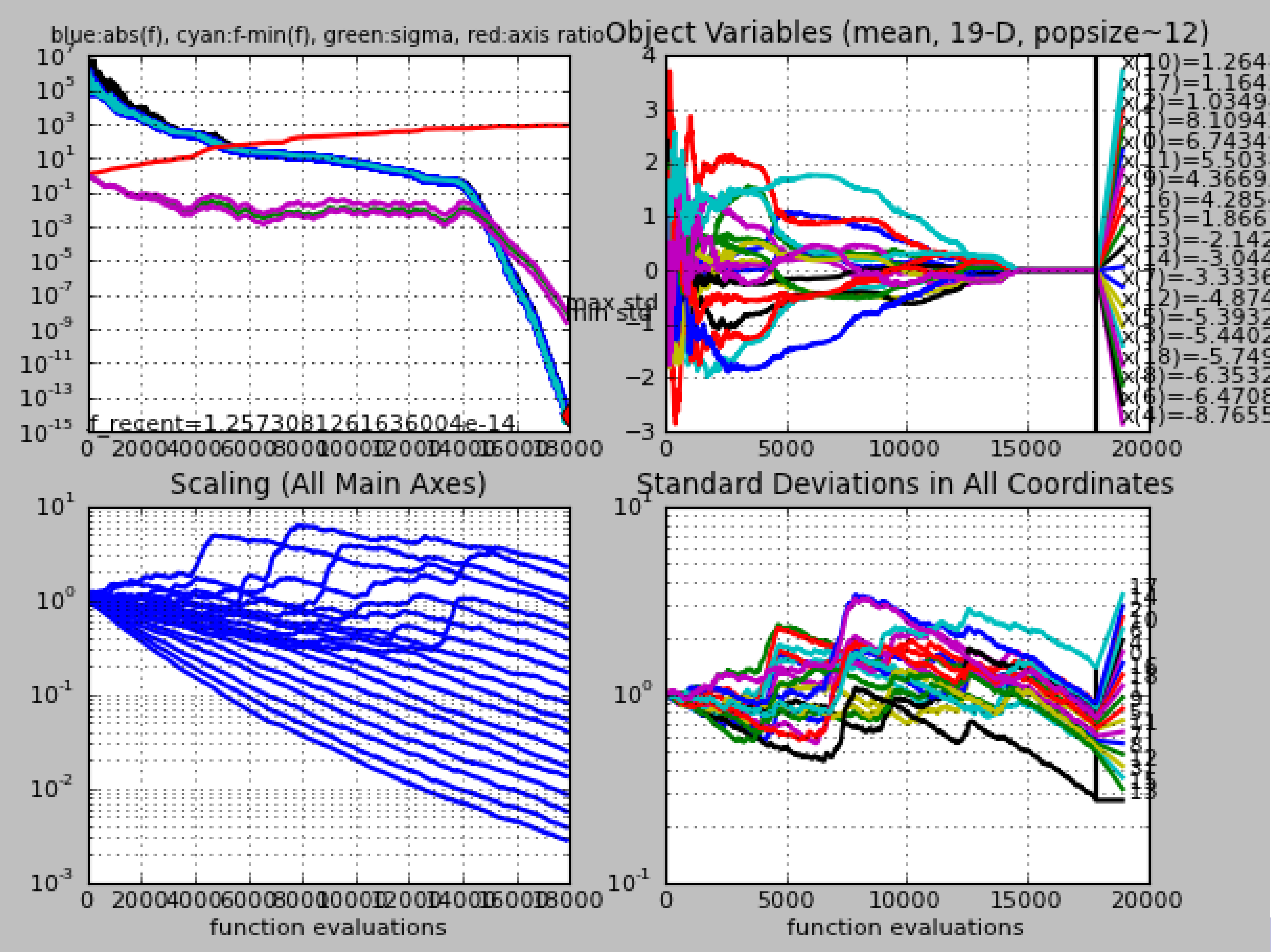
Displaying 51 Runs

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

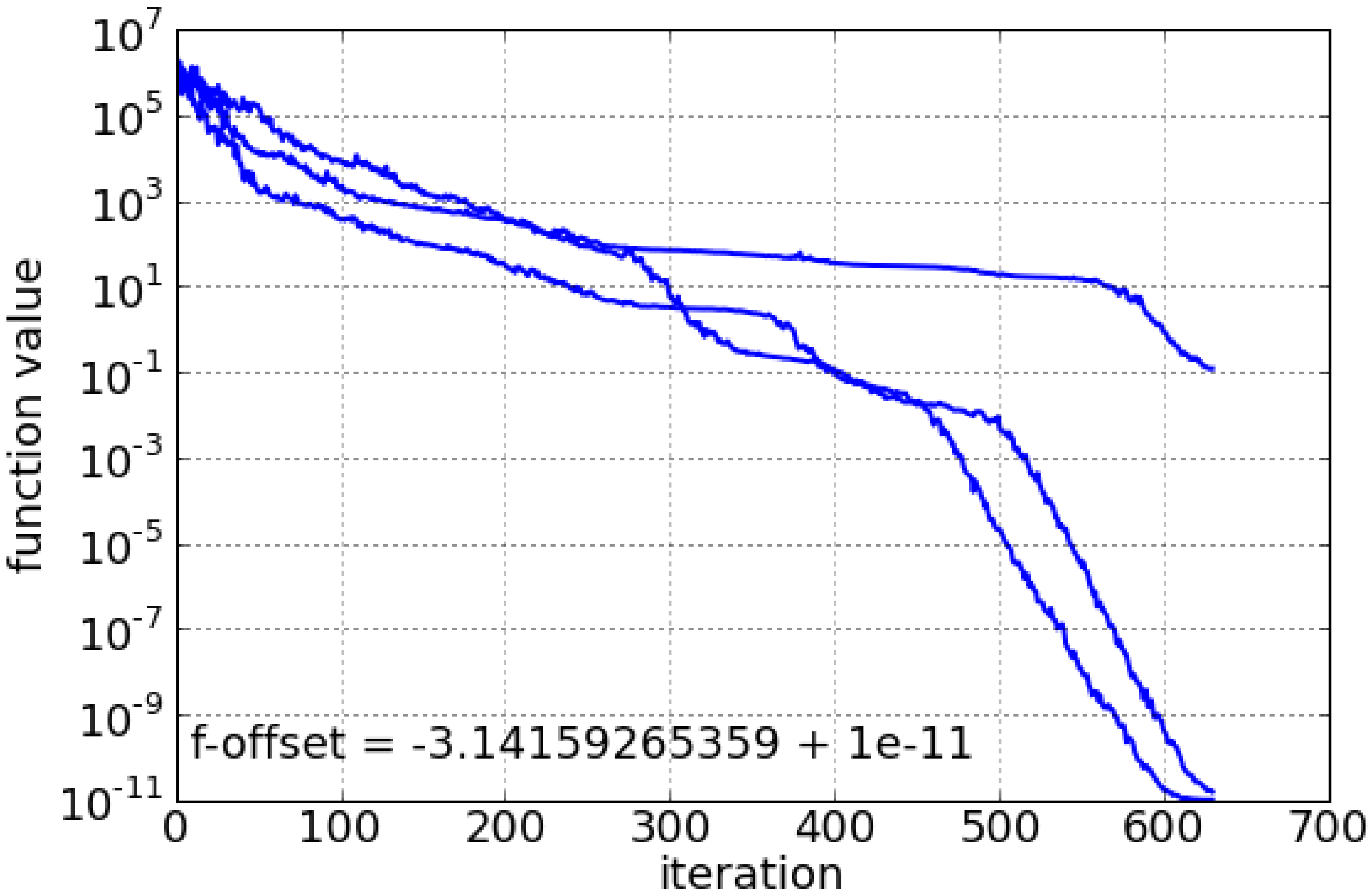


★ : final value

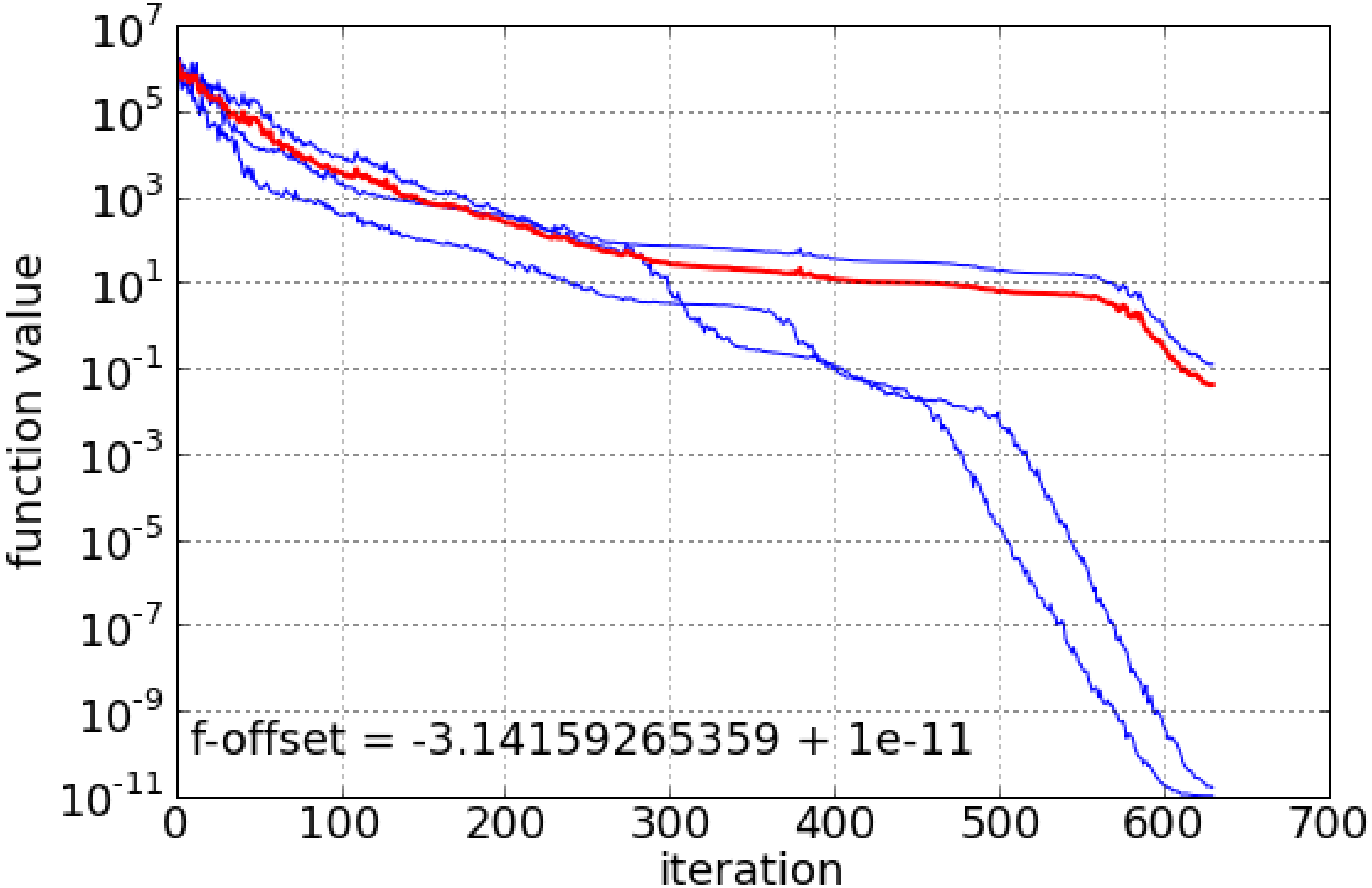
observation: three different "modes", which would be difficult to represent or recover in single statistics


blue:abs(f), cyan:f-min(f), green:sigma, red:axis ratio
Object Variables (mean, 19-D, popsize~12)
max std
min std
f_recent=1.2573081261636004e-14
x(10)=1.264
x(17)=1.164
x(2)=1.0349
x(1)=8.1094
x(0)=6.7434
x(11)=5.503
x(9)=4.3669
x(16)=4.285
x(15)=1.866
x(13)=-2.142
x(14)=-3.044
x(7)=-3.3336
x(12)=-4.874
x(5)=-5.3932
x(3)=-5.4402
x(18)=-5.749
x(8)=-6.3532
x(6)=-6.4708
x(4)=-8.7655
Scaling (All Main Axes)
Standard Deviations in All Coordinates
function evaluations
function evaluations

Which Statistic?



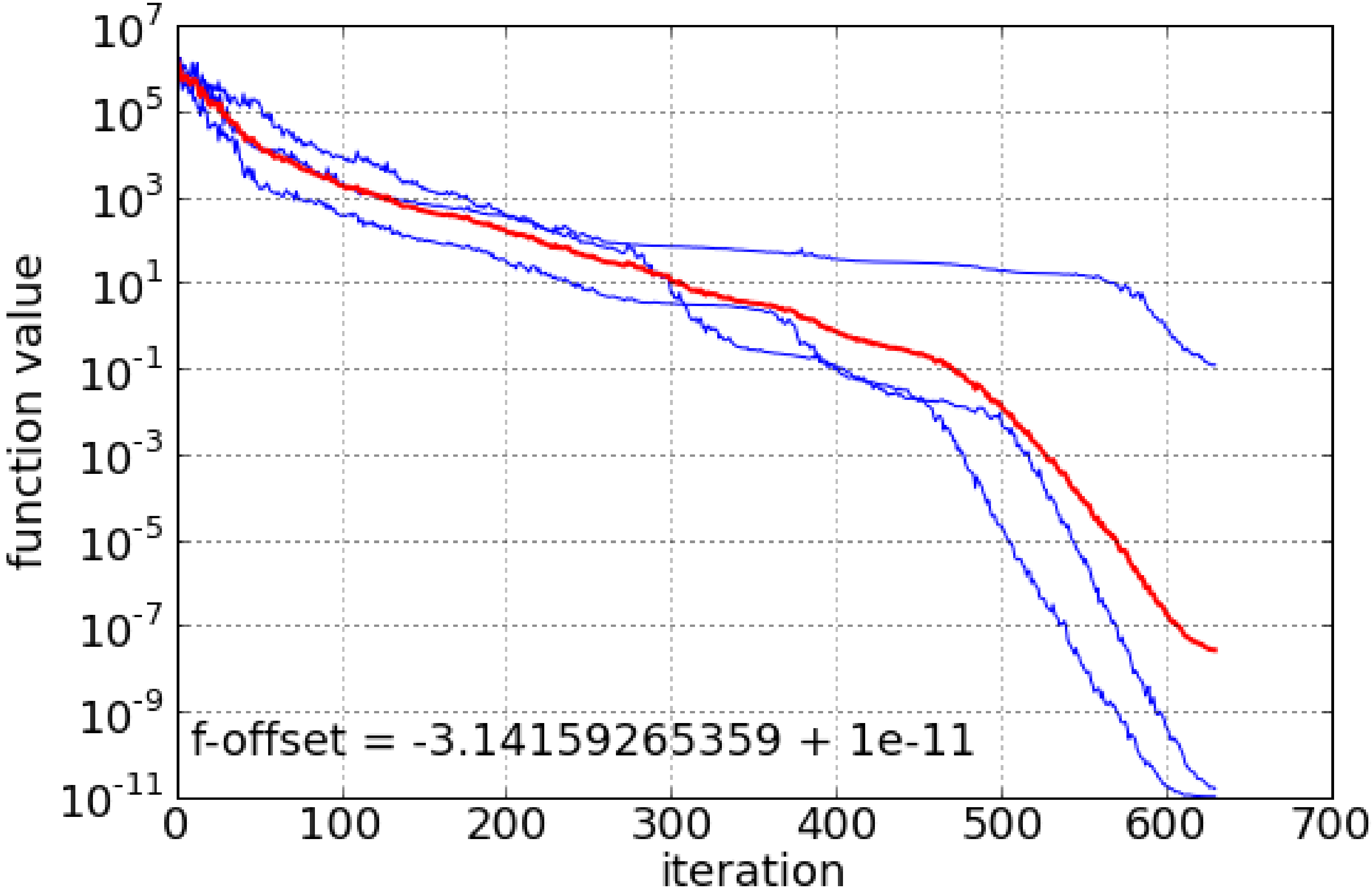
Which Statistic?



mean/average function value

- tends to emphasize large values

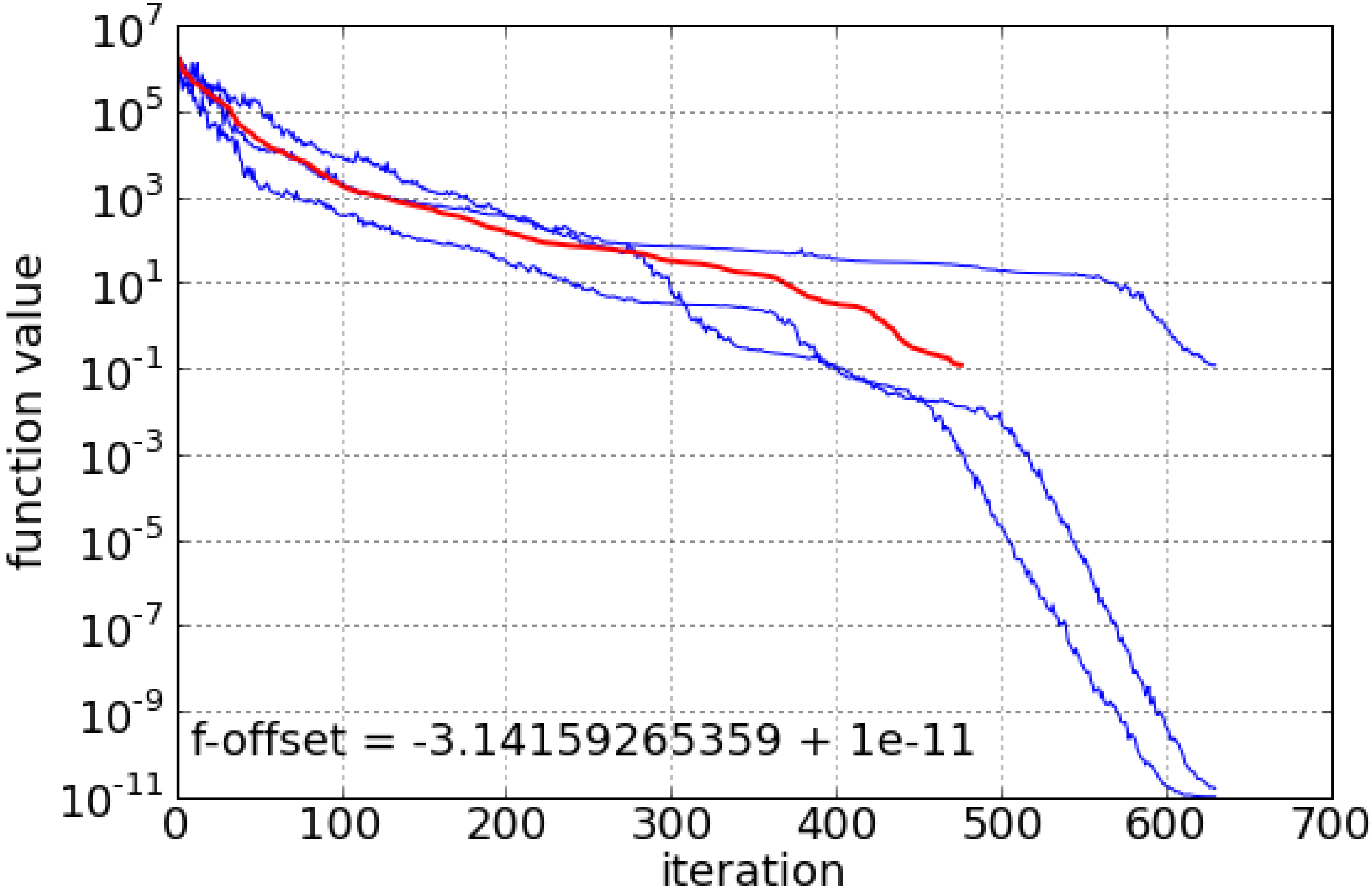
Which Statistic?



geometric average function value $\exp(\mathrm{mean_i}(\log(f_i))) = (\prod_{i=1}^{N} f_i)^{1/N}$

- reflects "visual" average
- depends on offset
- artefact due to adding 1e-11

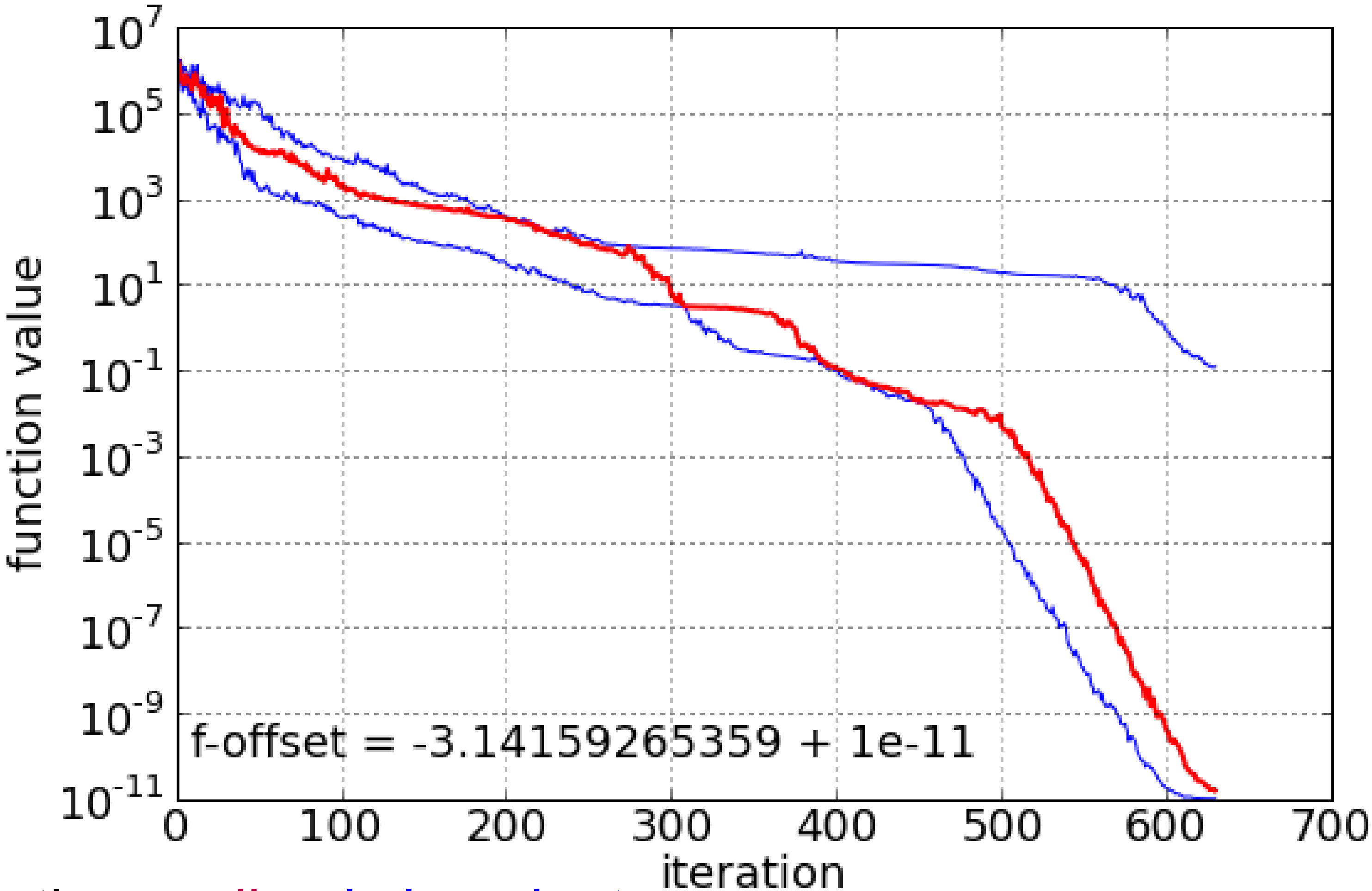
Which Statistic?



average iterations

- reflects "visual" average
- here: incomplete

Which Statistic?



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

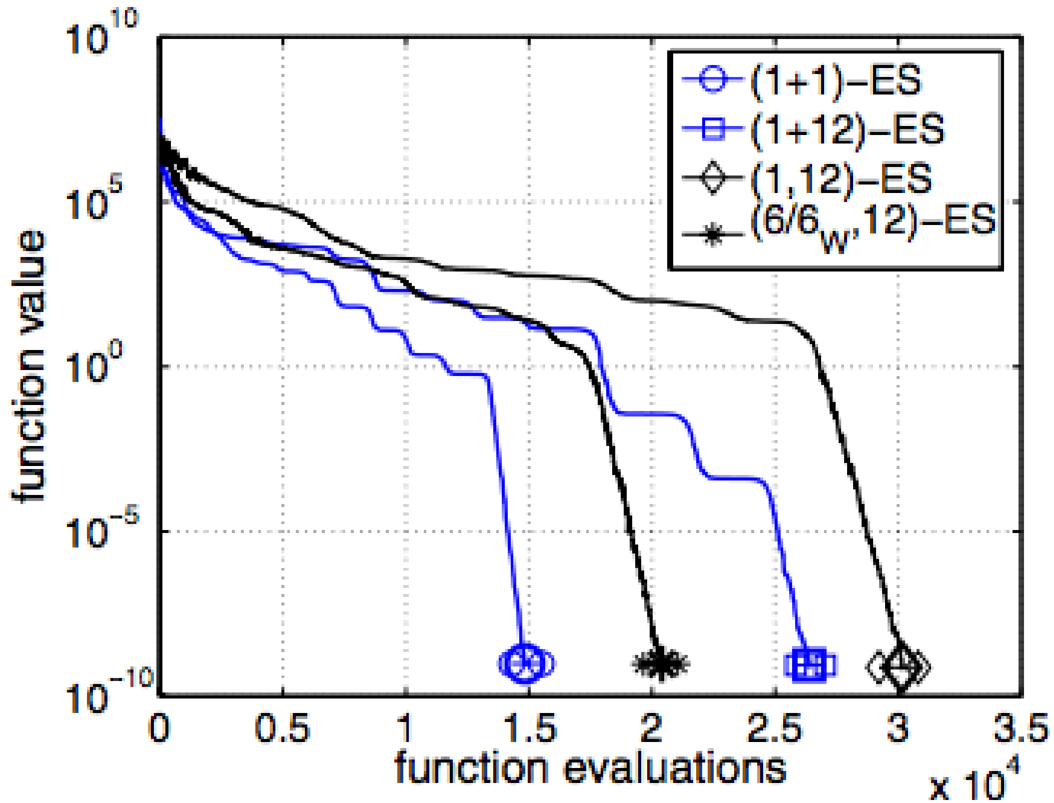
Implication

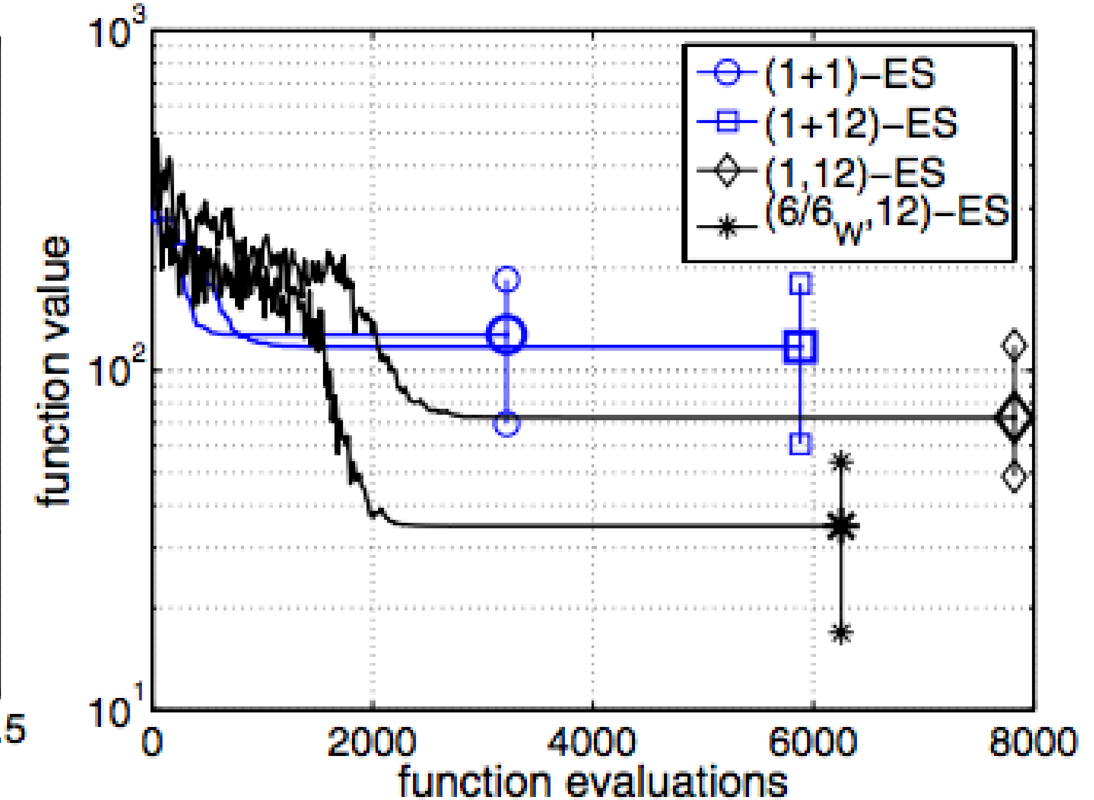
- use the median as summary datum
- more general: use quantiles as summary data

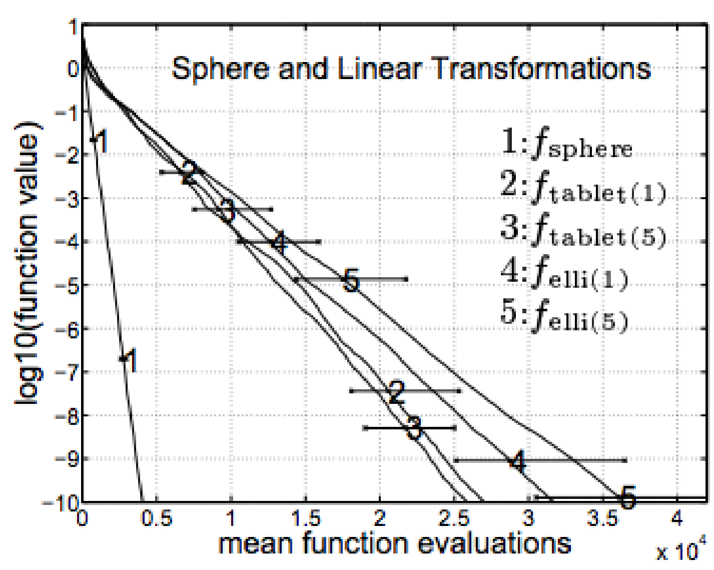
for example out of 15 data: 2nd, 8th, and 14th value represent the 10%, 50%, and 90%-tile

unless there are good reasons for a different statistic

Examples



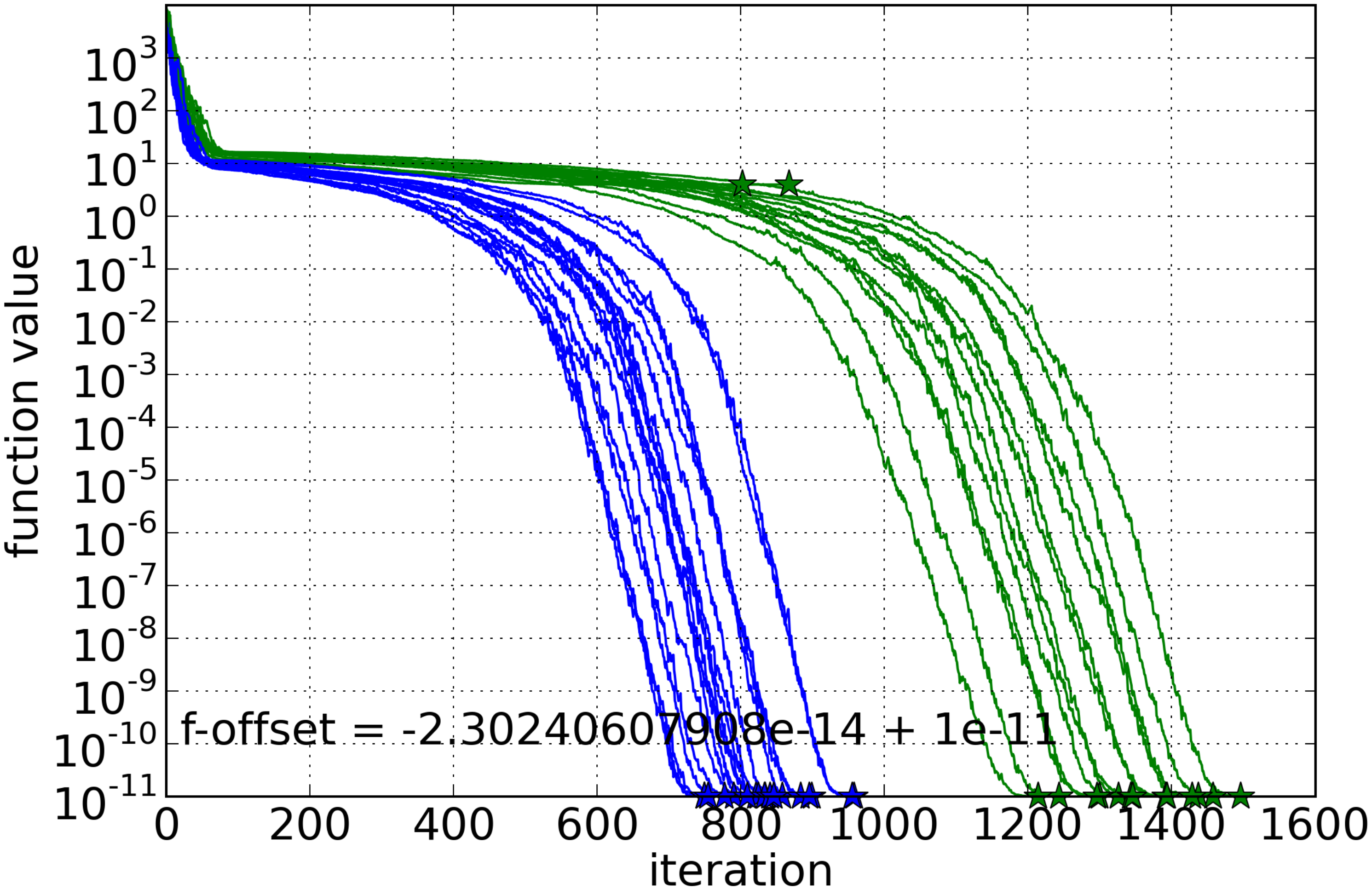




Comparison of 4 algorithms using the "median run" and the 90% central range of the final value on two different functions (Ellipsoid and Rastrigin)

caveat: this range display with simple error bars fails, if, e.g., 30% of all runs "converge"

Examples: Plotting All Data



Experiments from two algorithms, A1 and A2

Statistical Assessment

- Don't be scared!

1) Assess the meaning/relevance of a difference first (the only difficult part)

2) Apply rank-sum test (Wilcoxon, Mann-Whitney U)

   - only assumption: no equal data values

     as usual: useful even if assumptions do not hold,
     for categorical data: χ^2-test

   - hypothesis: $p(x > y) \neq p(x < y) \neq 1/2$
   - compares sum of ranks in a combined ranking
   - two-sided 1%-significance p-value needs only 2x5 data values

- For the same p-value, fewer significant data are better

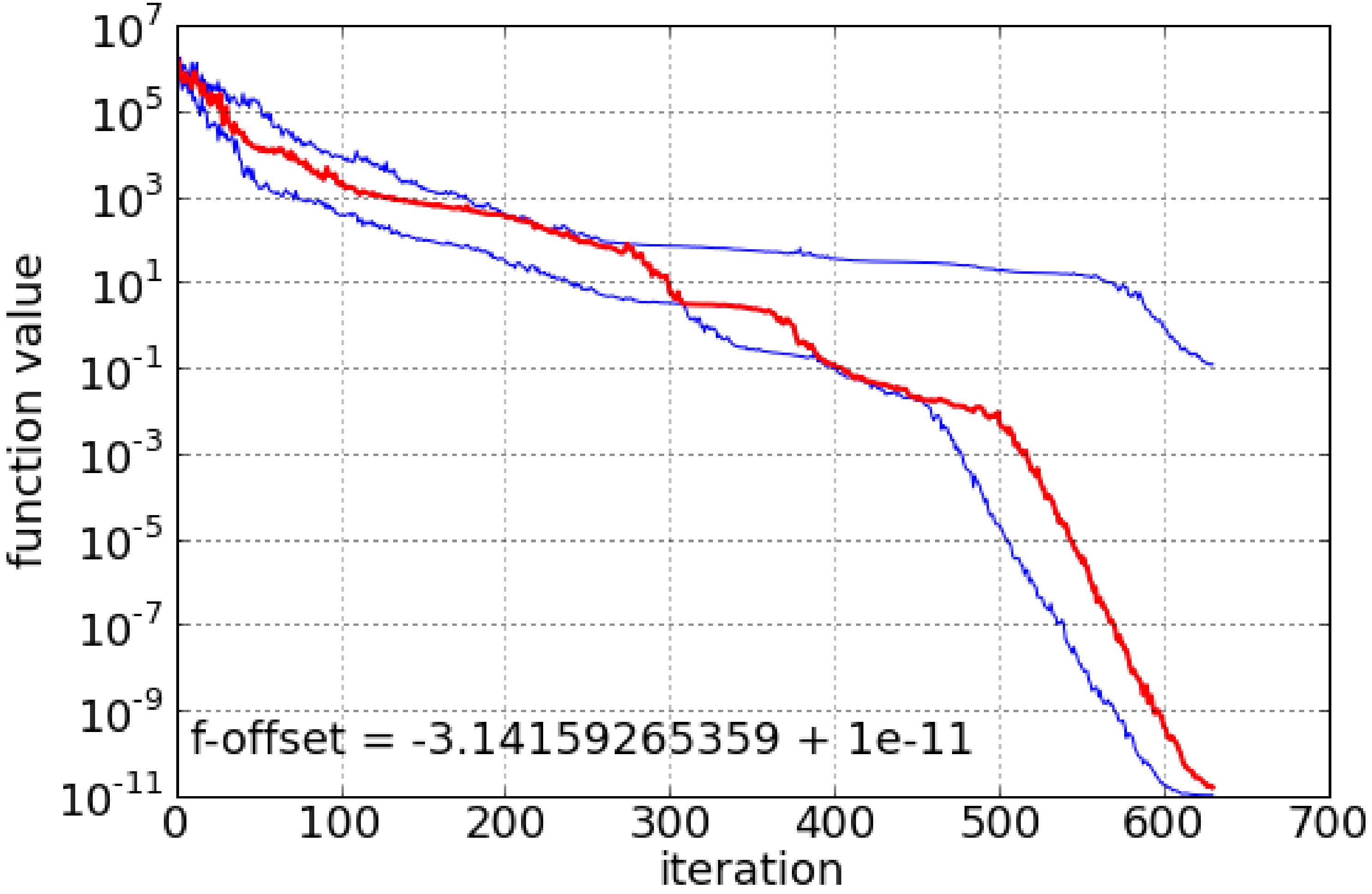
  using enough data, *any* difference
  can be made significant

Generally: non-parametric tests, Kolmogorov-Smirnov test for ECDFs, no need to use the t-test

Performance Measure(s)

Runtime

Three Convergence Graphs



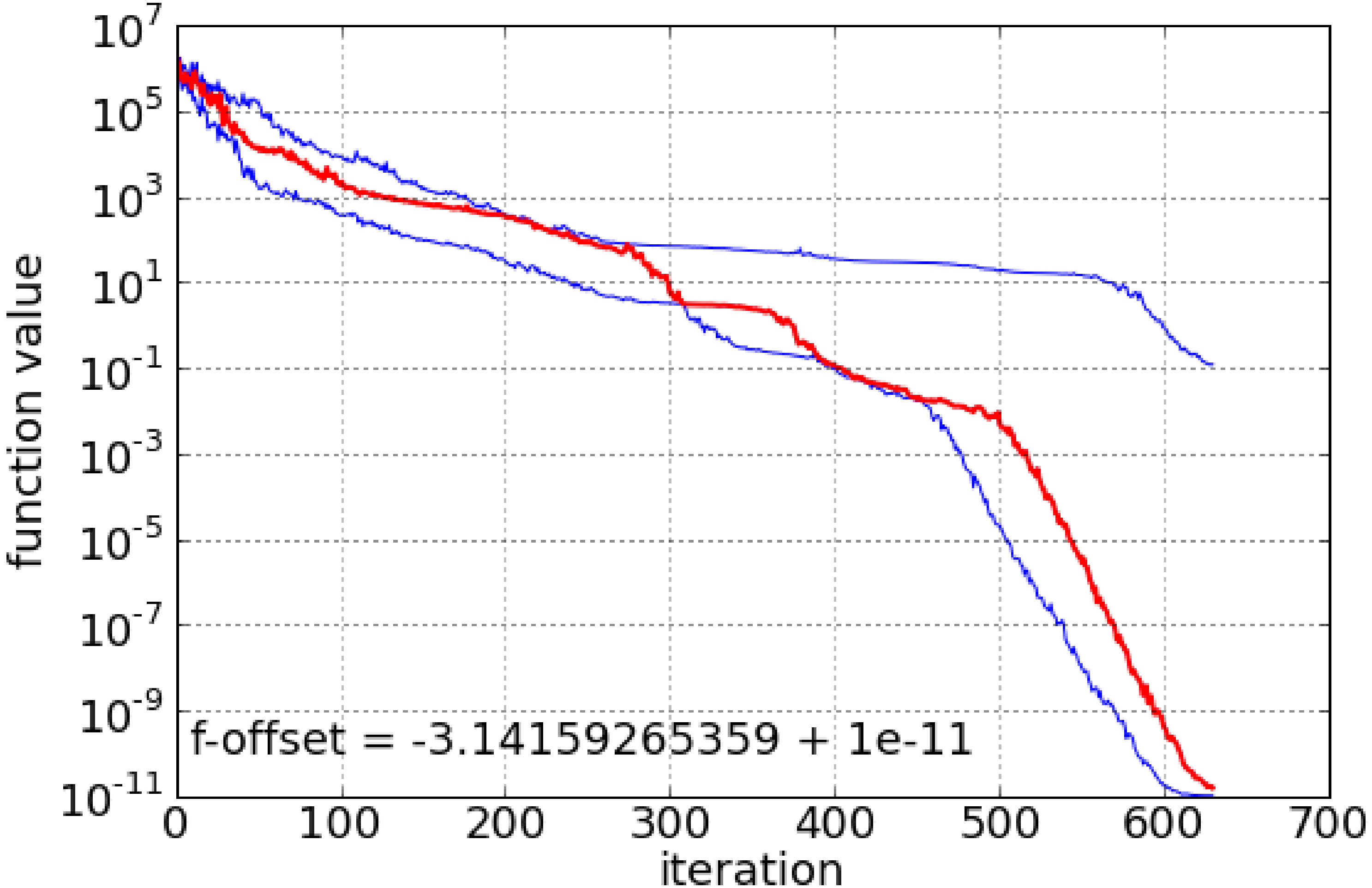
recall: convergence graphs is all we have

(recall) Black-Box Optimization

Two objectives:

- Find solution with small(est possible) function value
- With the least possible search costs (number of function evaluations)
- For measuring performance: fix one and measure the other

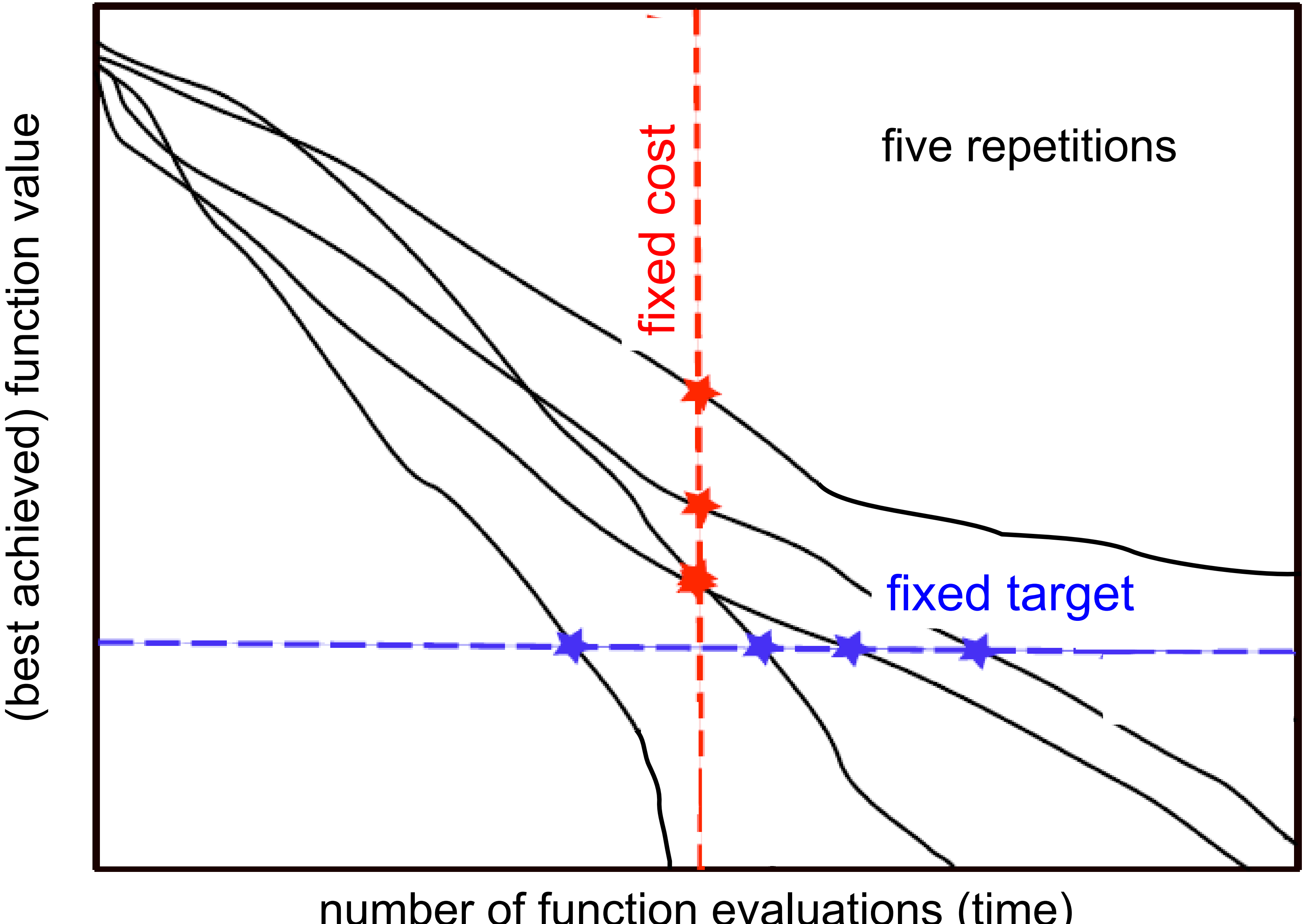
Two objectives



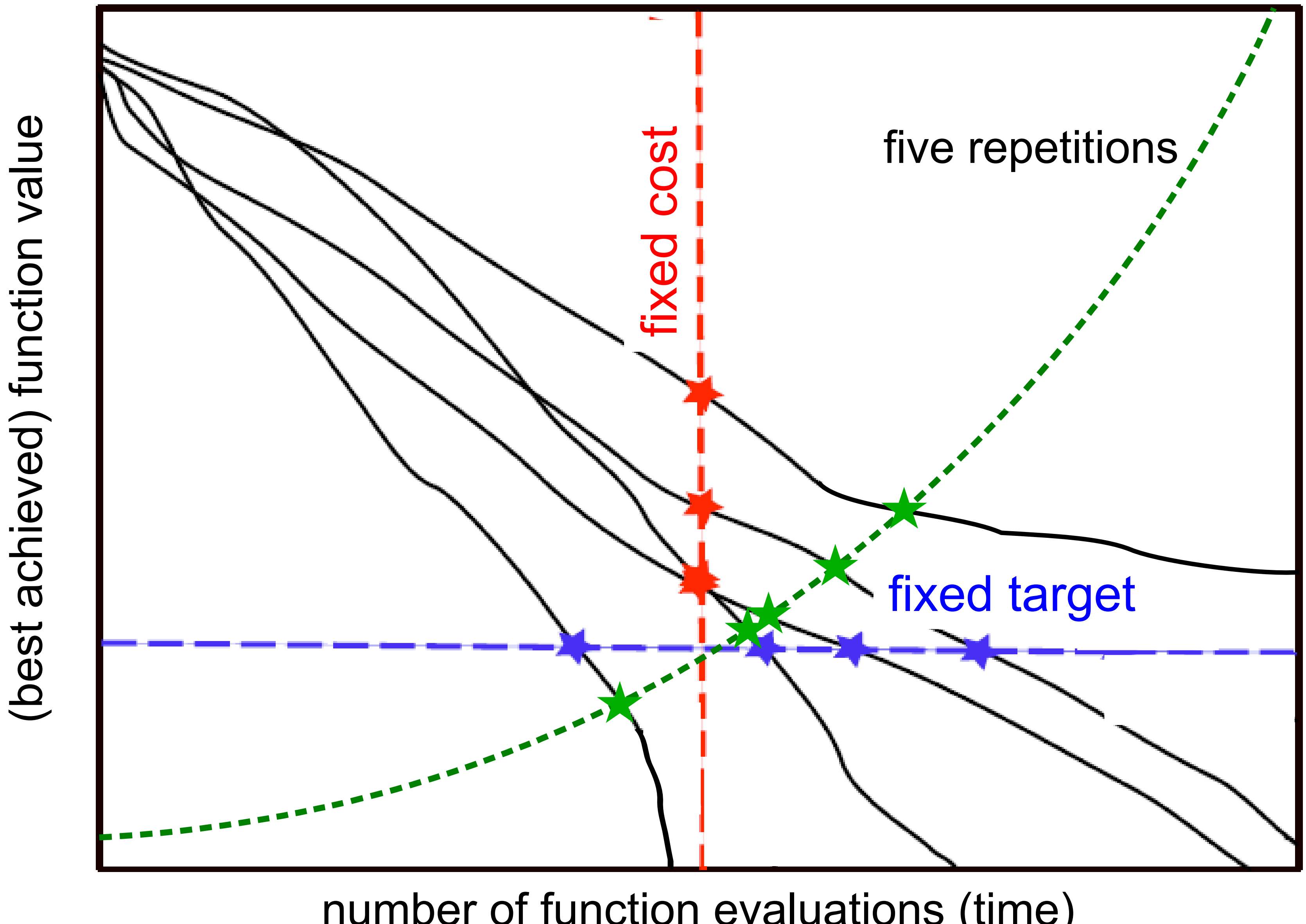
convergence graph is a plot in objective space

Measuring Performance from Convergence Graphs

fixed-cost versus fixed-target

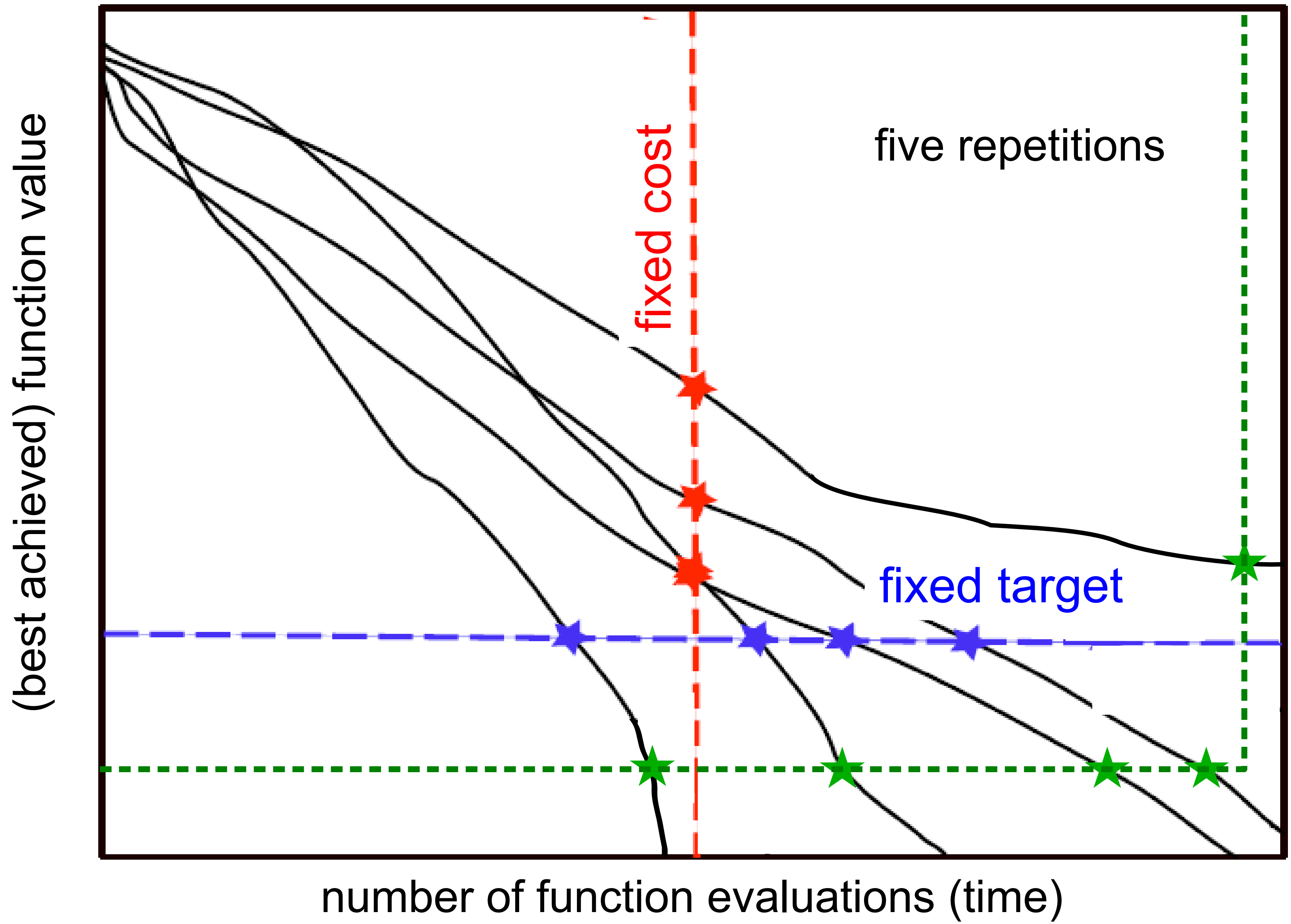


Measuring Performance from Convergence Graphs
fixed-cost versus fixed-target



Measuring Performance from Convergence Graphs

fixed-cost versus fixed-target



a performance should be

- quantitative on the ratio scale (highest possible)
  - “algorithm A is two *times* better than algorithm B” is a meaningful statement

- can 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)

The performance measure we use

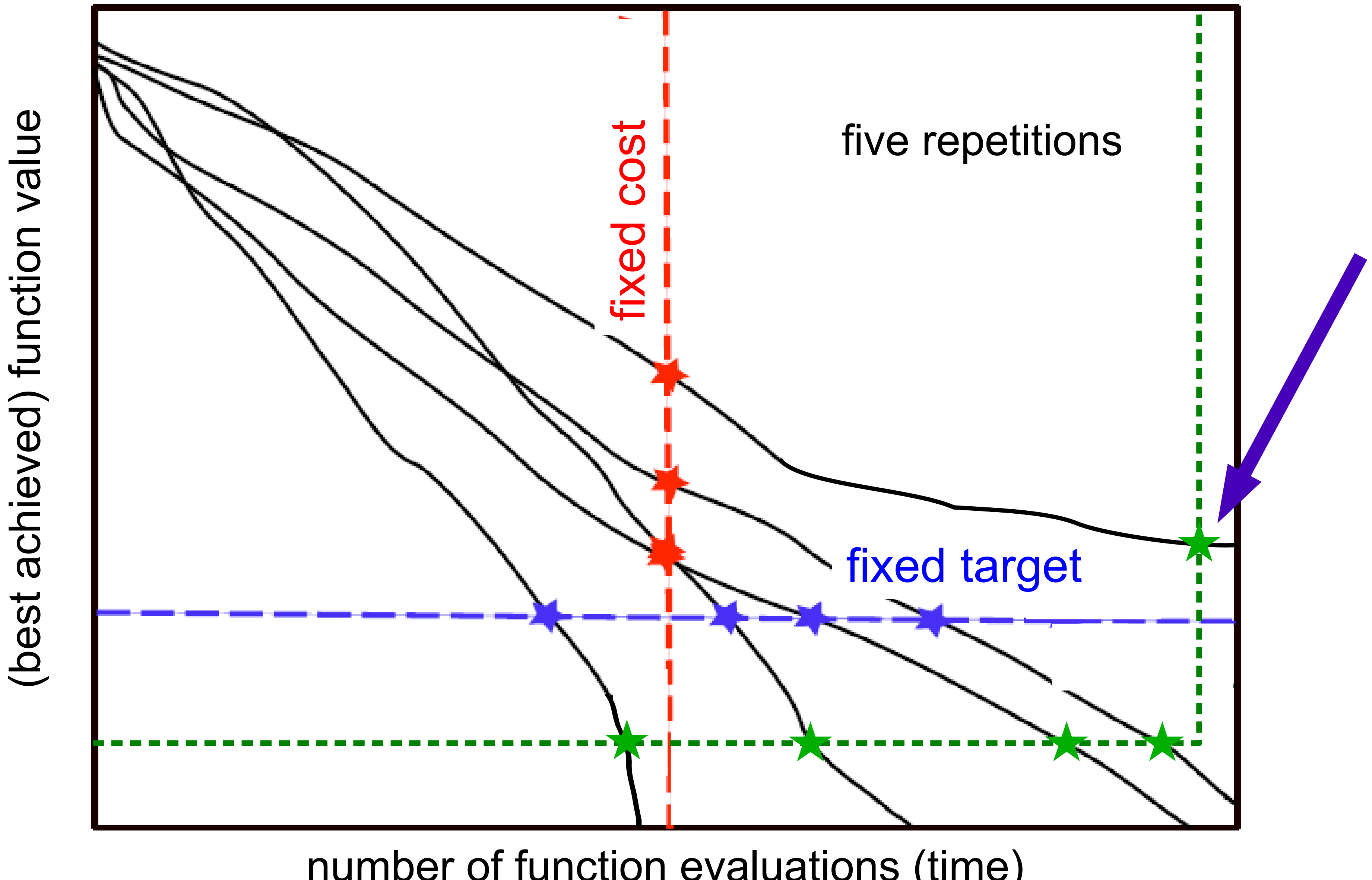
Run length or runtime or first hitting time to a given target function value measured in number of fitness function evaluations

equivalent to first hitting time of a sublevel set in search space

How can we deal with "missing values"?

Measuring Performance from Convergence Graphs

fixed-cost versus fixed-target



Fixed-target: Measuring Runtime

1. Fix a target f-value (most difficult part)

2. Compute the success rate $\widehat{p}$ as

$$\widehat{p} = \frac{\text{\# of successful runs (that reached the target)}}{\text{\# of all runs}} \in [0, 1]$$

$$\widehat{R} = \frac{1 - \widehat{p}}{\widehat{p}} = \frac{\text{\# of unsuccessful runs}}{\text{\# of successful runs}} \in [0, \infty]$$

$\widehat{R}$ is the odds ratio to be unsuccessful

$\widehat{R}$ is the number of unsuccessful runs observed *for each single successful* run (i.e. normalized by # of successful runs)

Fixed-target: Measuring Runtime

$$\widehat{R} = \frac{1-\widehat{p}}{\widehat{p}} = \frac{\text{\# of unsuccessful runs}}{\text{\# of successful runs}} \in [0, \infty]$$

$\widehat{R}$ is the odds ratio to be unsuccessful

$\widehat{R}$ is the number of unsuccessful runs observed *for each single successful* run (i.e. normalized by # of successful runs)

3. Compute "expected runtime" to hit the target

$$\mathrm{ERT} := \overbrace{\overline{\mathrm{RT}}_{\mathrm{succ}}}^{\text{average runtime for a single successful run}} + \underbrace{\widehat{R} \times \overline{\mathrm{RT}}_{\mathrm{unsucc}}}_{\text{average runtime spent in unsuccessful runs to achieve one successful run}}$$

$\mathrm{SP1} := \overline{\mathrm{RT}}_{\mathrm{succ}} + \widehat{R} \times \overline{\mathrm{RT}}_{\mathrm{succ}} = \overline{\mathrm{RT}}_{\mathrm{succ}}(1 + \widehat{R})$ disregarding runlength of unsuccessful runs

if $\widehat{R} < \infty$, else we can assume ERT $\geq \sum \mathrm{RT}_{\mathrm{unsucc}}$

Fixed-target: Measuring Runtime

$\widehat{R}$ is the number of unsuccessful runs observed *for each single successful* run (i.e. normalized by # of successful runs)

3. Compute "expected runtime" to hit the target

$$\text{ERT} := \overbrace{\overline{\text{RT}}_{\text{succ}}}^{\text{average runtime for a single successful run}} + \underbrace{\widehat{R} \times \overline{\text{RT}}_{\text{unsucc}}}_{\text{average runtime spent in unsuccessful runs to achieve one successful run}}$$

$\text{SP1} := \overline{\text{RT}}_{\text{succ}} + \widehat{R} \times \overline{\text{RT}}_{\text{succ}} = \overline{\text{RT}}_{\text{succ}}(1 + \widehat{R})$ disregarding runlength of unsuccessful runs

We can simulate a single runtime by "restarting" until the first success

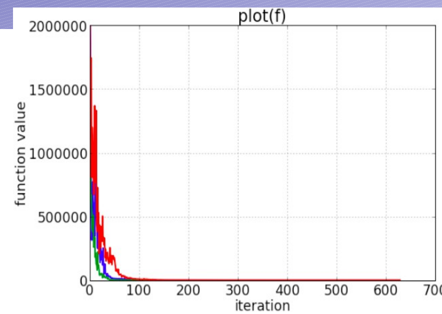
$$\text{RT} = \text{RT}_{\text{succ}} + \sum \text{RT}_{\text{unsucc}}$$

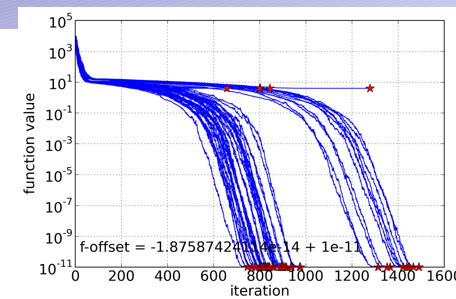
$\Longrightarrow$ distribution of runtimes incorporating unsuccessful runs
$\Longrightarrow$ display the distribution or a statistic of it

Break

Summary

- plot carefully
- display all data

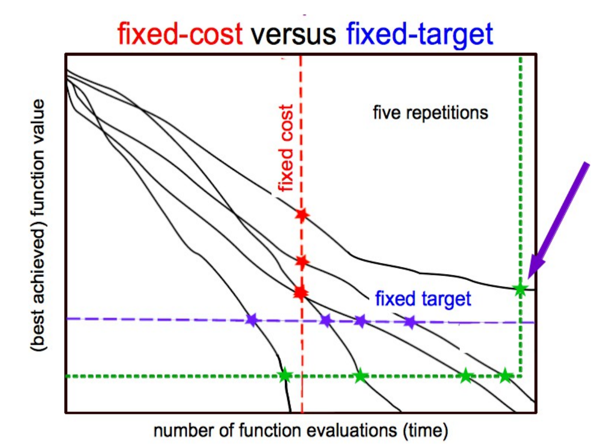




- use the median as summary datum

  unless for runtimes or you know exactly what you do

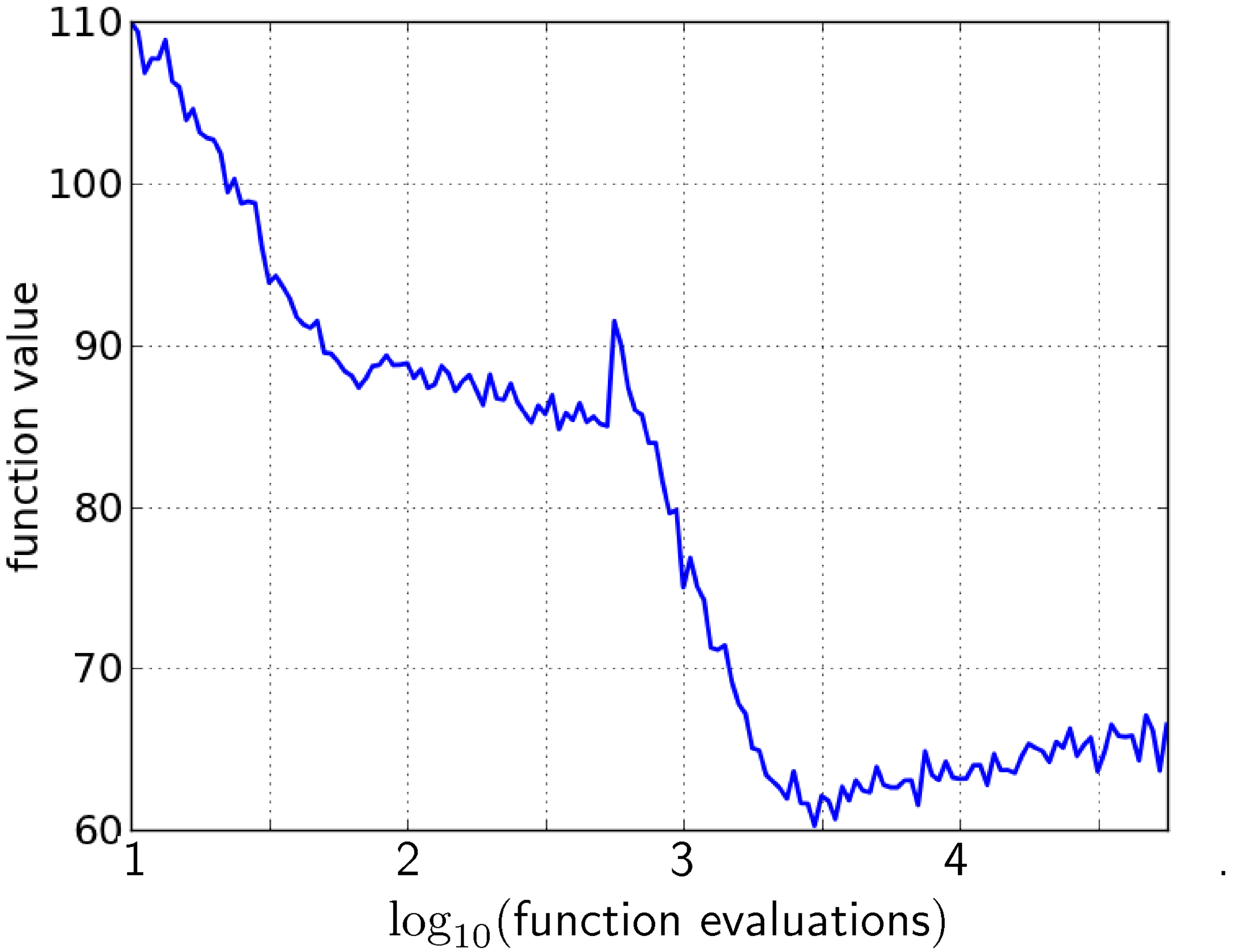
- more general: use quantiles as summary data
- assess a performance difference *before* to worry about statistical significance
- vertical vs. horizontal view-point
- run"time" RT and
  - ERT (expected RT)
  - runtime ECDF (empirical cumulative distribution fct)



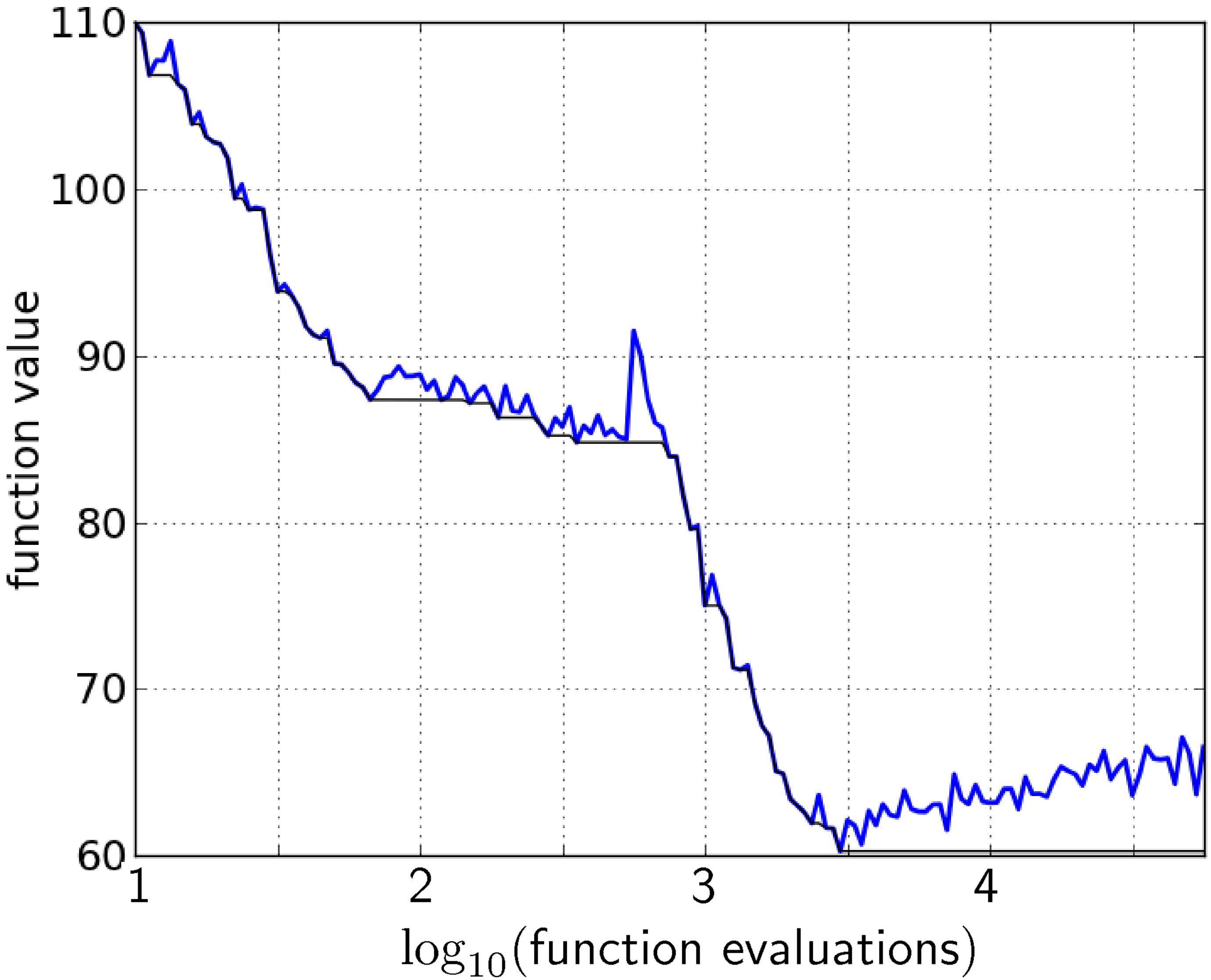
ECDF:

Empirical Cumulative Distribution Function of the Runtime

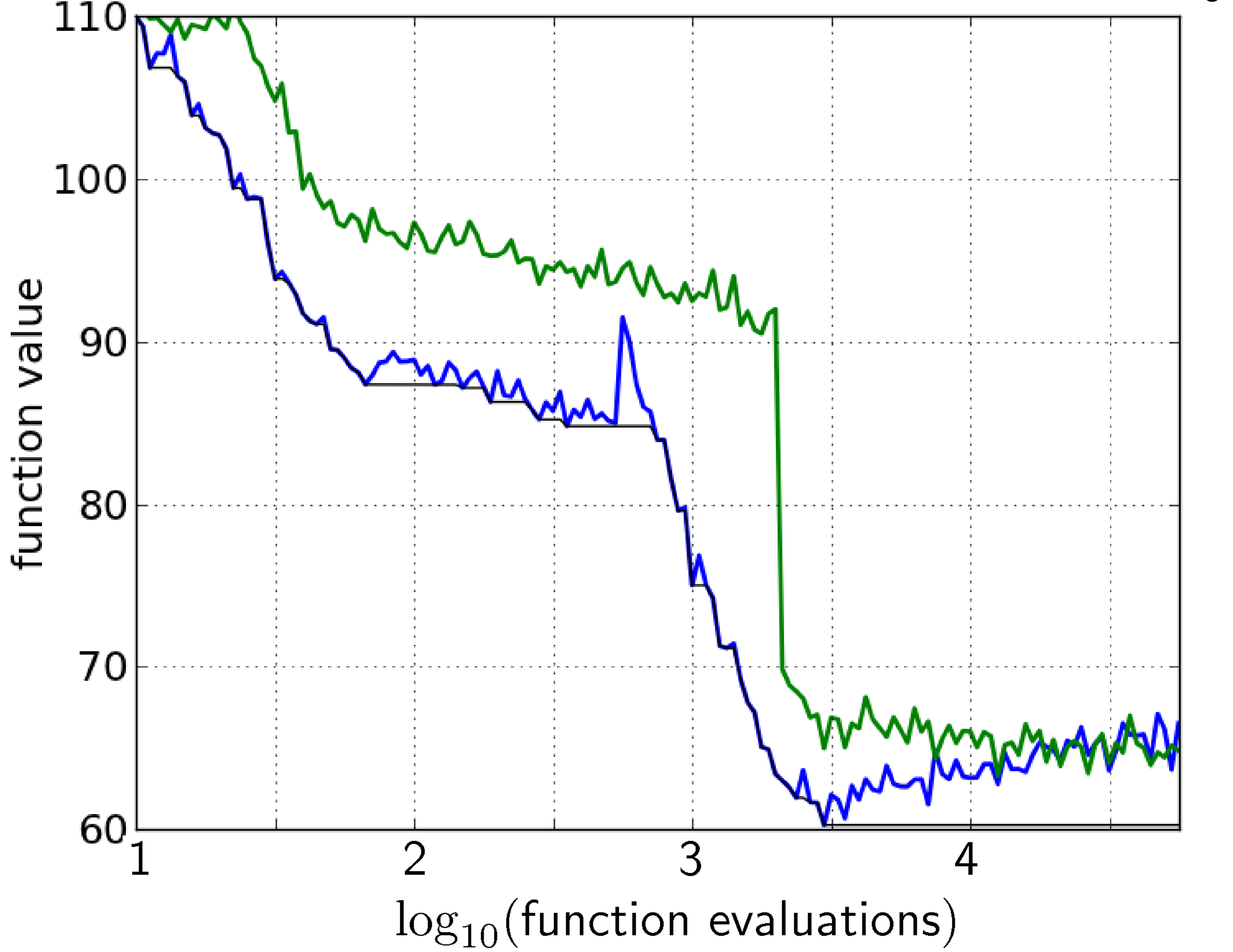
A Convergence Graph



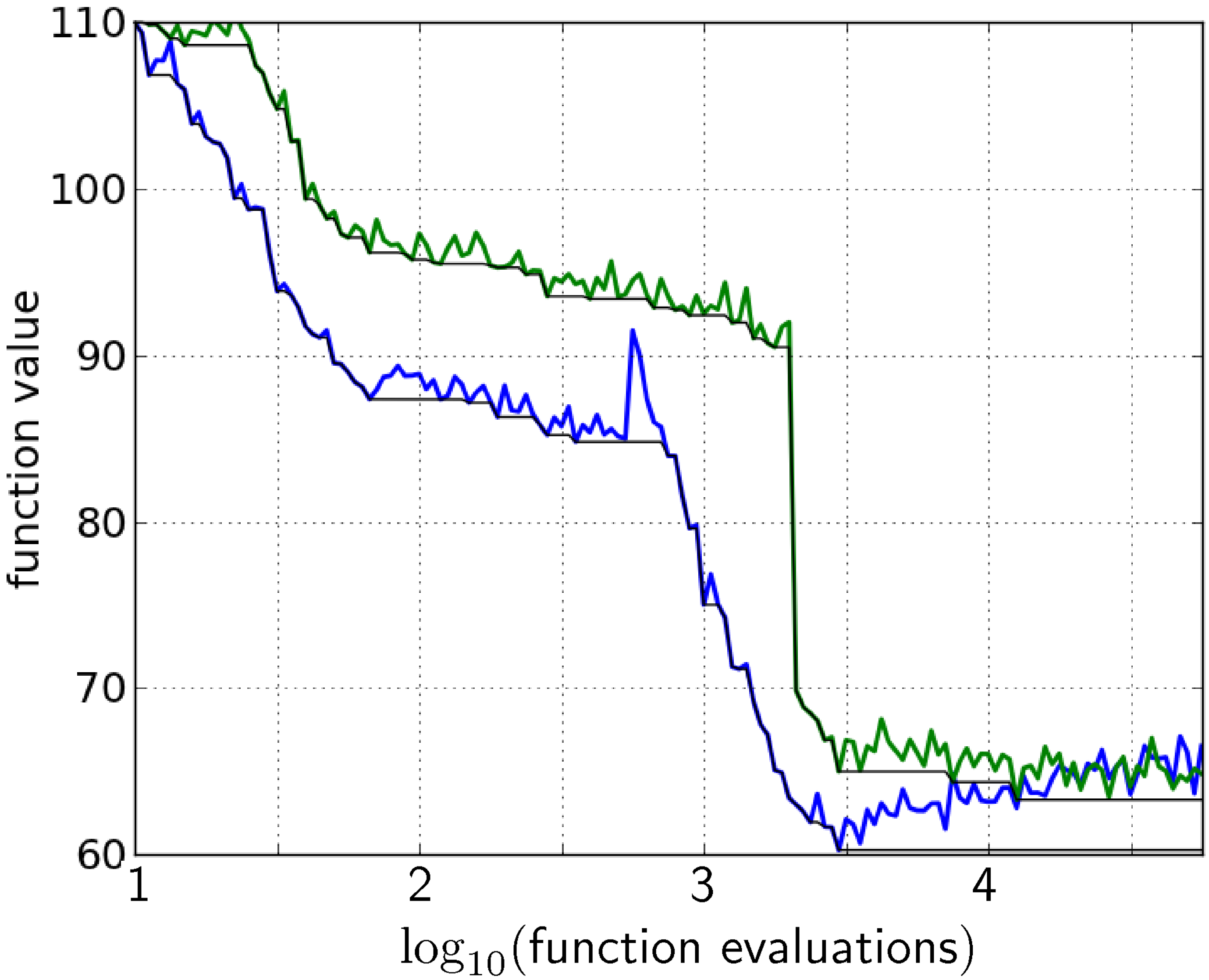
First hitting time is monotonous



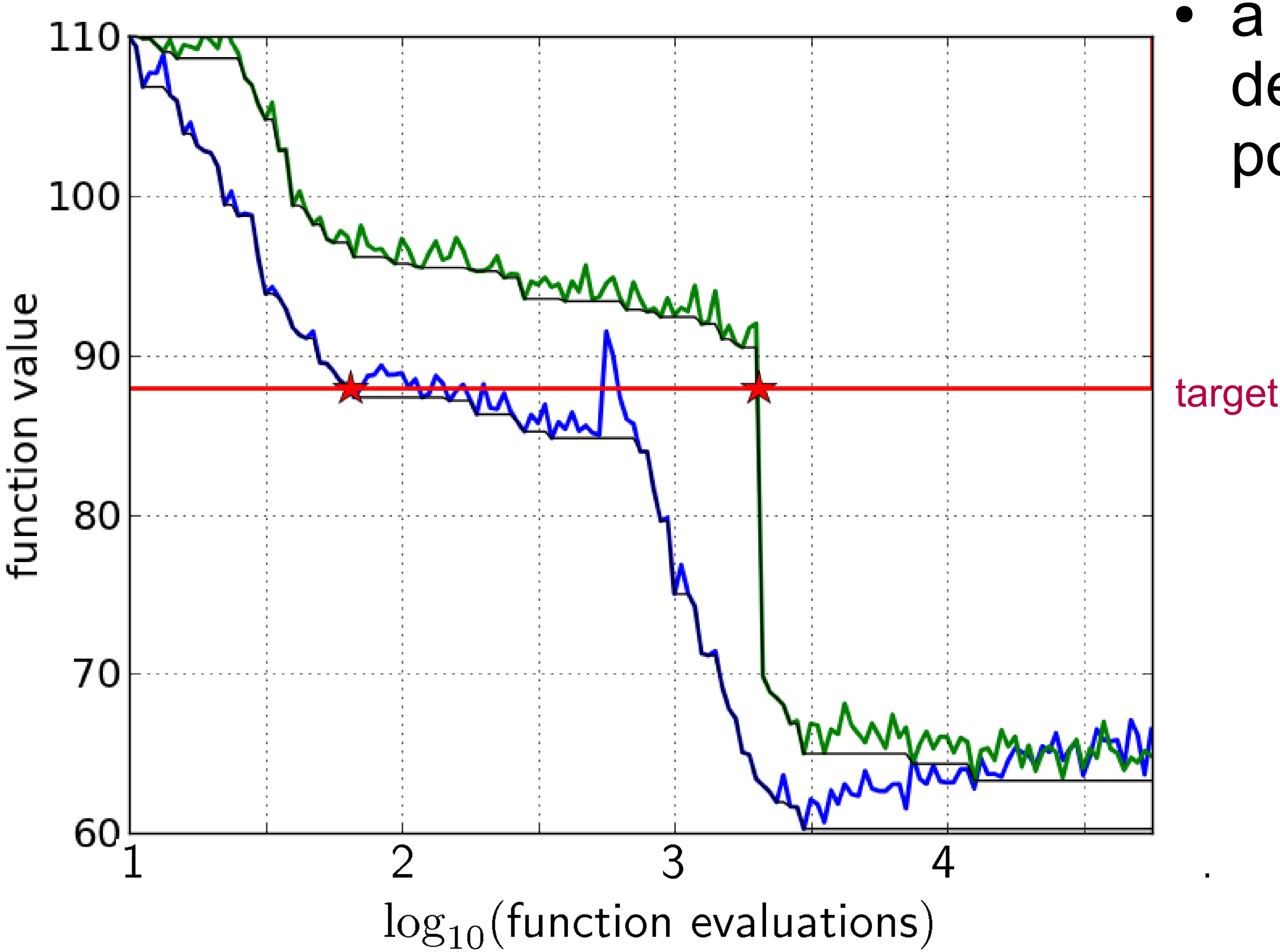
- first hitting time: a monotonous graph



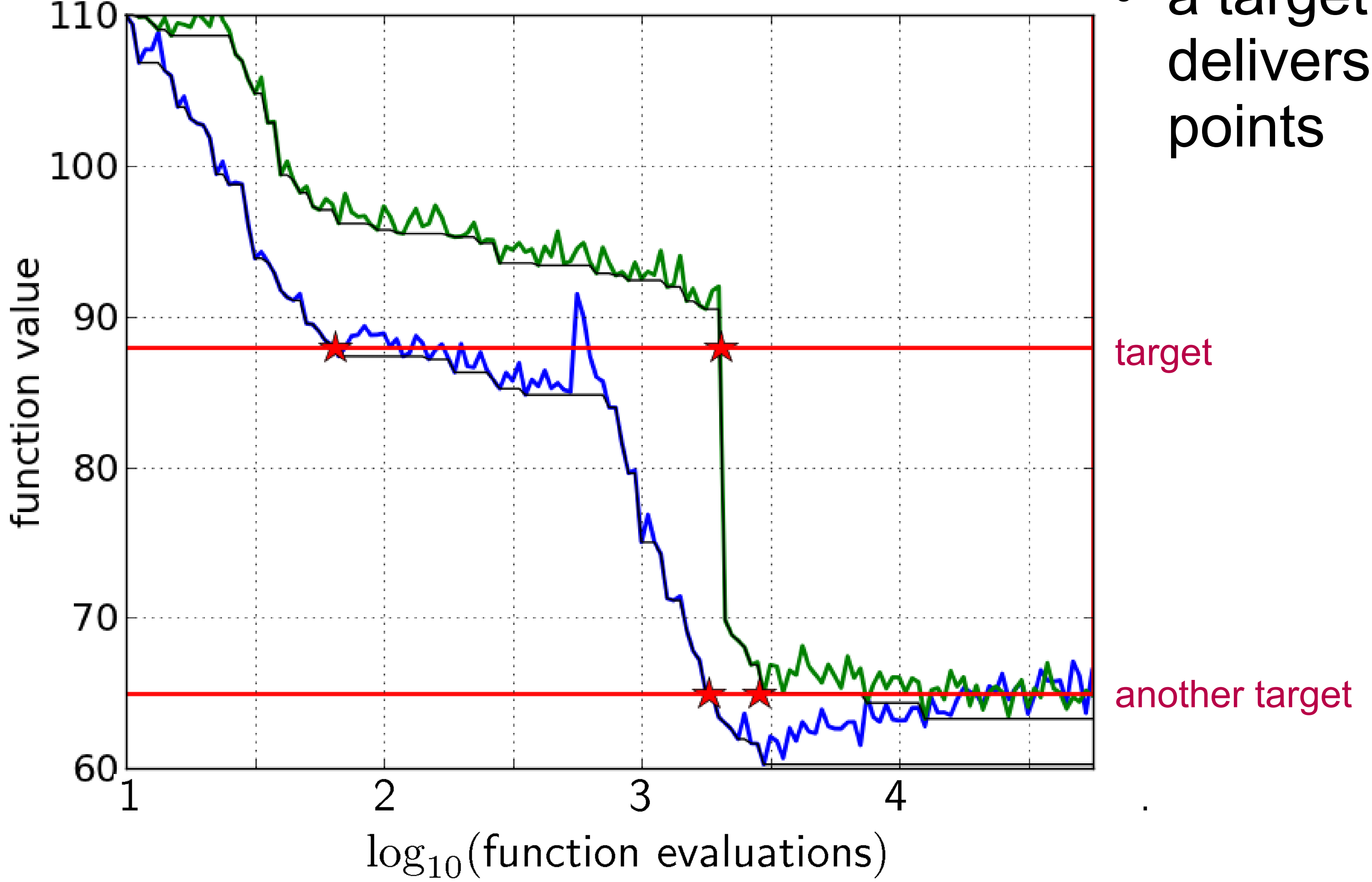
- another convergence graph



- another convergence graph with hitting time

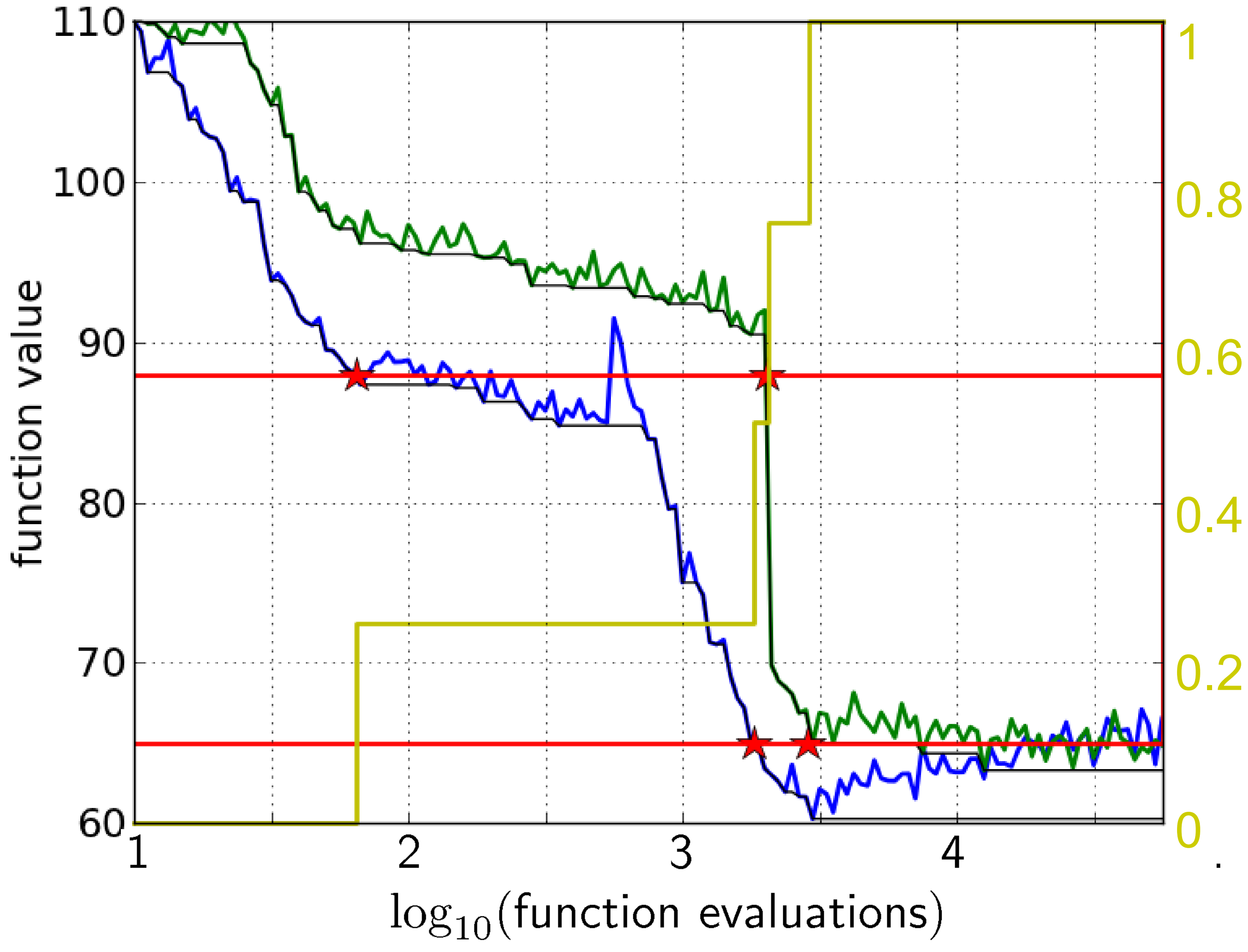


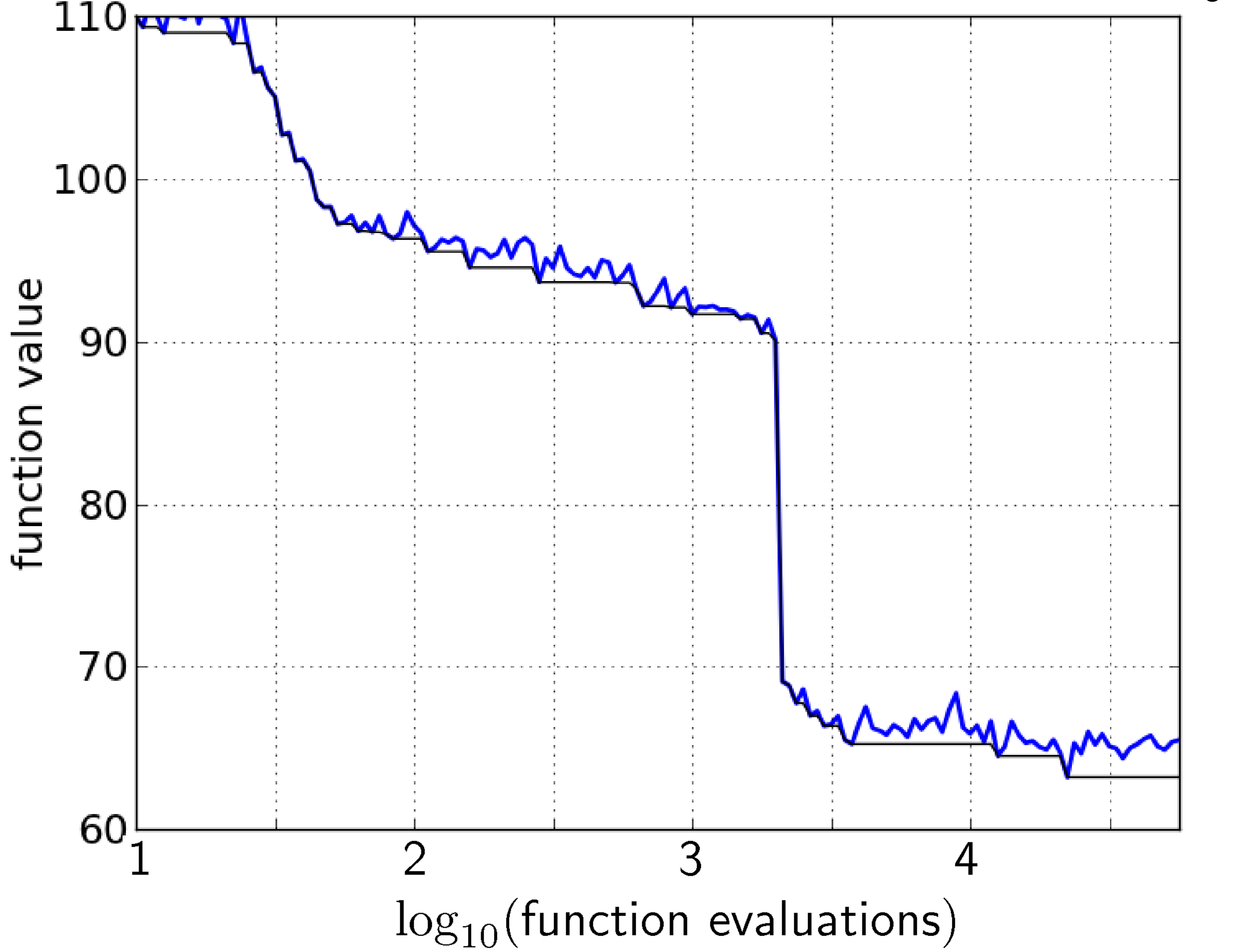
- a target value delivers two data points



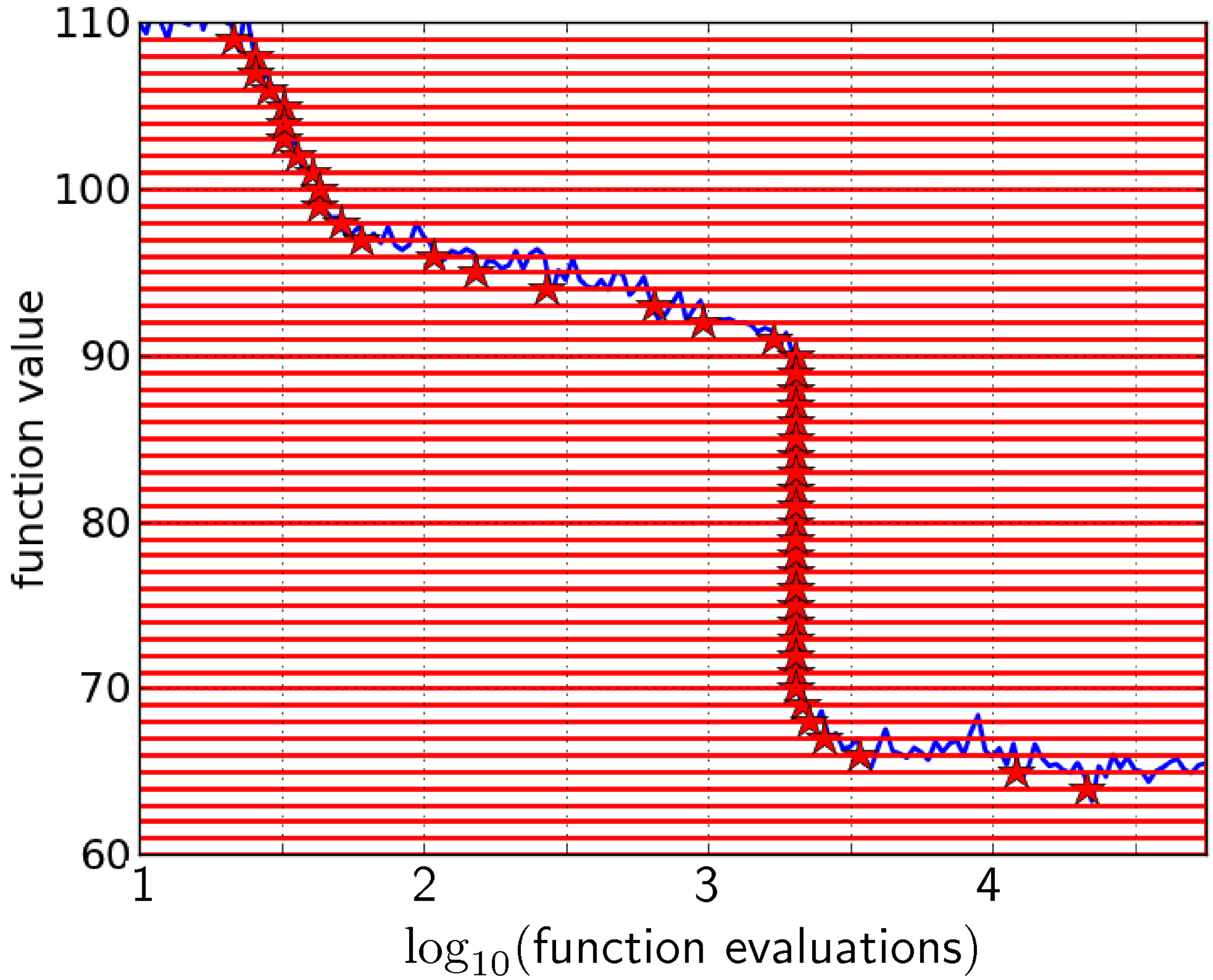
- a target value delivers two data points

ECDF with four data points

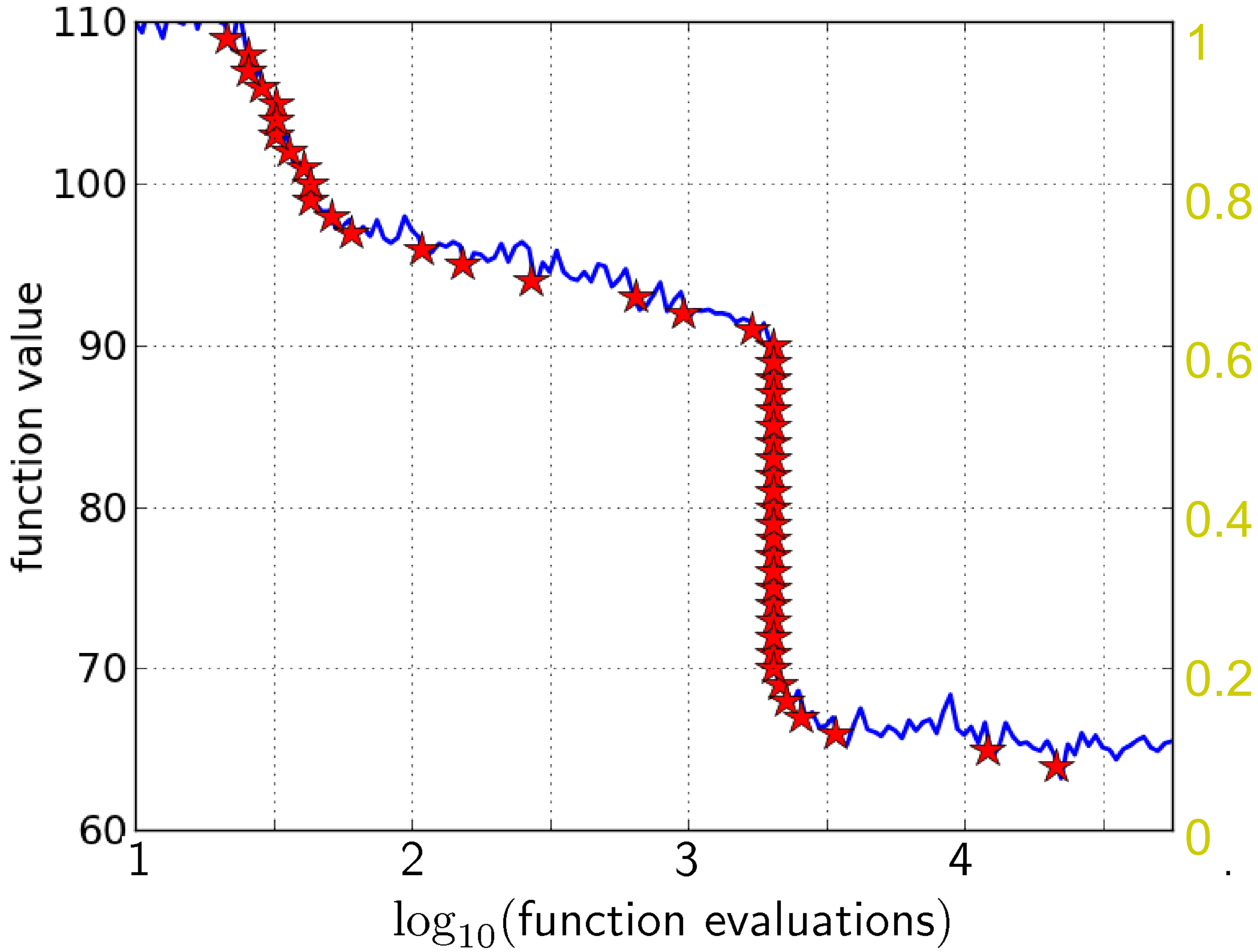


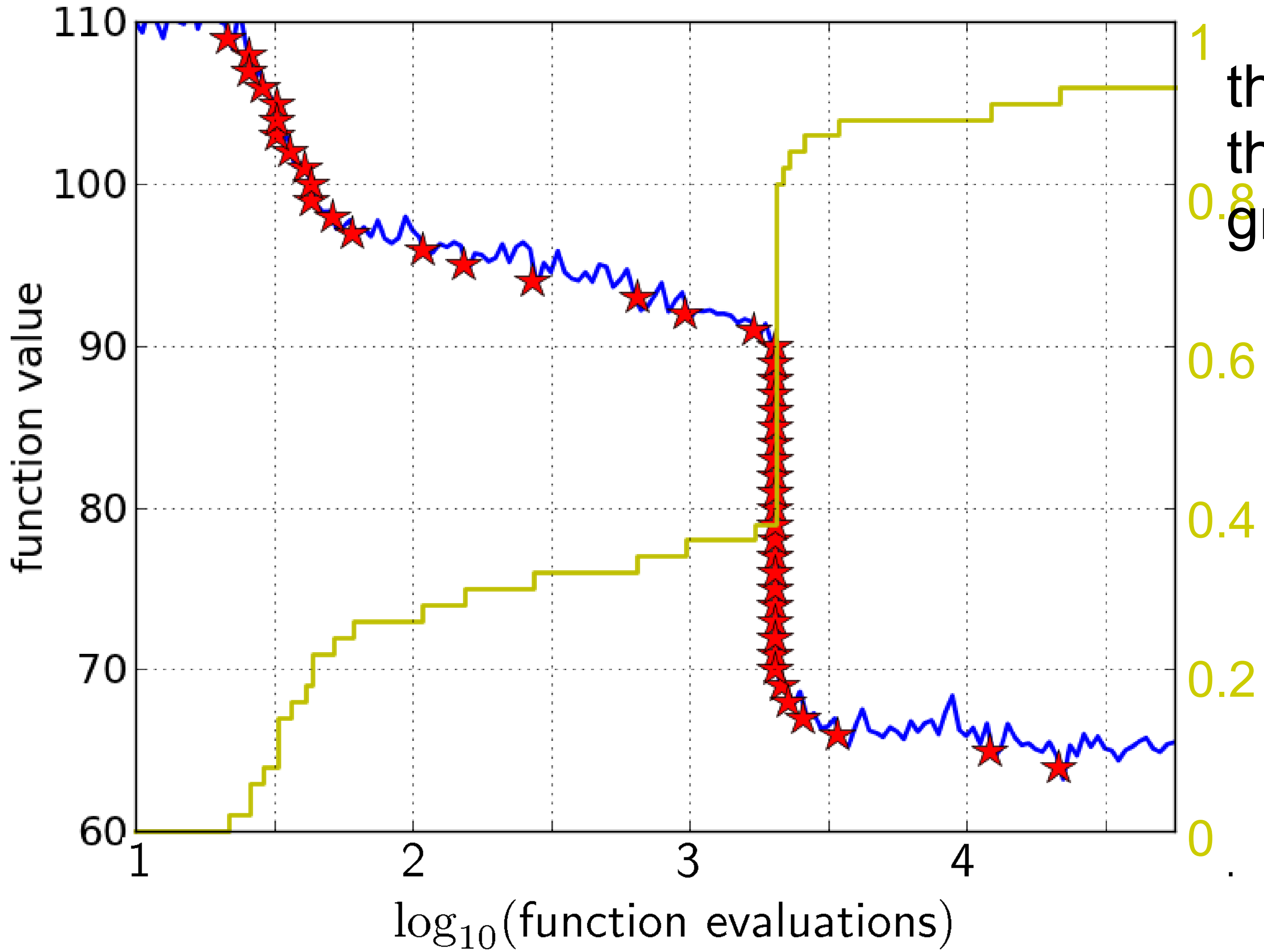


- reconstructing a single run

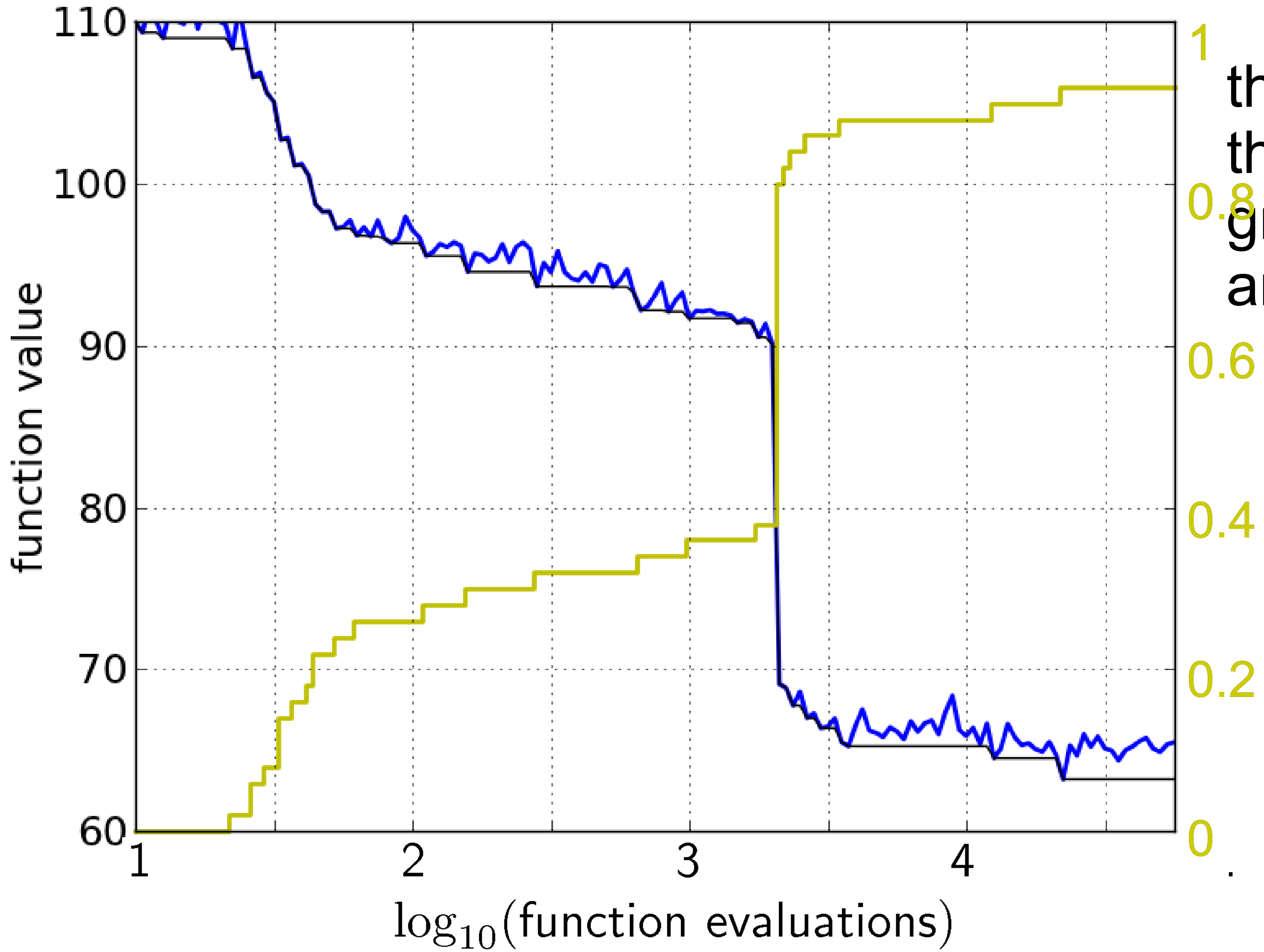


50 equally spaced targets

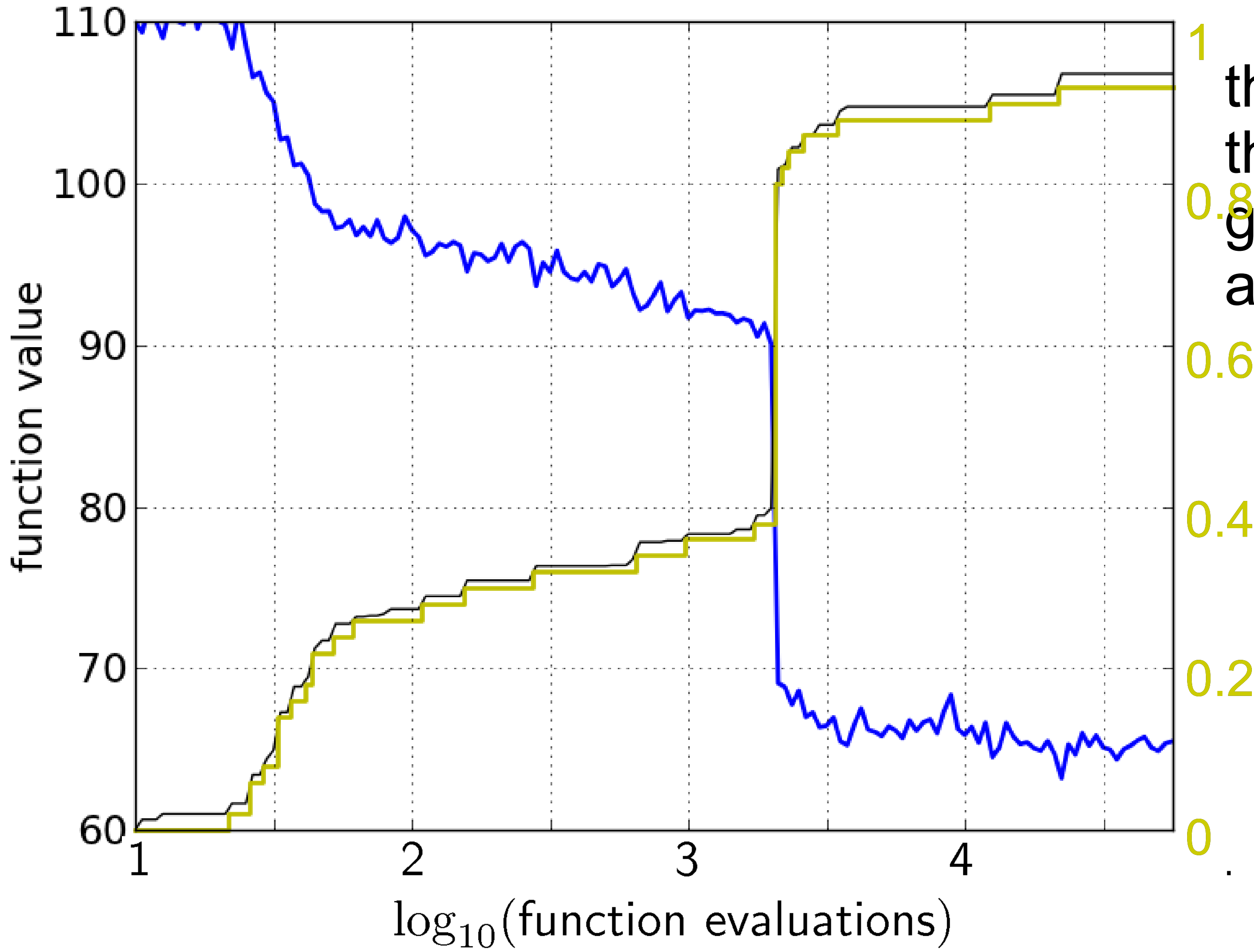

110
100
90
80
70
60
function value
1
0.8
0.6
0.4
0.2
0
1
2
3
4
log10(function evaluations)



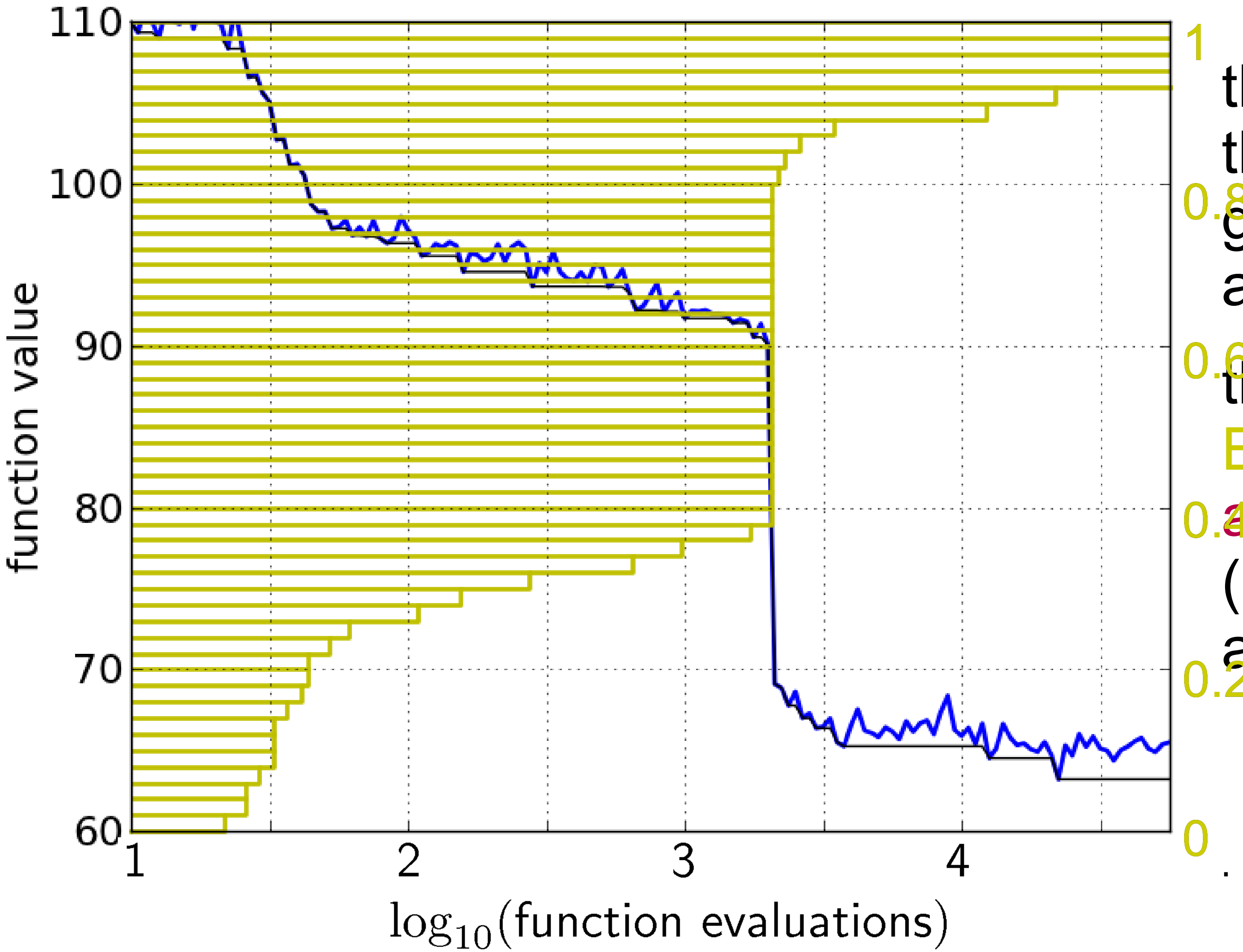
the ECDF recovers the monotonous graph



the ECDF recovers the monotonous graph, discretised and flipped.

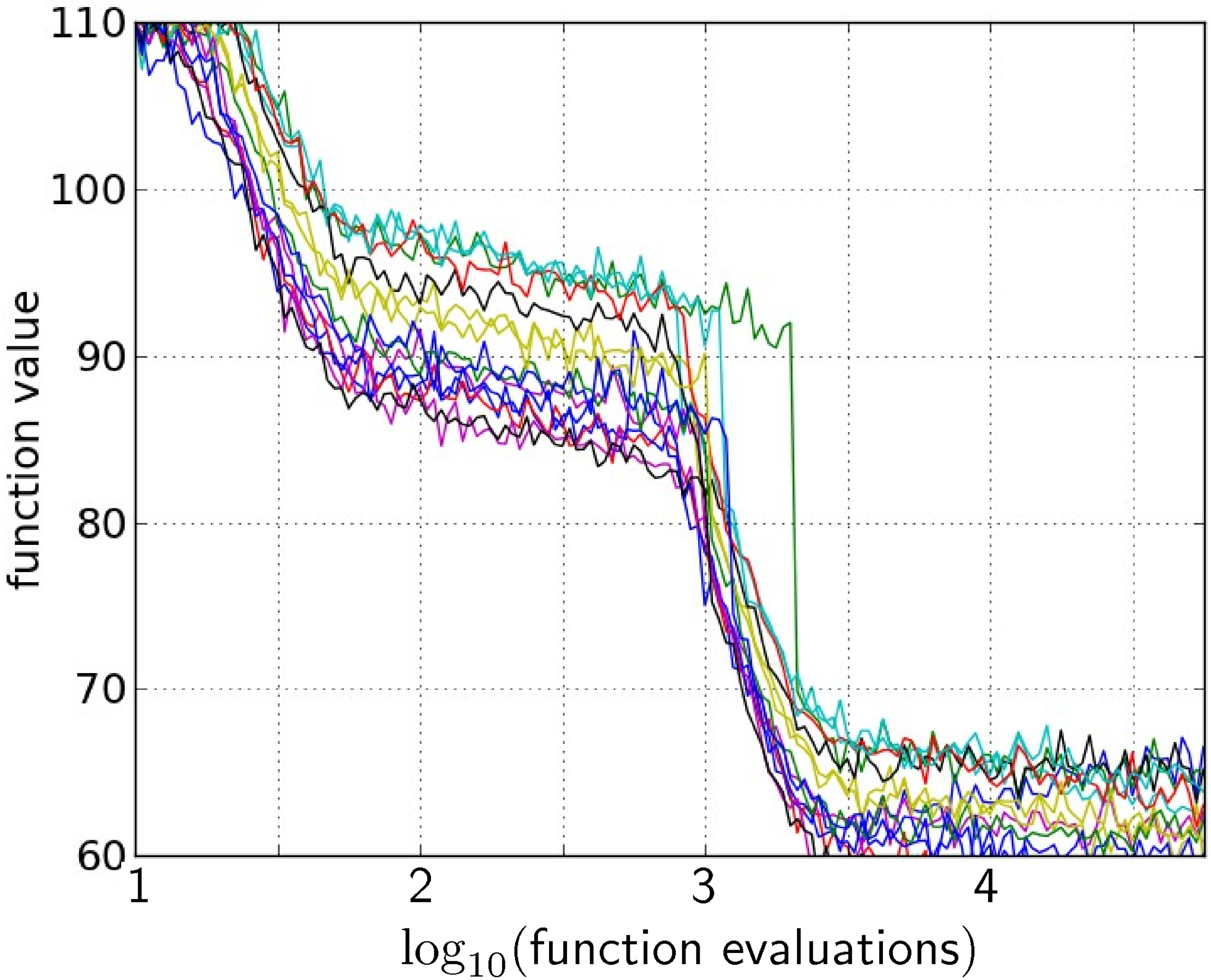


the ECDF recovers the monotonous graph, discretised and flipped

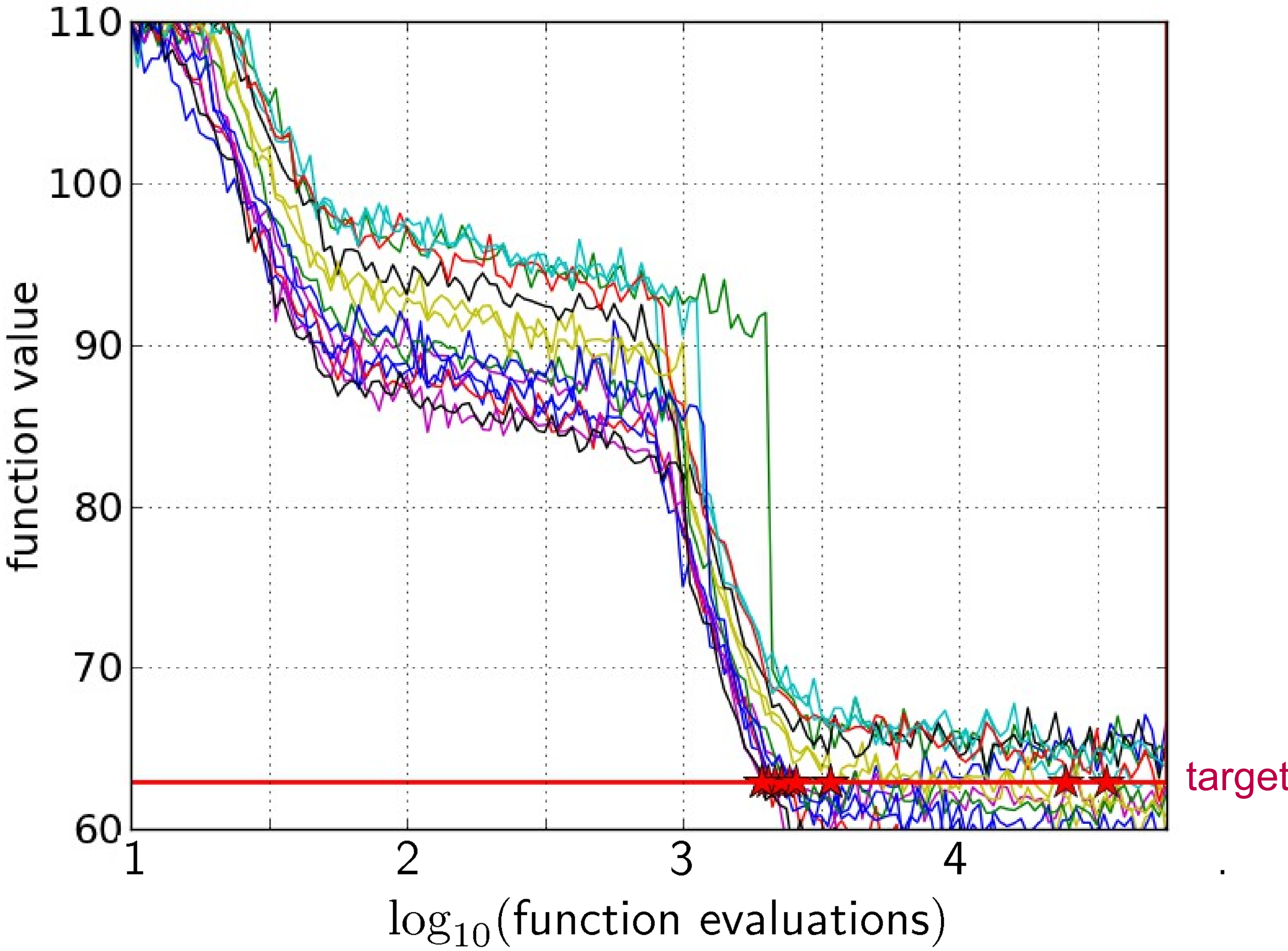


the ECDF recovers the monotonous graph, discretised and flipped

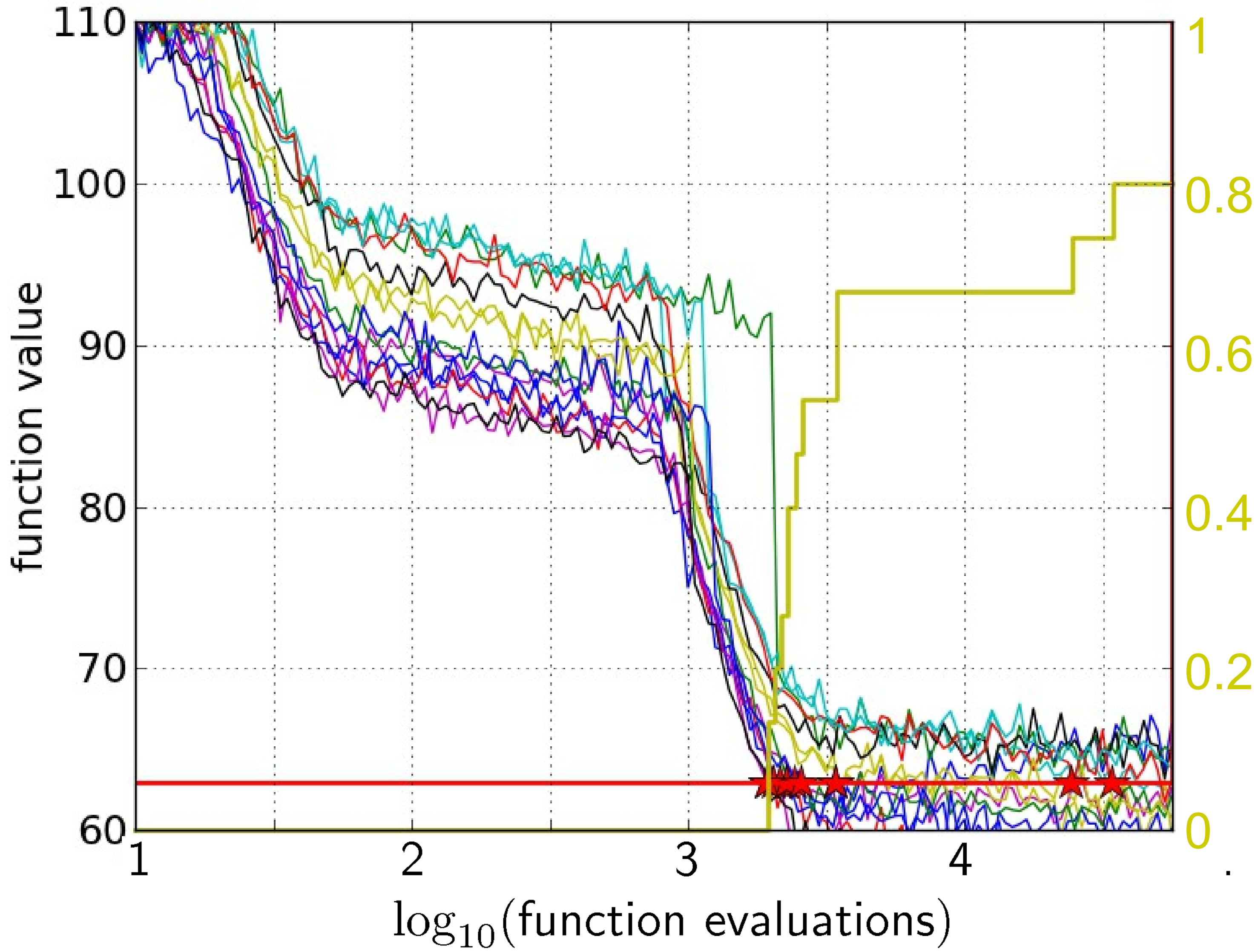
the area over the ECDF curve is the average log runtime (or geometric average runtime).


110
100
90
80
70
60
function value
1
2
3
4
log10(function evaluations)

15 runs

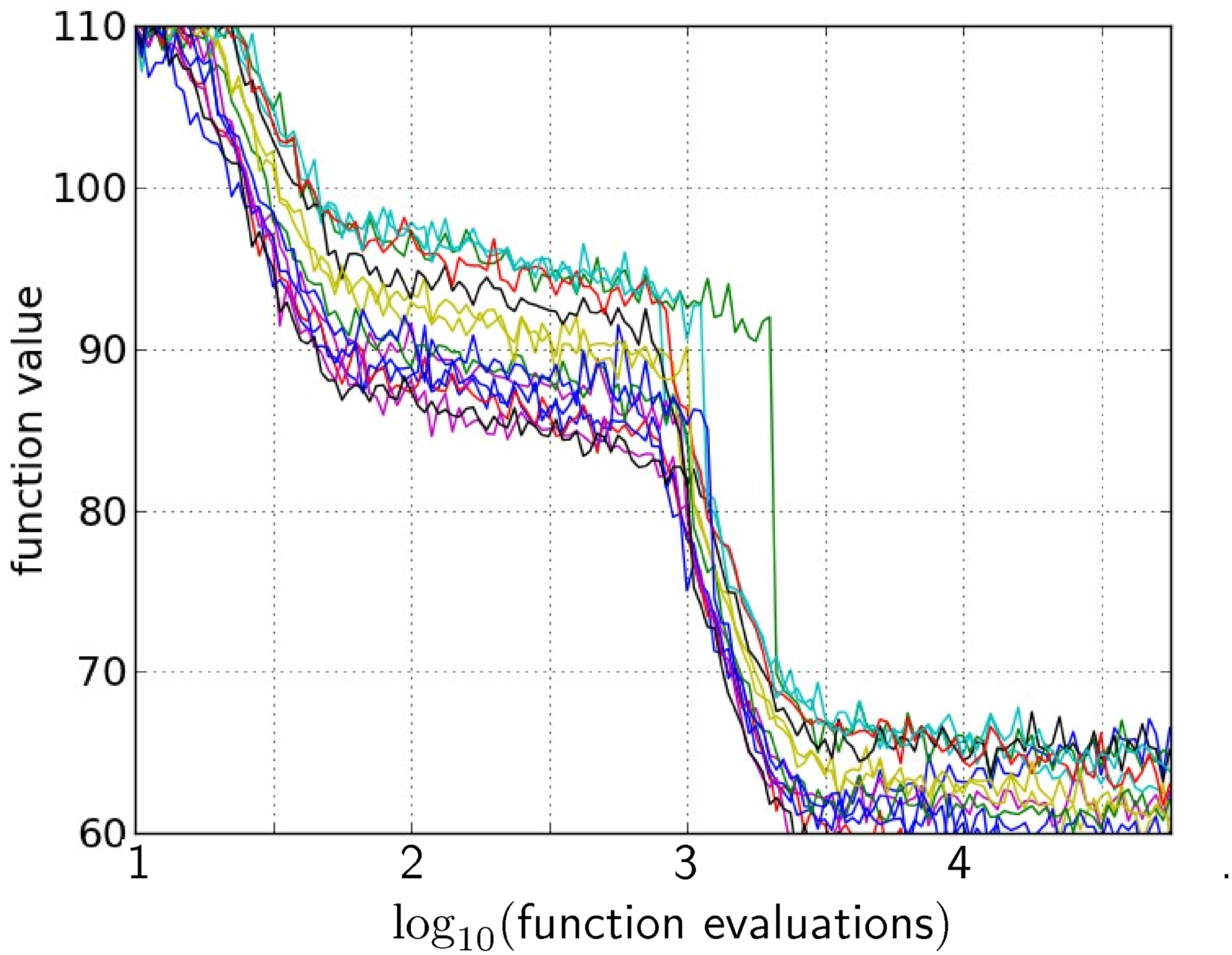

110
100
90
80
70
60
function value
1
2
3
4
$\log_{10}$(function evaluations)
target

15 runs

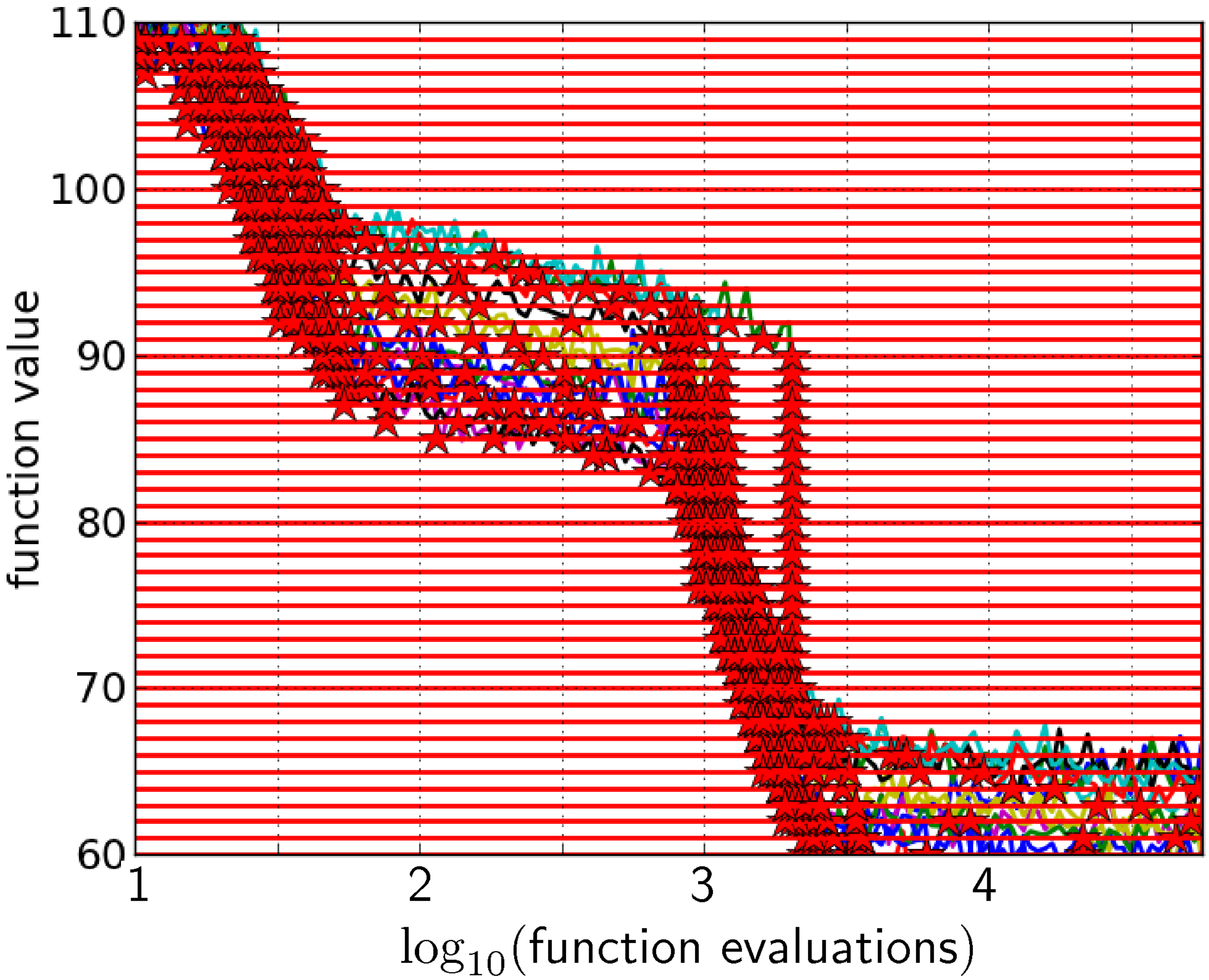


the ECDF of run lengths (runtimes)

80% of the runs reached the target

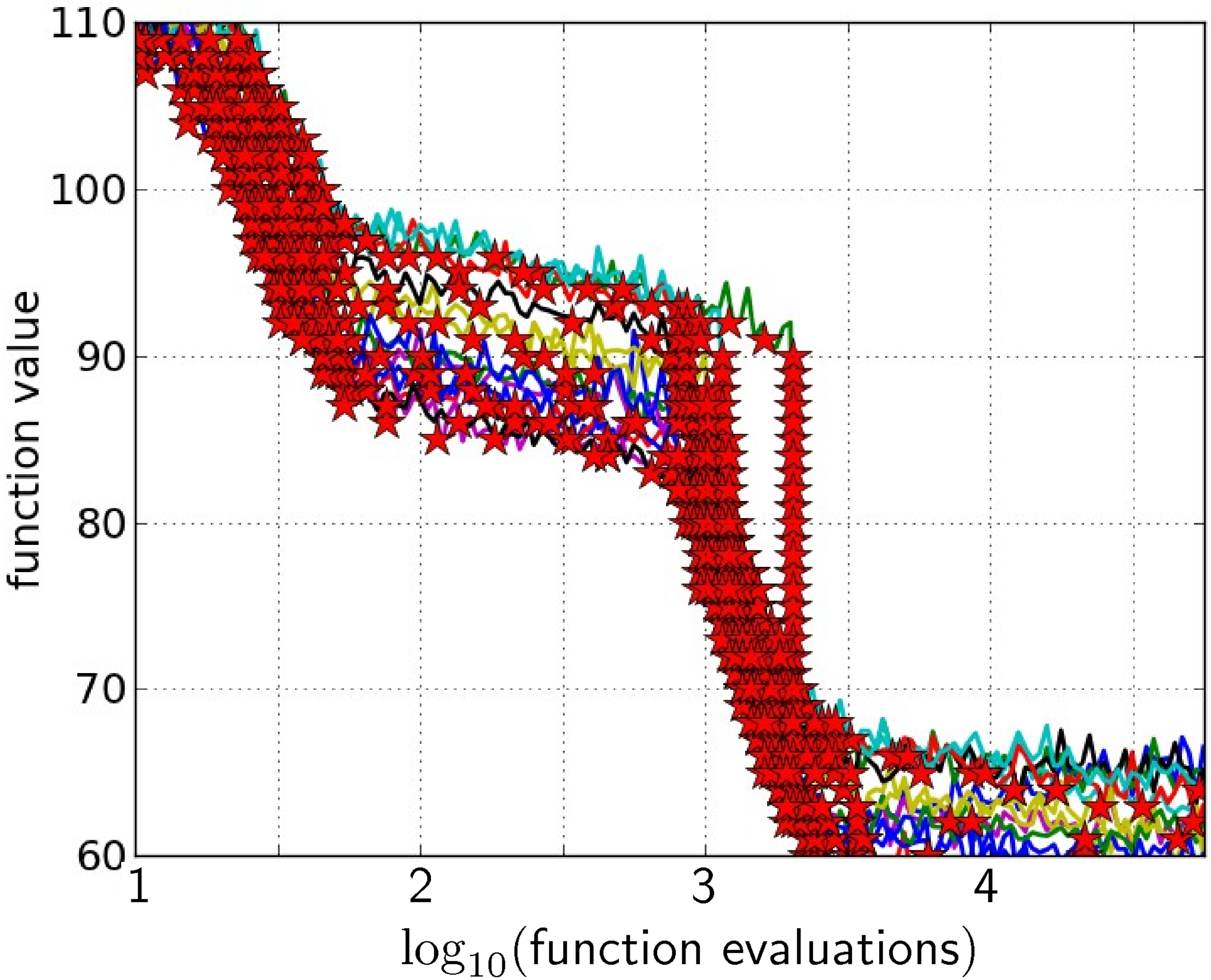

110
100
90
80
70
60
1
2
3
4
function value
$\log_{10}$(function evaluations)

15 runs



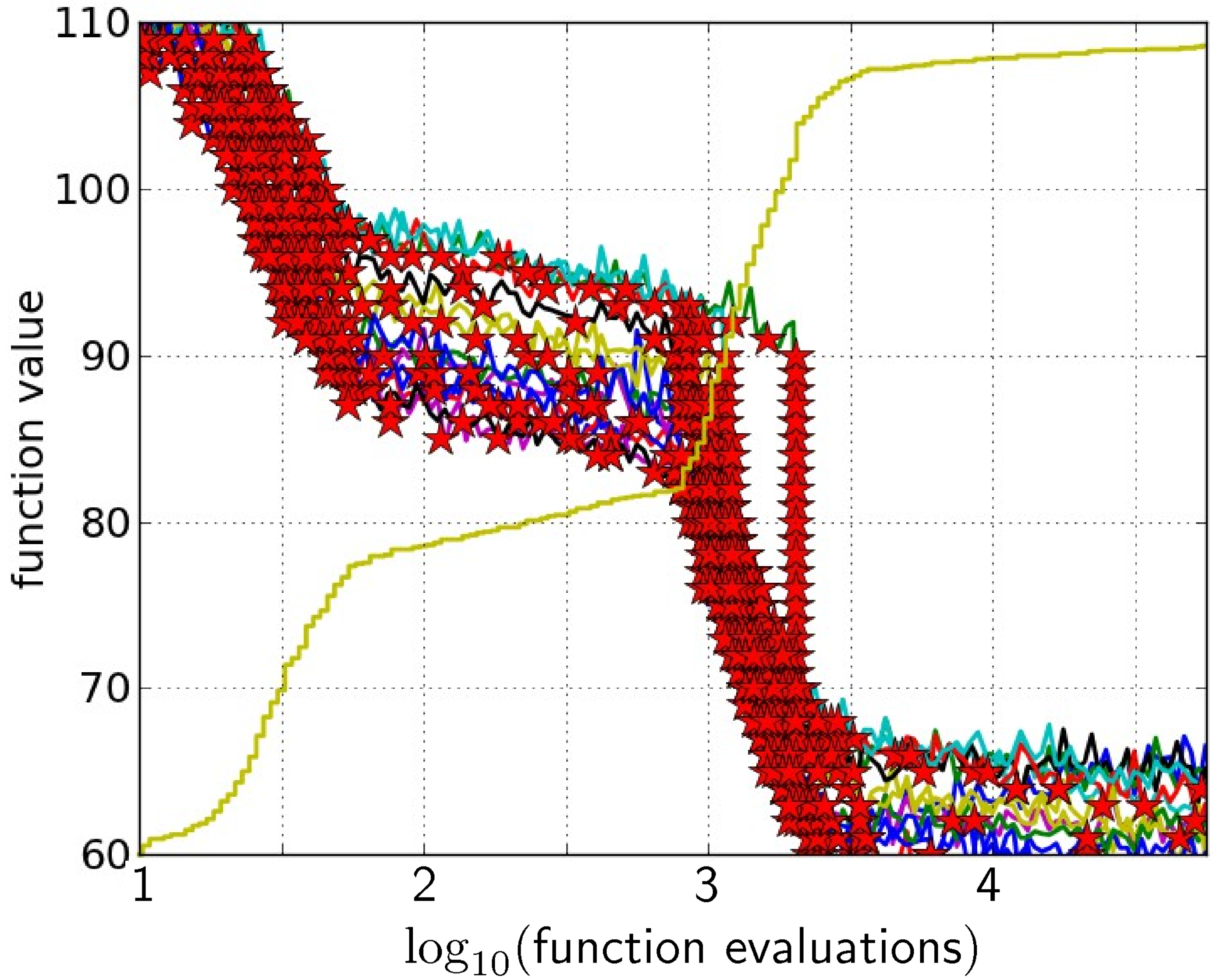
15 runs

50 targets



15 runs

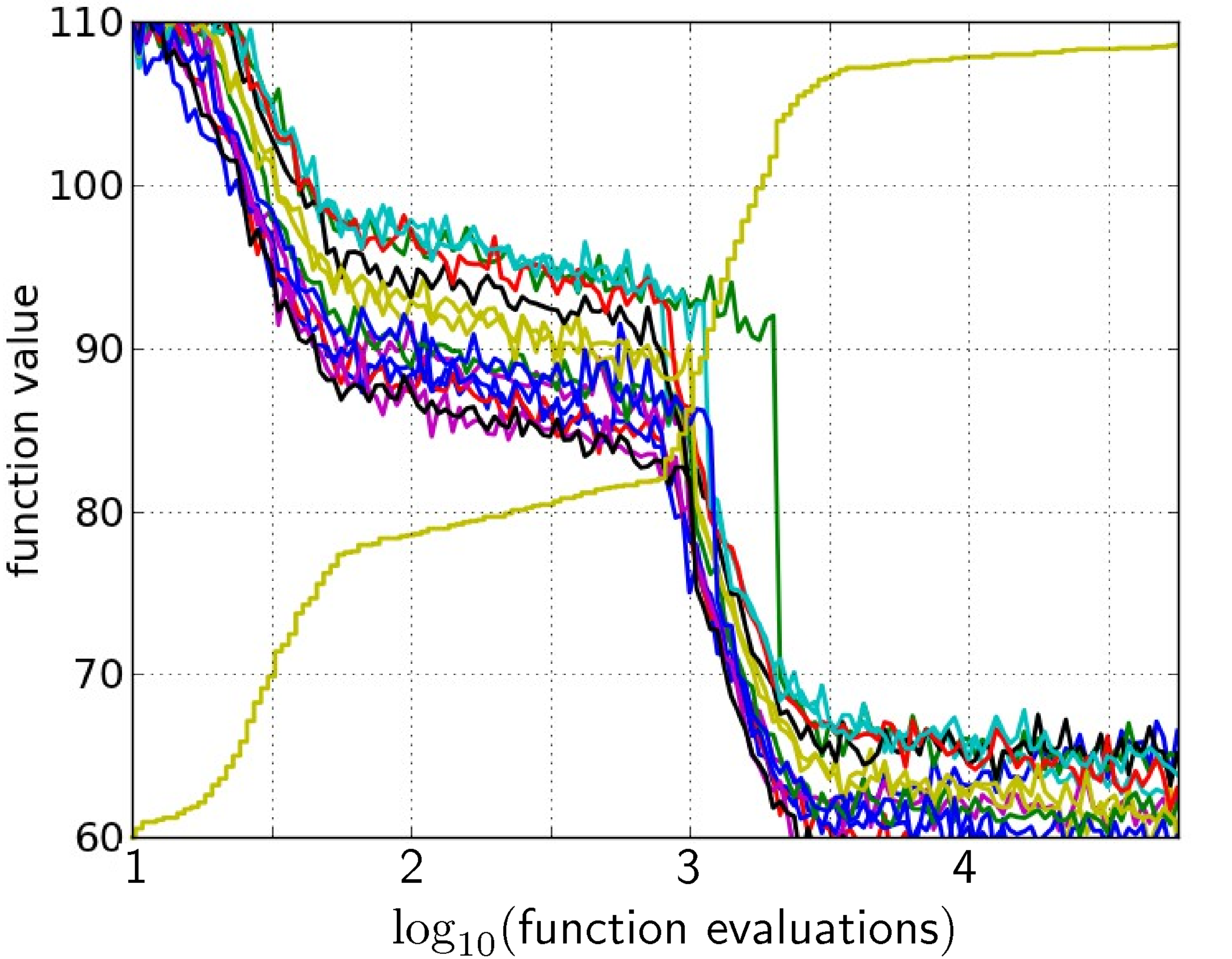
50 targets



15 runs

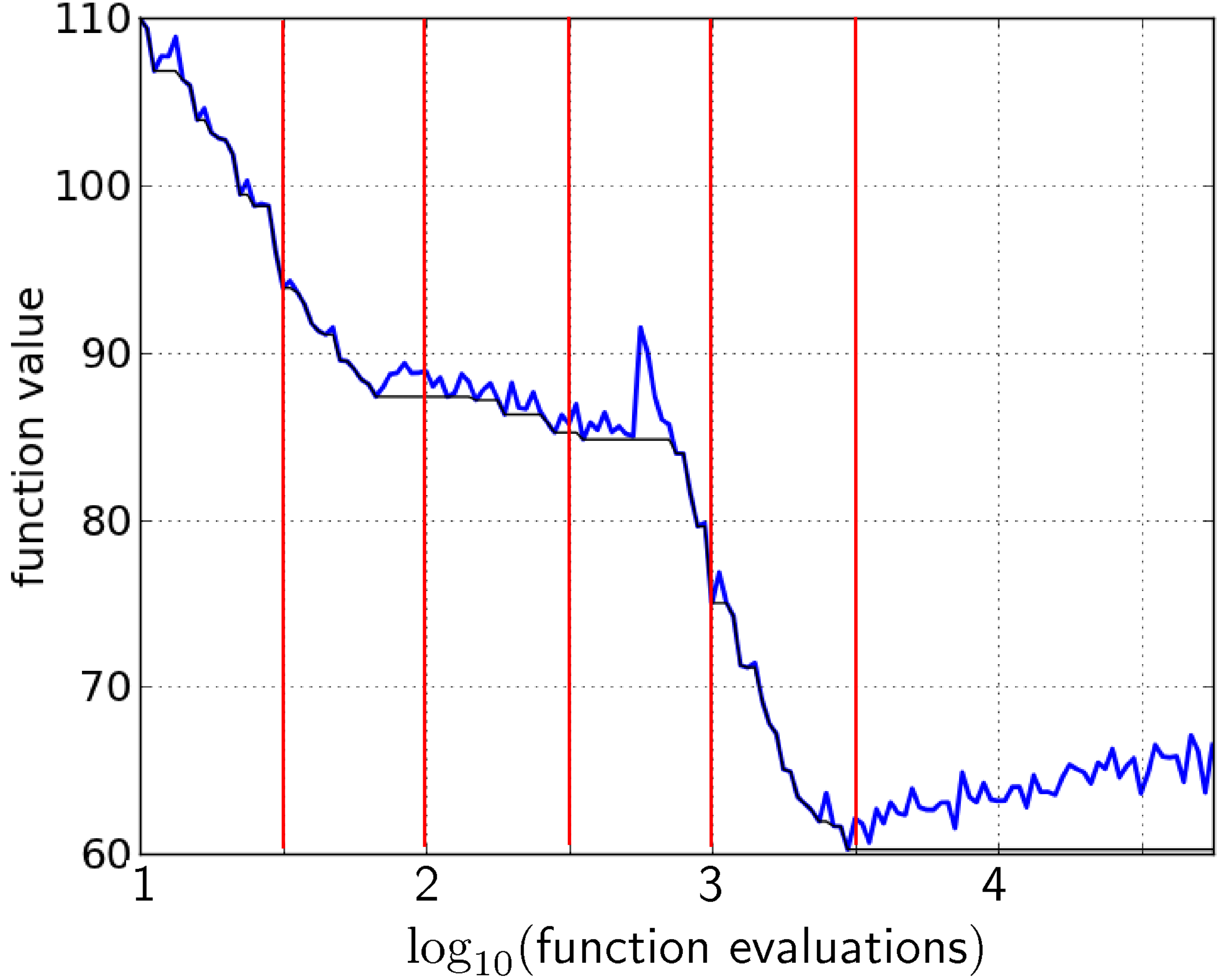
50 targets

ECDF with 750 steps



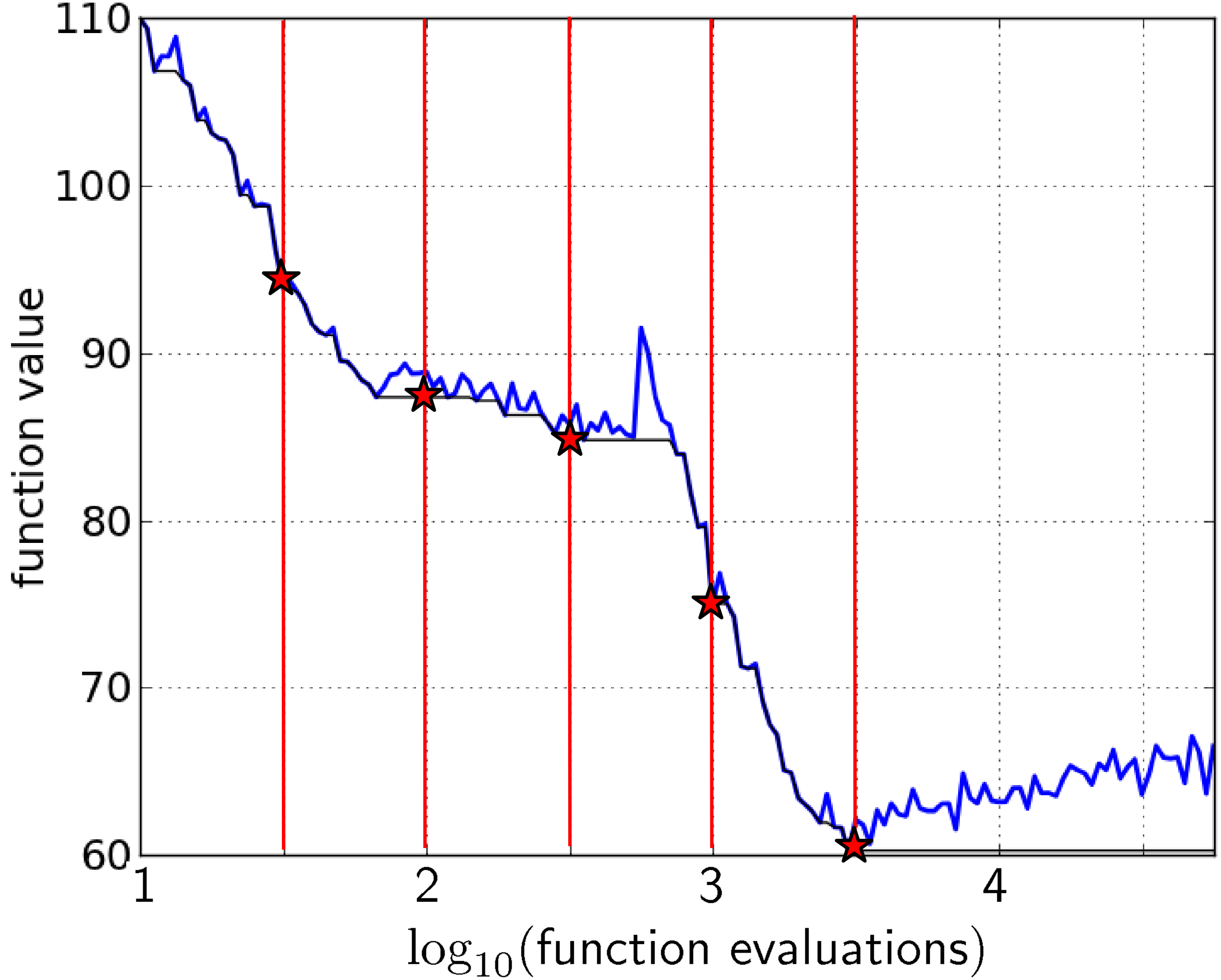
15 runs integrated in a single graph

Target budgets/run-lengths



1) define reference target budgets

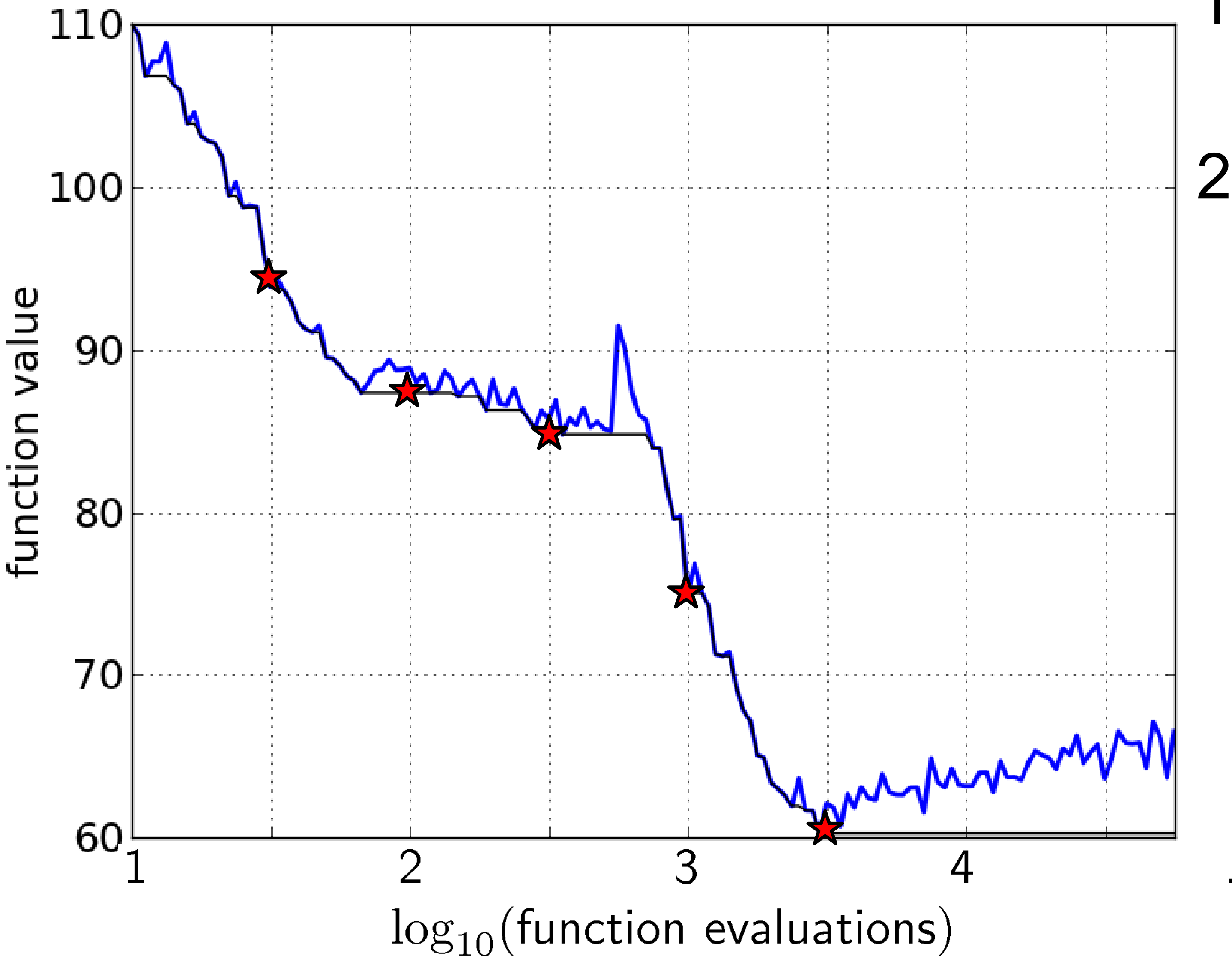
Target budgets on the reference algorithm



1) define reference target budgets

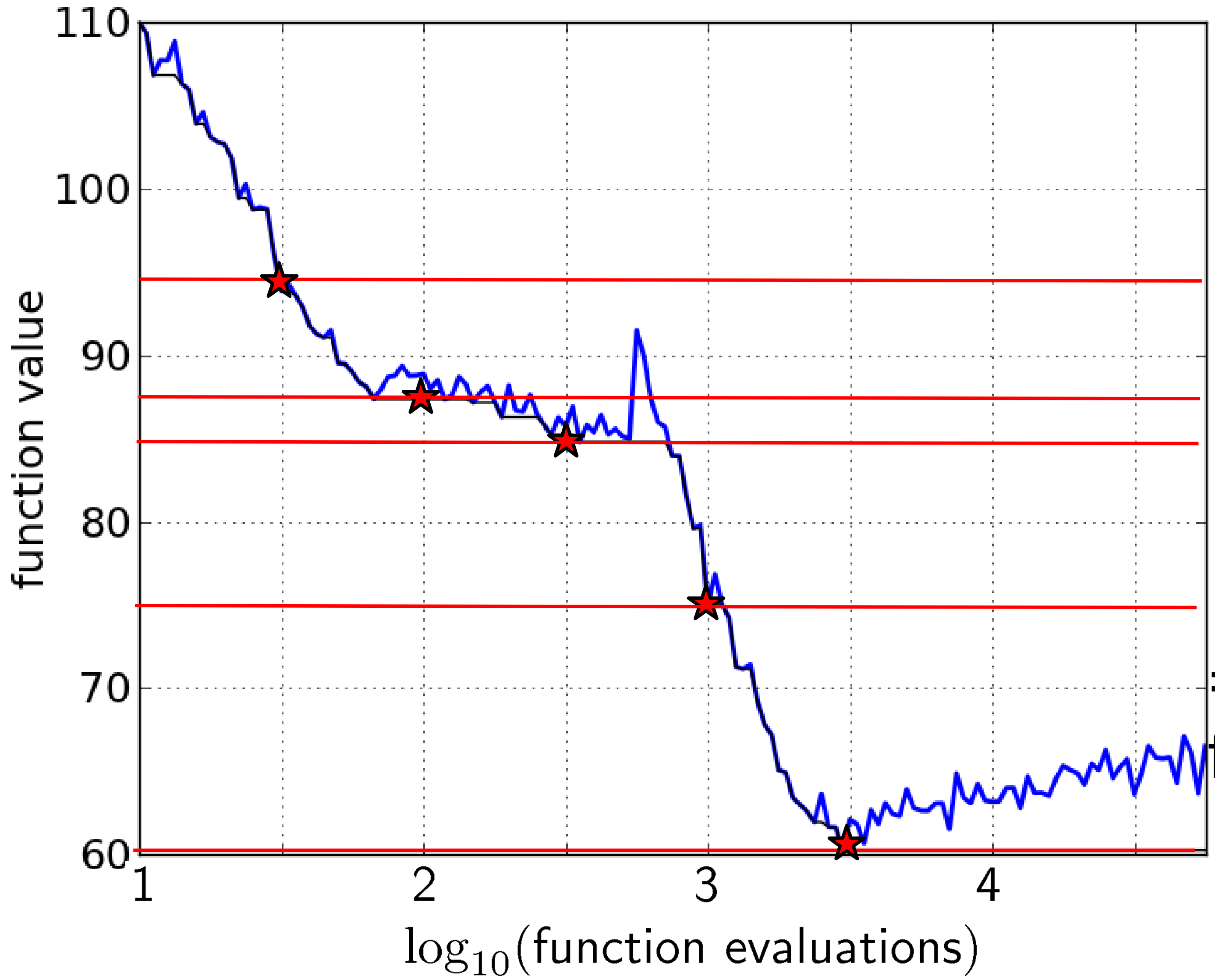
2) compute best function value achieved by a reference algorithm

Target budgets on the reference algorithm



1) define reference target budgets
2) compute best function value achieved by a reference algorithm

Run-length based target f-values

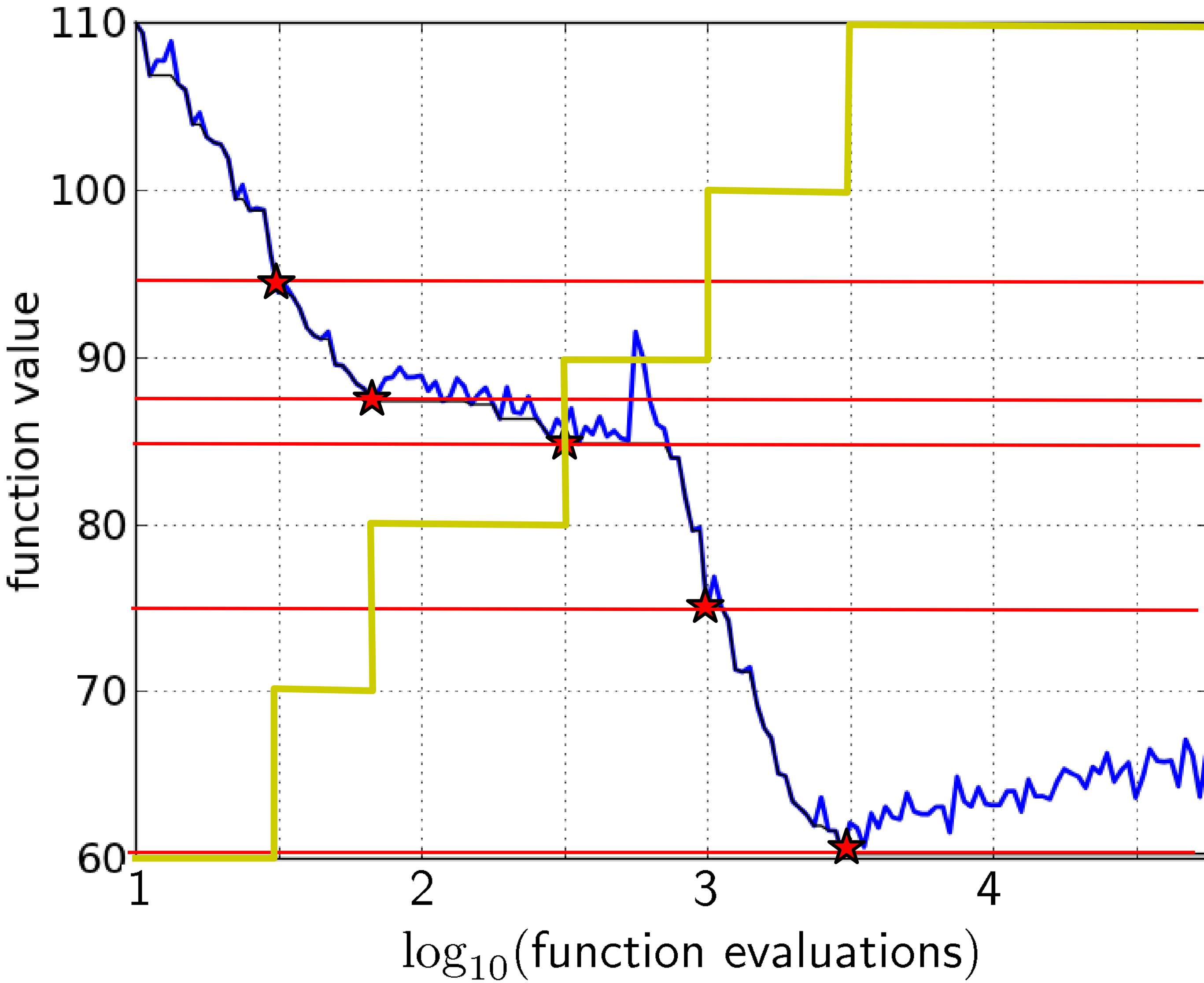


1) define reference target budgets

2) compute best function value achieved by a reference algorithm

=> set of target function values

Run-length based target f-values

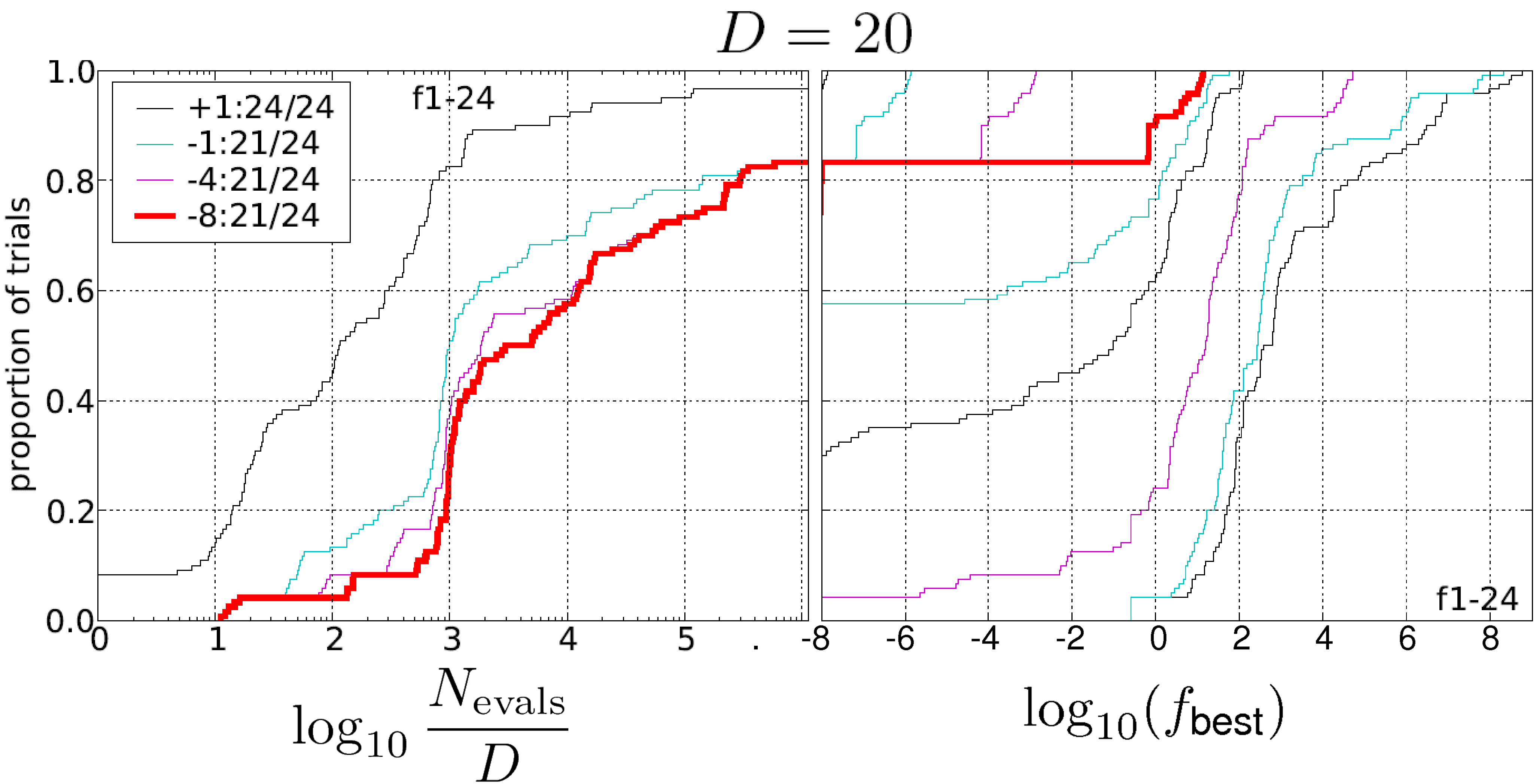


1) define reference target budgets

2) compute best function value achieved by a reference algorithm

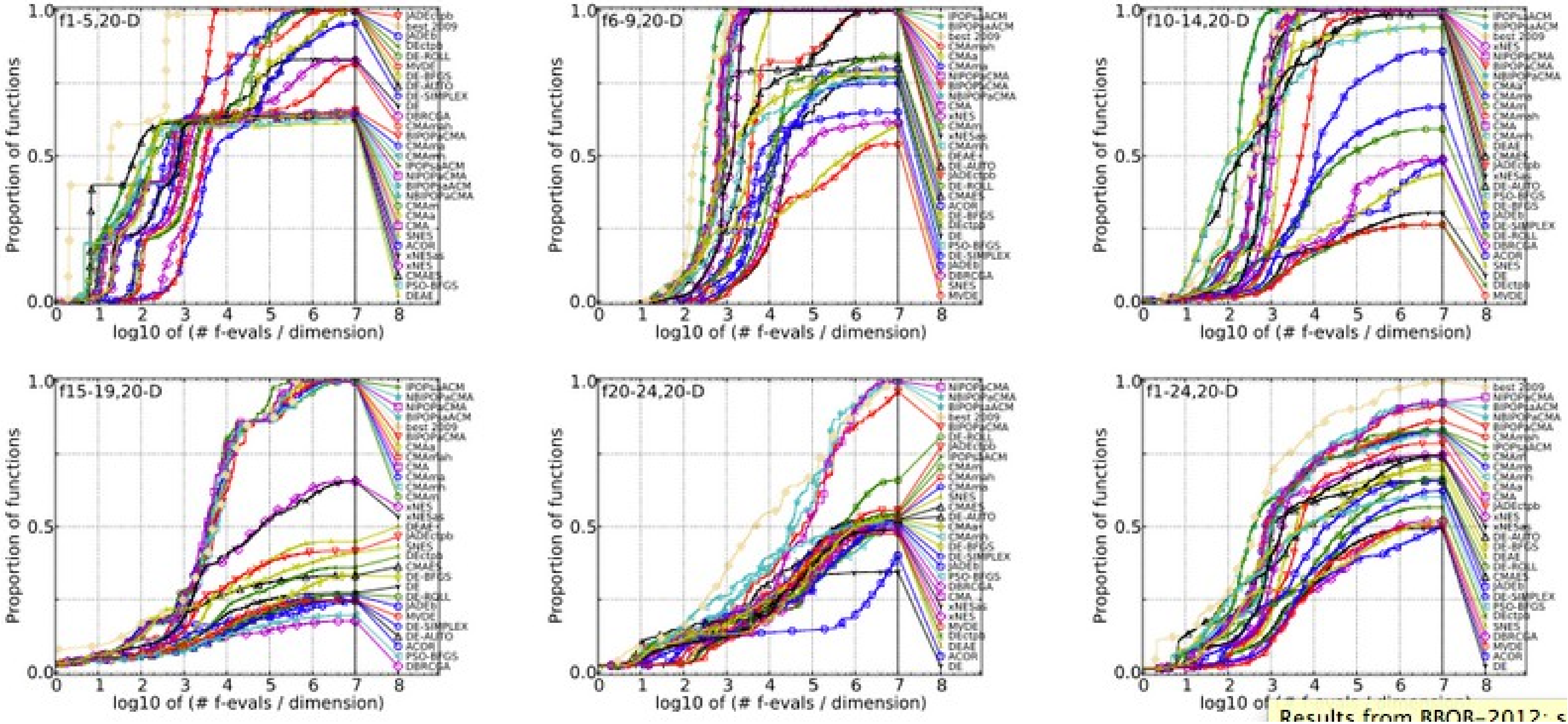
=> set of target function values

Example for ECDFs

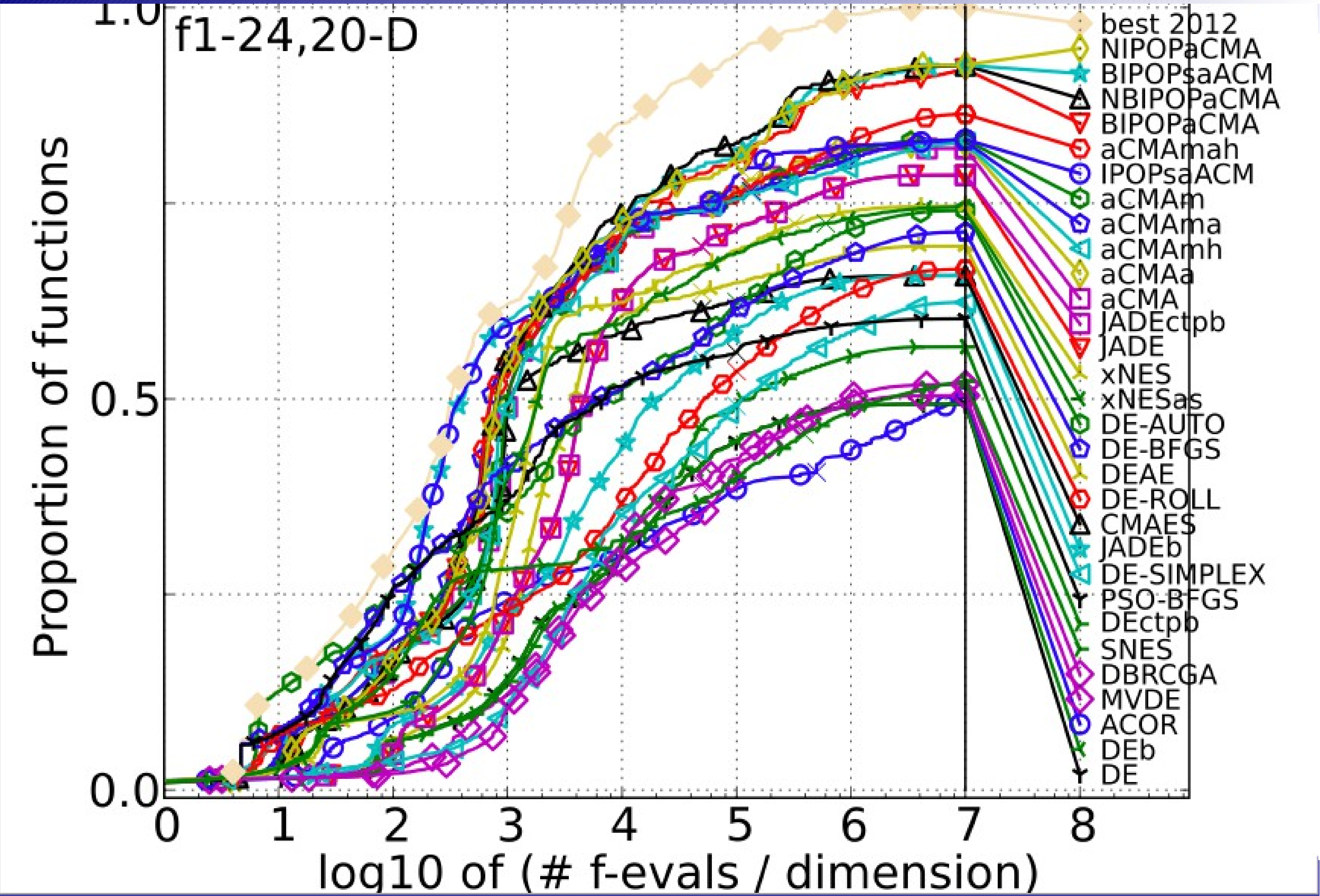


Empirical cumulative distribution functions (ECDFs) of running lengths (left) and function values (right)

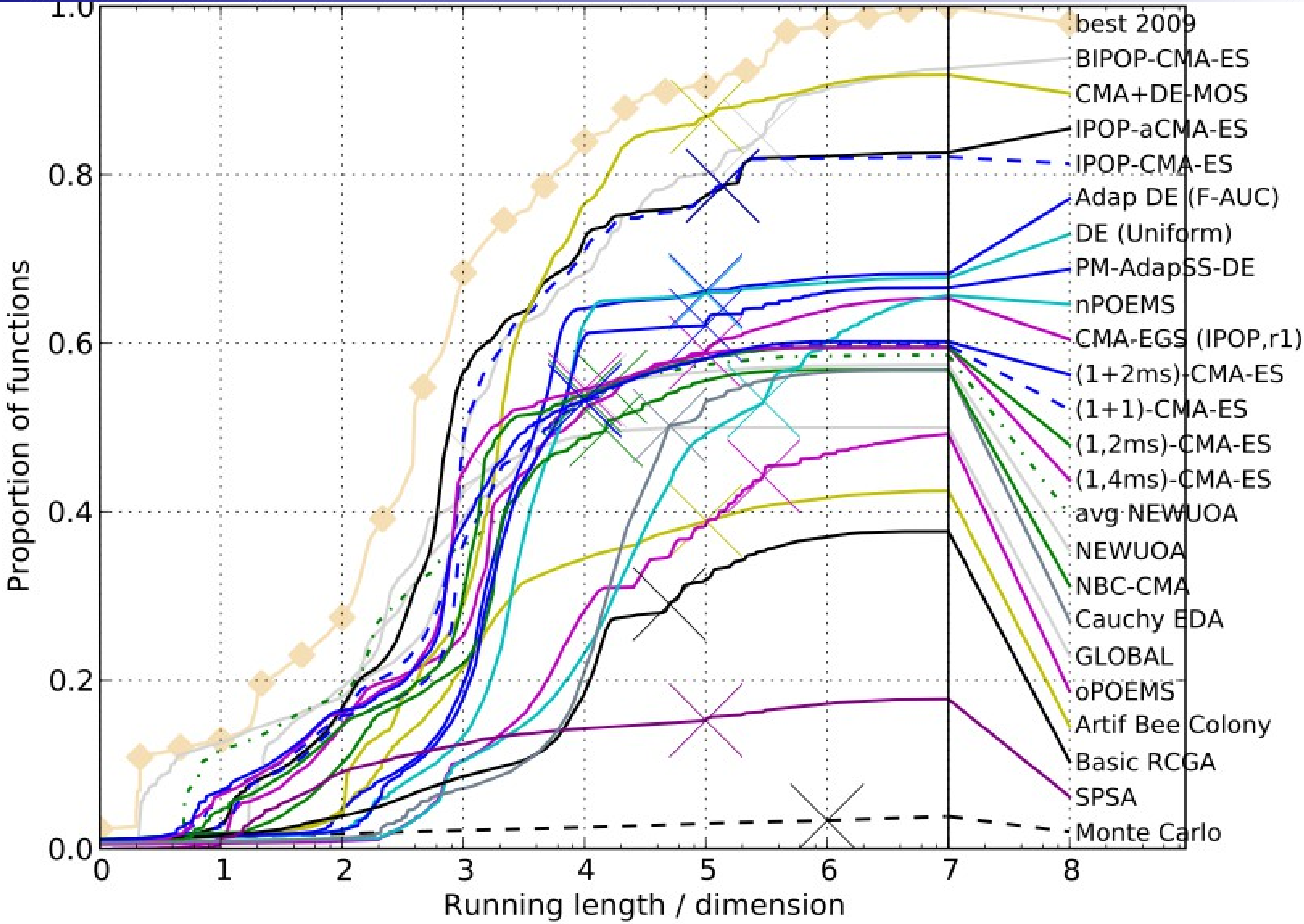
Overview results 2012



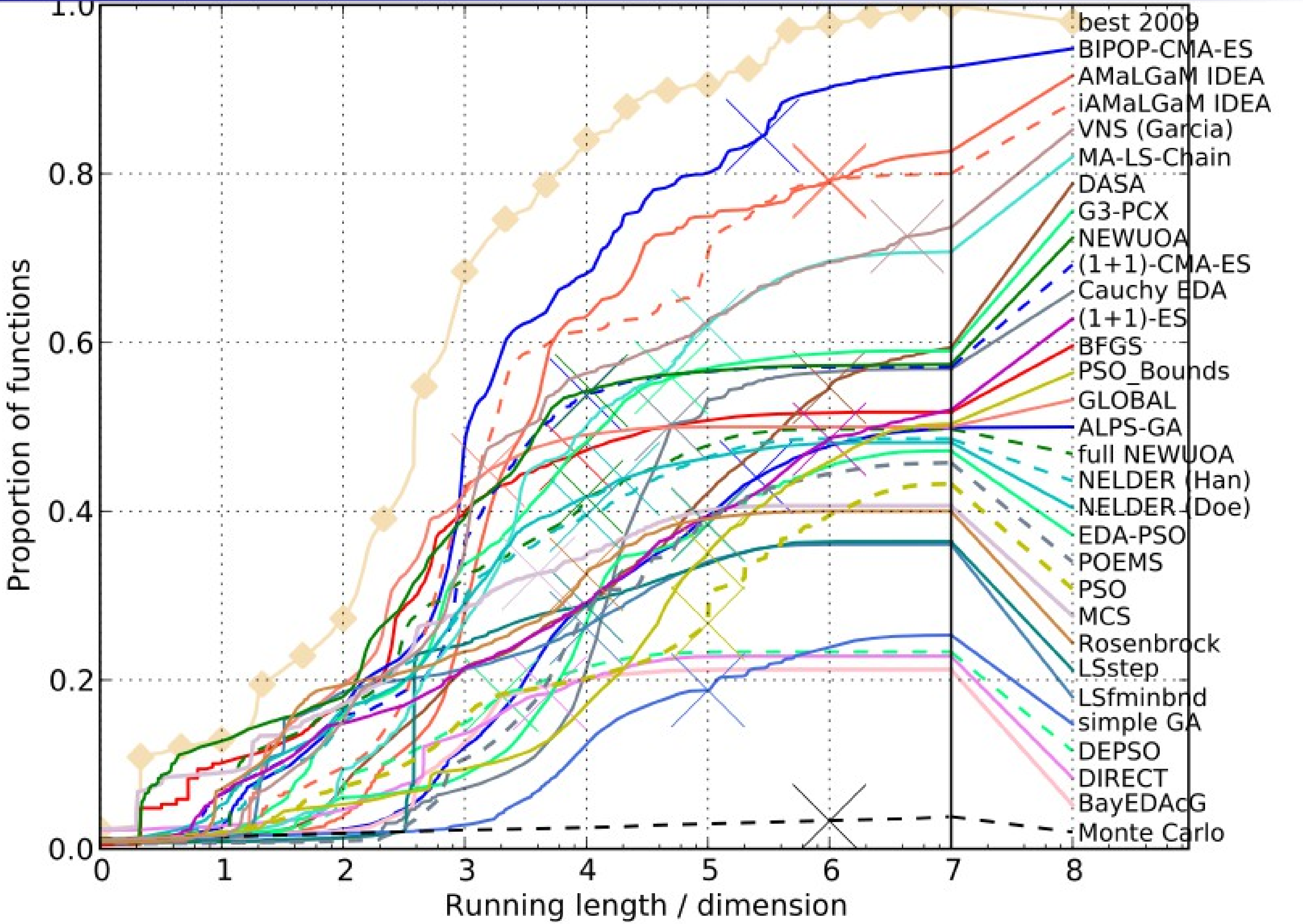
Results of 2012 (20-D)



Results of 2010 (20-D)



Results of 2009 (20-D)



ECDF: Summary

Empirical Cumulative Distribution Functions

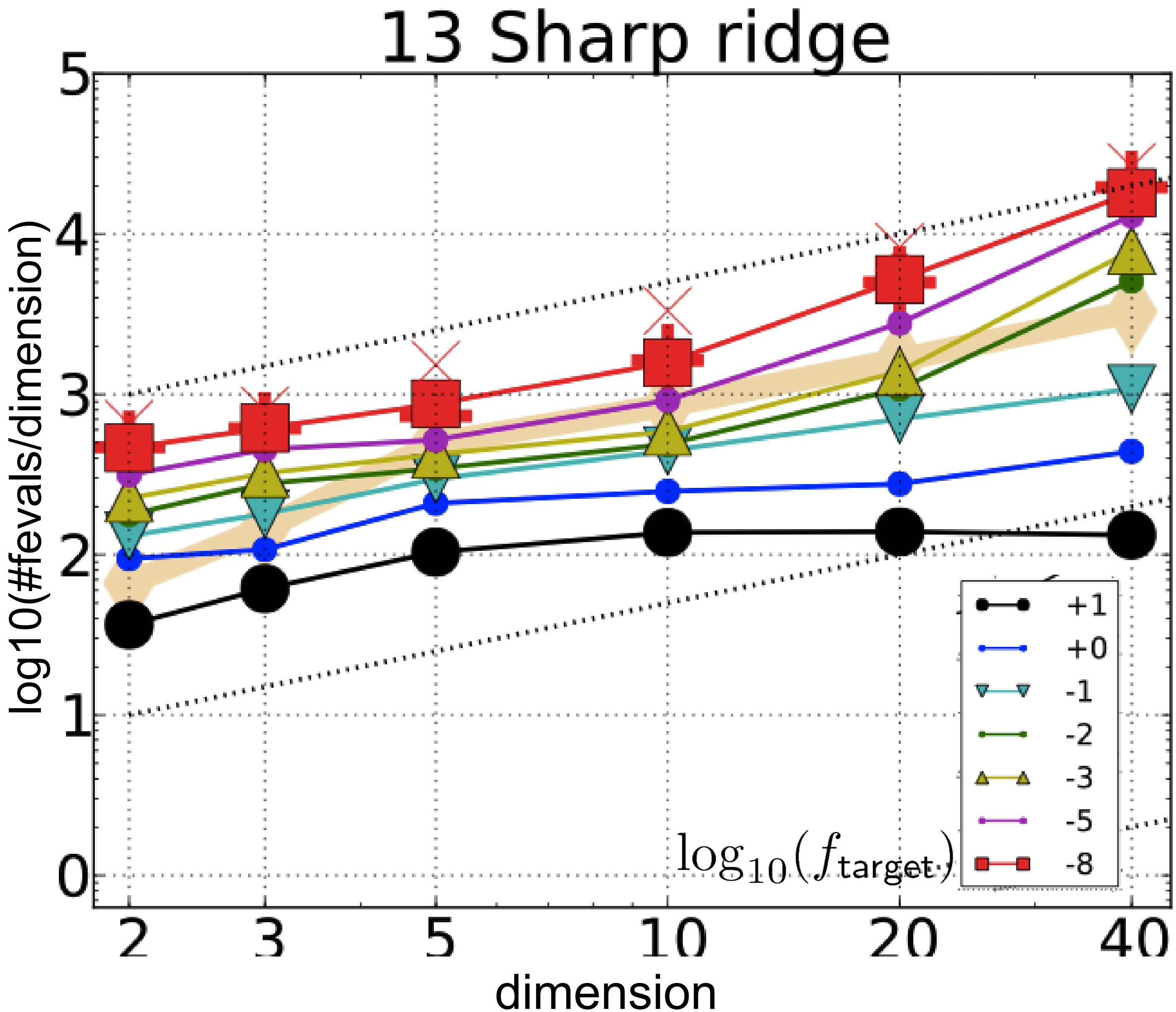
- recover a single convergence graph (and generalize)
- can aggregate over any set of functions and target values

they display a set of run lengths or runtimes (RT)

- for RT on a single problem (function & target value) allow to estimate any statistics of interest from them, like median, expectation (ERT),… in a meaningful way
- AKA data profile [Moré&Wild 2009]
- Performance profile [Dolan&Moré 2002]: ECDFs of run lengths divided by the smallest observed run length

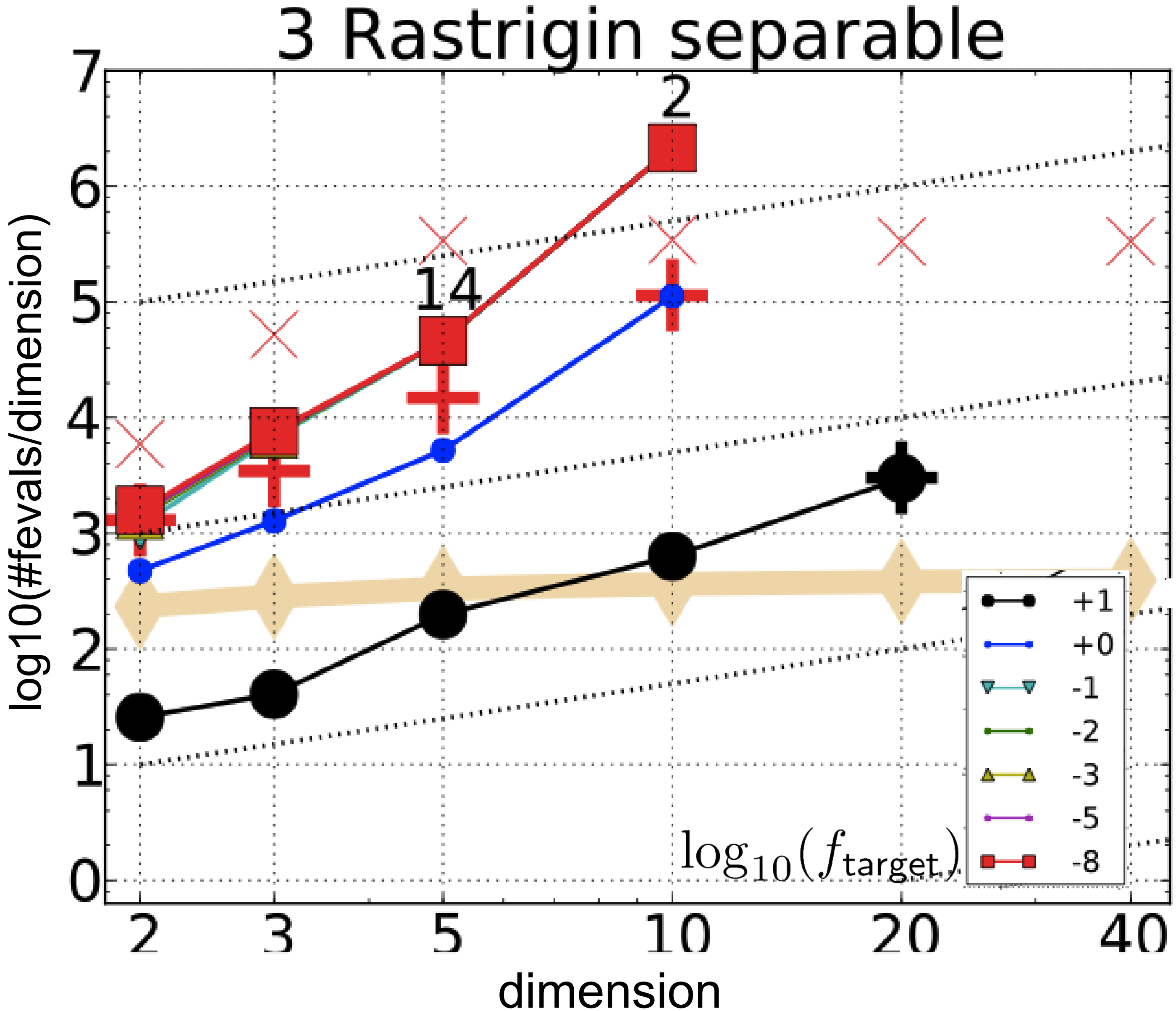
Different Displays of Runtimes

Scaling Behaviour with Dimension



- slanted grid lines: quadratic scaling
- horizontal lines: linear scaling
- light brown: artificial best 2009

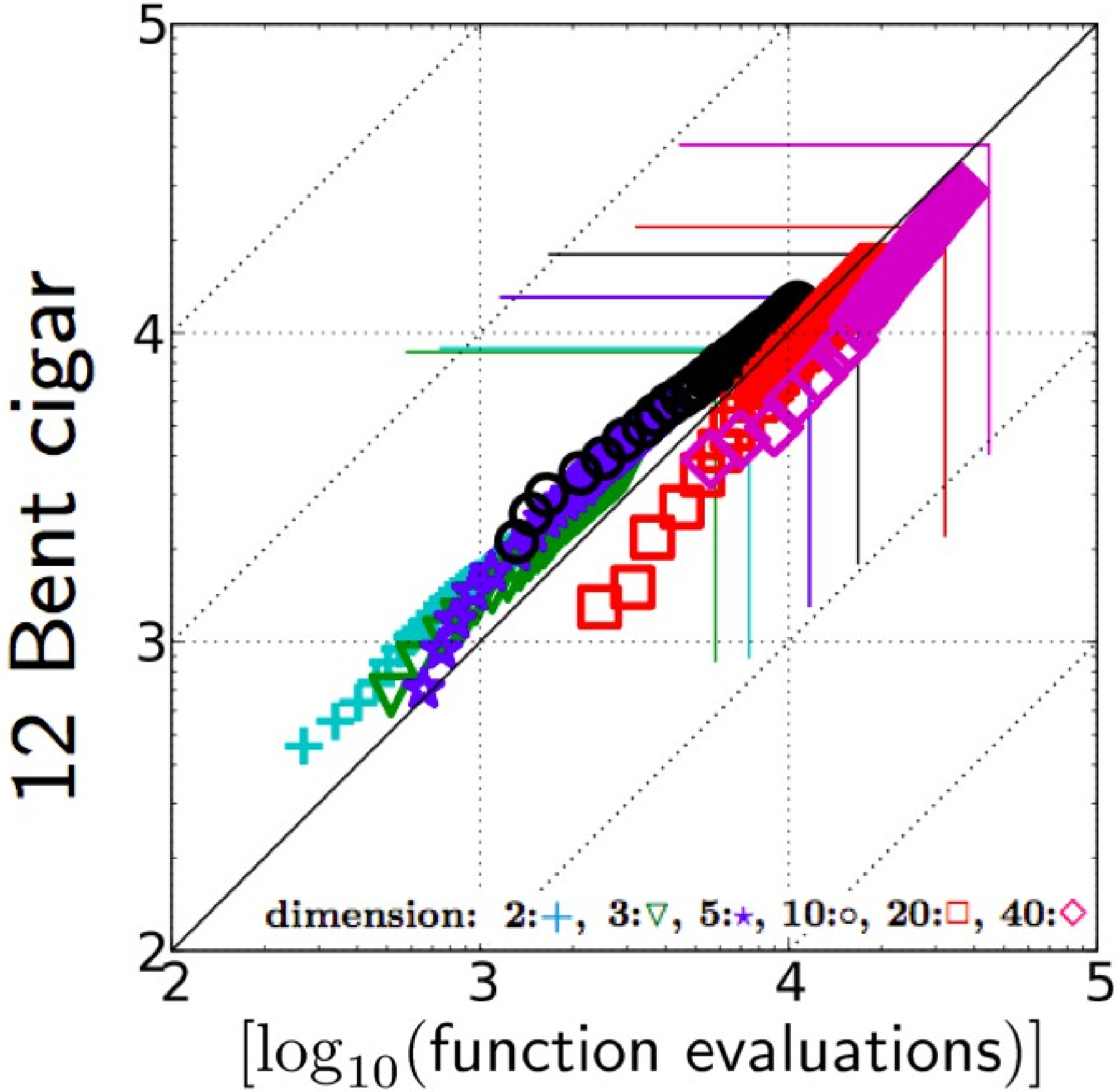
Example: Scaling Behaviour



- slanted grid lines: quadratic scaling
- horizontal lines: linear scaling
- light brown: artificial best 2009

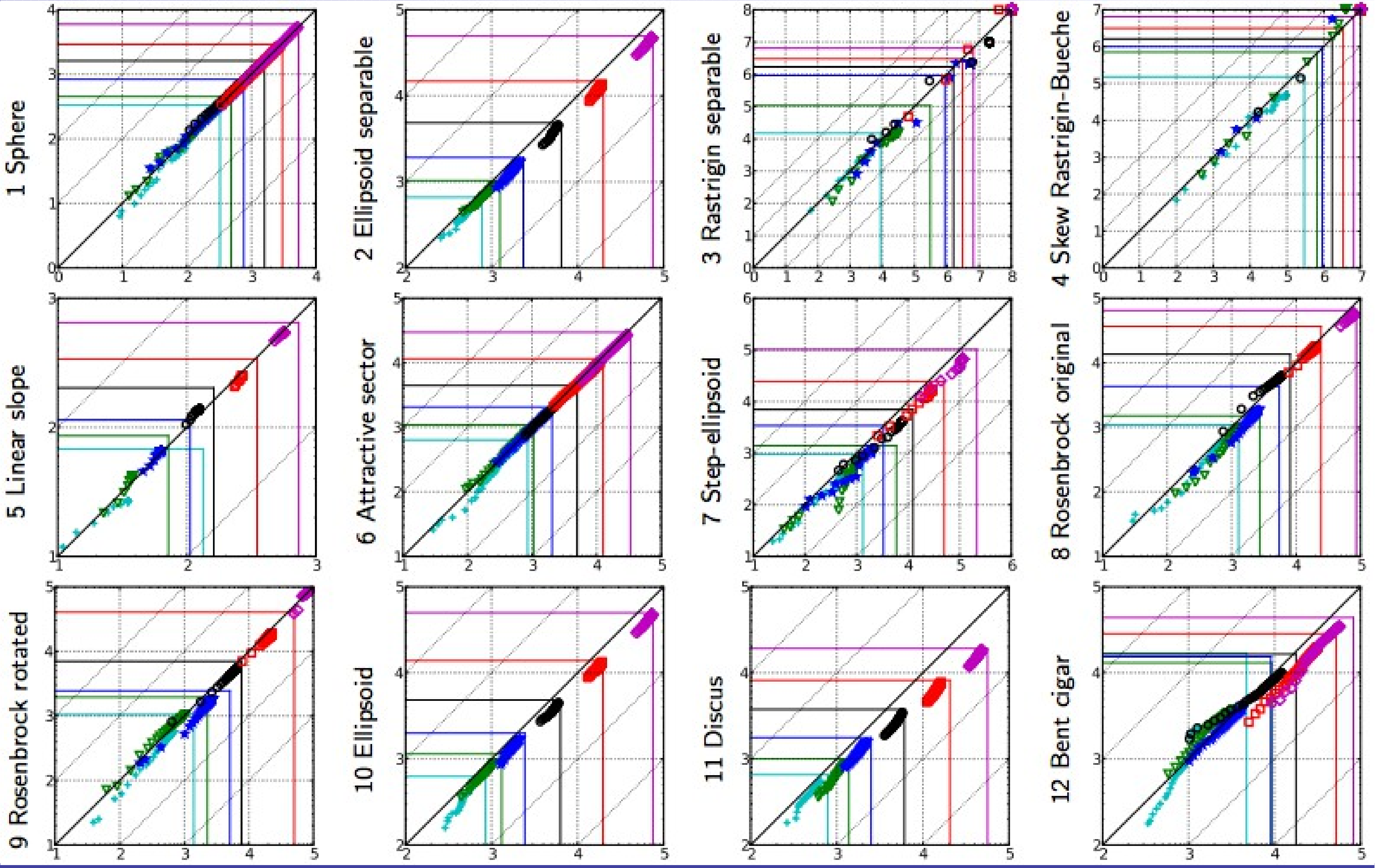
⟹ Experiments in >40-D are more often than not virtually superfluous

ERT scatter plots, all dimensions&targets



- estimated Expected Run Time (ERT), two algorithms
- 2-10 D: first algorithm "dominates"
- 20 & 40 D: second algorithm "dominates"

ERT scatter plots, all dimensions&targets



Single Function Table

Table 6: 20-D, running time excess ERT/ERT_{best} on f_6, in italics is given the median final function value and the median number of function evaluations to reach this value divided by dimension

6 Attractive sector

Δftarget	1e+03	1e+02	1e+01	1e+00	1e-01	1e-02	1e-03	1e-04	1e-05	1e-07	Δftarget
ERT_{best}/D	4.03	26	64.7	87.2	123	152	184	219	248	309	ERT_{best}/D
ALPS	59	25	34	54	64	78	100	150	370	*14e-7/2e5*	ALPS [17]
AMaLGaM IDEA	26	22	19	22	21	22	22	21	22	22	AMaLGaM IDEA [4]
avg NEWUOA	**2.3**	**1.1**	1	**1**	1	**1**	1	**1**	**1**	1	avg NEWUOA [31]
BayEDAcG	46	41	*60e+0/2e3*	.	.	.	.	.	.	.	BayEDAcG [10]
BFGS	**2.2**	**2.7**	3.6	4.7	4.7	4.9	5	4.8	4.9	61	BFGS [30]
Cauchy EDA	6200	1500	1e3	1700	*17e-1/5e4*	.	.	.	.	.	Cauchy EDA [24]
BIPOP-CMA-ES	**2.9**	**2.2**	**1.5**	**1.7**	**1.6**	**1.6**	**1.6**	**1.5**	**1.6**	**1.6**	BIPOP-CMA-ES [15]
(1+1)-CMA-ES	**1.9**	4.5	13	180	1200	*13e-1/1e4*	.	.	.	.	(1+1)-CMA-ES [2]
DASA	12	6.8	9.9	19	25	33	49	58	63	74	DASA [19]
DEPSO	11	7.5	12	64	*13e-1/2e3*	.	.	.	.	.	DEPSO [12]
DIRECT	18	31	*40e+0/5e3*	.	.	.	.	.	.	.	DIRECT [25]
EDA-PSO	27	46	40	45	44	44	44	44	44	44	EDA-PSO [6]
full NEWUOA	5	**1.9**	**1.5**	**1.4**	**1.4**	**1.4**	**1.4**	**1.4**	**1.4**	**1.4**	full NEWUOA [31]
G3-PCX	4.1	**1.4**	**1.4**	**2**	**2.1**	**2.1**	**2.2**	**2.2**	**2.3**	**2.4**	G3-PCX [26]
simple GA	320	130	2e3	*11e+0/1e5*	.	.	.	.	.	.	simple GA [22]
GLOBAL	5	**2.9**	3.6	4.9	8.5	*42e-3/2e3*	.	.	.	.	GLOBAL [23]
iAMaLGaM IDEA	5.1	5.6	5.4	6.8	7.1	7.7	7.8	7.7	8	8.3	iAMaLGaM IDEA [4]
LSfminbnd	9	31	160	760	1100	960	*72e-1/1e4*	.	.	.	LSfminbnd [28]
LSstep	140	260	2300	*59e+0/1e4*	.	.	.	.	.	.	LSstep [28]
MA-LS-Chain	11	4.9	7.5	8.9	8	7.7	7.2	6.7	6.5	6	MA-LS-Chain [21]
MCS (Neum)	**1.8**	33	*42e+0/4e3*	.	.	.	.	.	.	.	MCS (Neum) [18]
NELDER (Han)	**2.2**	**2.4**	**2.7**	3.3	3.2	3.5	3.5	3.5	4	7.4	NELDER (Han) [16]
NELDER (Doe)	**1.5**	**2.3**	9.1	20	28	65	110	430	*46e-5/2e4*	.	NELDER (Doe) [5]
NEWUOA	**1**	**1**	1	**1.3**	**1.4**	**1.5**	**1.6**	**1.6**	**1.7**	**1.7**	NEWUOA [31]
(1+1)-ES	**2**	**2.2**	**2.1**	**2.8**	3.9	5.2	6.1	6.5	6.4	6.7	(1+1)-ES [1]
POEMS	89	26	31	37	36	36	36	35	36	37	POEMS [20]
PSO	6.4	280	1100	1400	980	820	710	620	570	790	PSO [7]
PSO_Bounds	9.5	45	120	150	140	140	140	130	160	220	PSO_Bounds [8]
Monte Carlo	2.4e5	*48e+1/1e6*	.	.	.	.	.	.	.	.	Monte Carlo [3]
Rosenbrock	**2.1**	3.9	31	76	210	230	810	*21e-2/1e4*	.	.	Rosenbrock [27]
IPOP-SEP-CMA-ES	3.2	**2.1**	**1.7**	**1.9**	**1.9**	**1.9**	**1.9**	**1.9**	**2**	**2**	IPOP-SEP-CMA-ES [29]
VNS (Garcia)	5	**2.8**	**1.9**	**1.9**	**1.7**	**1.7**	**1.7**	**1.6**	**1.6**	**1.6**	VNS (Garcia) [11]

Questions?

Python

Python

- a general-purpose, well-designed, modern high-level programming language
- dynamically-typed, highly object-oriented (not enforced), highly modularized
- for scripting, for programming, for interactive usage
- comes with thousands of packages
- the Python *programming language* is much better designed than Matlab/Octave
- IPython can replace Matlab/Octave for interactive usage
- (I)Python is free and available on almost every computer

Popularity of Programming Languages (TIOBE)

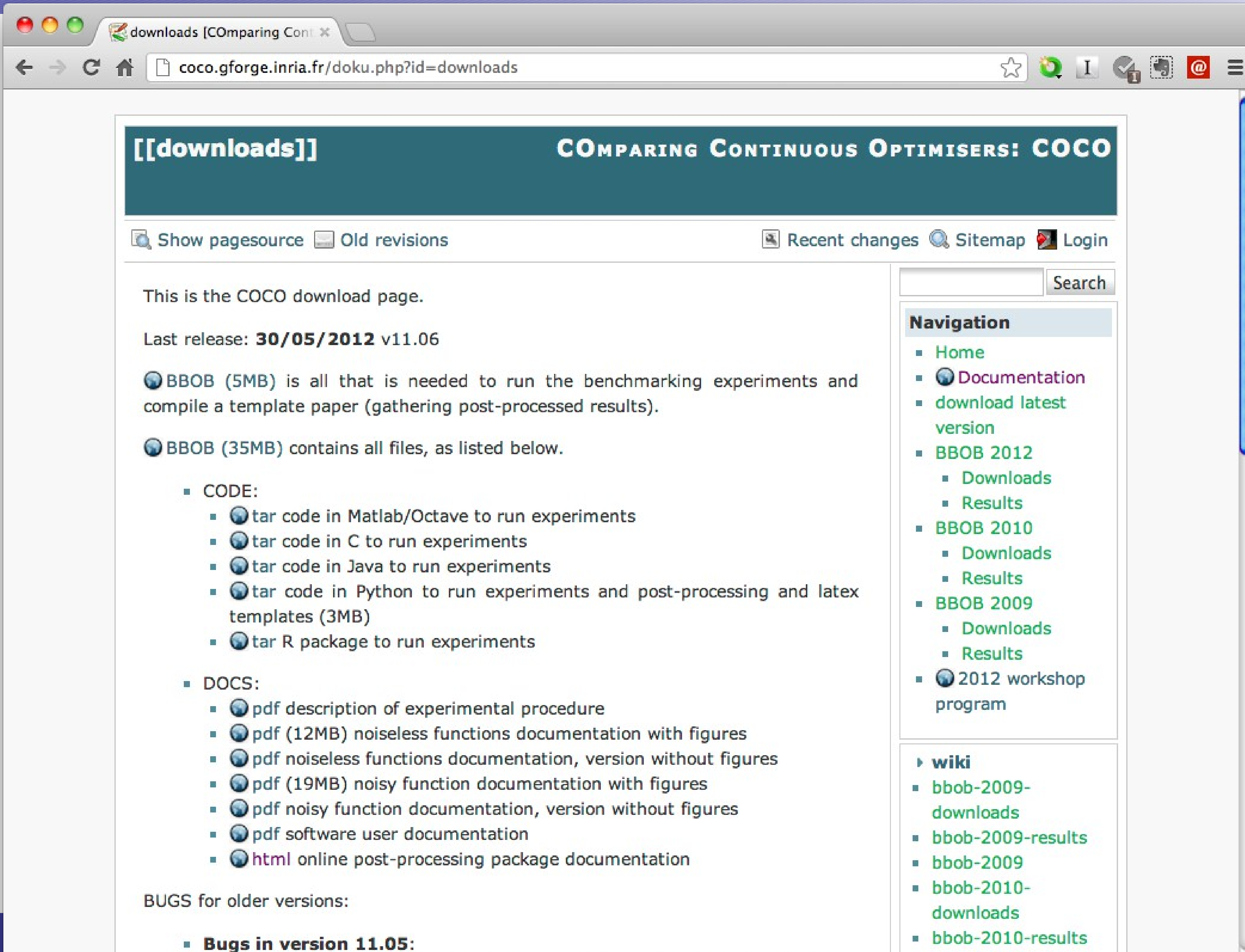
www.tiobe.com/index.php/content/paperinfo/tpci/index....

Position May 2013	Position May 2012	Delta in Position	Programming Language	Ratings May 2013	Delta May 2012
1	1	=	C	18.729%	+1.38%
2	2	=	Java	16.914%	+0.31%
3	4	↑	Objective-C	10.428%	+2.12%
4	3	↓	C++	9.198%	-0.63%
5	5	=	C#	6.119%	-0.70%
6	6	=	PHP	5.784%	+0.07%
7	7	=	(Visual) Basic	4.656%	-0.80%
8	8	=	Python	4.322%	+0.50%
9	9	=	Perl	2.276%	-0.53%
10	11	↑	Ruby	1.670%	+0.22%
11	10	↓	JavaScript	1.536%	-0.60%
12	12	=	Visual Basic .NET	1.131%	-0.14%
20	22	↑↑	MATLAB	0.563%	0.00%
24			R	0.480%	

BBOB with COCO in practice (for dummies)

COCO (COmparing Continuous Optimizers): a tool for black-box optimization benchmarking

BBOB in practice



BBOB in practice

Name	Date Modified	Size	Kind
bbob.v11.06	Today, 1:15	--	Folder
c	May 30, 2012 12:07	--	Folder
docs	October 27, 2012 0:58	--	Folder
java	May 30, 2012 12:07	--	Folder
latextemplates	May 30, 2012 12:06	--	Folder
matlab	May 30, 2012 12:06	--	Folder
python	May 30, 2012 12:06	--	Folder
r	May 30, 2012 12:07	--	Folder

Macintosh HD ▸ Users ▸ hansen ▸ Downloads ▸ bbob.v11.06

BBOB in practice

Name	Date Modified	Size	Kind
bbob.v11.06	Today, 1:15	--	Folder
c	May 30, 2012 12:07	--	Folder
docs	October 27, 2012 0:58	--	Folder
java	May 30, 2012 12:07	--	Folder
latextemplates	May 30, 2012 12:06	--	Folder
matlab	May 30, 2012 12:06	--	Folder
benchmarkinfos.txt	February 9, 2009 16:29	4 KB	Gedit …ument
benchmarks.m	February 10, 2011 16:24	86 KB	Objec…rce File
benchmarksnoisy.m	February 10, 2011 16:24	102 KB	Objec…rce File
exampleexperiment.m	February 1, 2012 19:52	4 KB	Objec…rce File
exampletiming.m	March 7, 2012 14:40	4 KB	Objec…rce File
fgeneric.m	December 7, 2011 17:56	33 KB	Objec…rce File
LICENSE.txt	February 1, 2012 20:23	4 KB	Gedit …ument
MY_OPTIMIZER.m	February 14, 2011 19:30	4 KB	Objec…rce File
README.txt	May 19, 2011 10:47	4 KB	Gedit …ument
python	May 30, 2012 12:06	--	Folder
r	May 30, 2012 12:07	--	Folder

Macintosh HD ▸ Users ▸ hansen ▸ Downloads ▸ bbob.v11.06

Matlab script (`exampleexperiment.m`):

```
dimensions = [2, 3, 5, 10, 20, 40];  % small dimensions first, for CPU reasons
functions = benchmarks('FunctionIndices');  % or benchmarksnoisy(...)
instances = [1:5, 31:40];  % 15 function instances

for dim = dimensions
  for ifun = functions
    for iinstance = instances
      fgeneric('initialize', ifun, iinstance, datapath, opt);
      MY_OPTIMIZER('fgeneric', dim, fgeneric('ftarget'), eval(maxfunevals) - f
      disp(sprintf(['  f%d in %d-D, instance %d: FEs=%d with %d restarts, fbes
      fgeneric('finalize');
    end
    disp(['      date and time: ' num2str(clock, ' %.0f')]);
  end
  disp(sprintf('---- dimension %d-D done ----', dim));
end
```

BBOB in practice

Running the experiment at an OS shell:

```
$ nohup nice octave < exampleexperiment.m > output.txt &
$ less output.txt
```

```
GNU Octave, version 3.6.3
Copyright (C) 2012 John W. Eaton and others.
This is free software; see the source code for copying conditions.
[...]
Read http://www.octave.org/bugs.html to learn how to submit bug reports.

For information about changes from previous versions, type `news'.

  f1 in 2-D, instance 1: FEs=242, fbest-ftarget=-8.1485e-10, elapsed time [h]: 0.00
  f1 in 2-D, instance 2: FEs=278, fbest-ftarget=-6.0931e-09, elapsed time [h]: 0.00
  f1 in 2-D, instance 3: FEs=242, fbest-ftarget=-9.2281e-09, elapsed time [h]: 0.00
  f1 in 2-D, instance 4: FEs=302, fbest-ftarget=-4.5997e-09, elapsed time [h]: 0.00
  f1 in 2-D, instance 5: FEs=230, fbest-ftarget=-9.8350e-09, elapsed time [h]: 0.00
  f1 in 2-D, instance 6: FEs=284, fbest-ftarget=-7.0829e-09, elapsed time [h]: 0.00
  f1 in 2-D, instance 7: FEs=278, fbest-ftarget=-6.5999e-09, elapsed time [h]: 0.00
  f1 in 2-D, instance 8: FEs=272, fbest-ftarget=-8.7044e-09, elapsed time [h]: 0.00
  f1 in 2-D, instance 9: FEs=248, fbest-ftarget=-2.6316e-09, elapsed time [h]: 0.00
  f1 in 2-D, instance 10: FEs=302, fbest-ftarget=-4.6779e-09, elapsed time [h]: 0.00
  f1 in 2-D, instance 11: FEs=272, fbest-ftarget=-5.1499e-09, elapsed time [h]: 0.00
  f1 in 2-D, instance 12: FEs=260, fbest-ftarget=-8.8635e-09, elapsed time [h]: 0.00
  f1 in 2-D, instance 13: FEs=266, fbest-ftarget=-2.5484e-09, elapsed time [h]: 0.00
  f1 in 2-D, instance 14: FEs=218, fbest-ftarget=-9.9961e-09, elapsed time [h]: 0.00
  f1 in 2-D, instance 15: FEs=248, fbest-ftarget=-7.5842e-09, elapsed time [h]: 0.00
      date and time: 2013 3 29 19 59 26
  f2 in 2-D, instance 1: FEs=824, fbest-ftarget=-7.0206e-09, elapsed time [h]: 0.00
  f2 in 2-D, instance 2: FEs=572, fbest-ftarget=-9.2822e-09, elapsed time [h]: 0.00
[...]
```

BBOB in practice

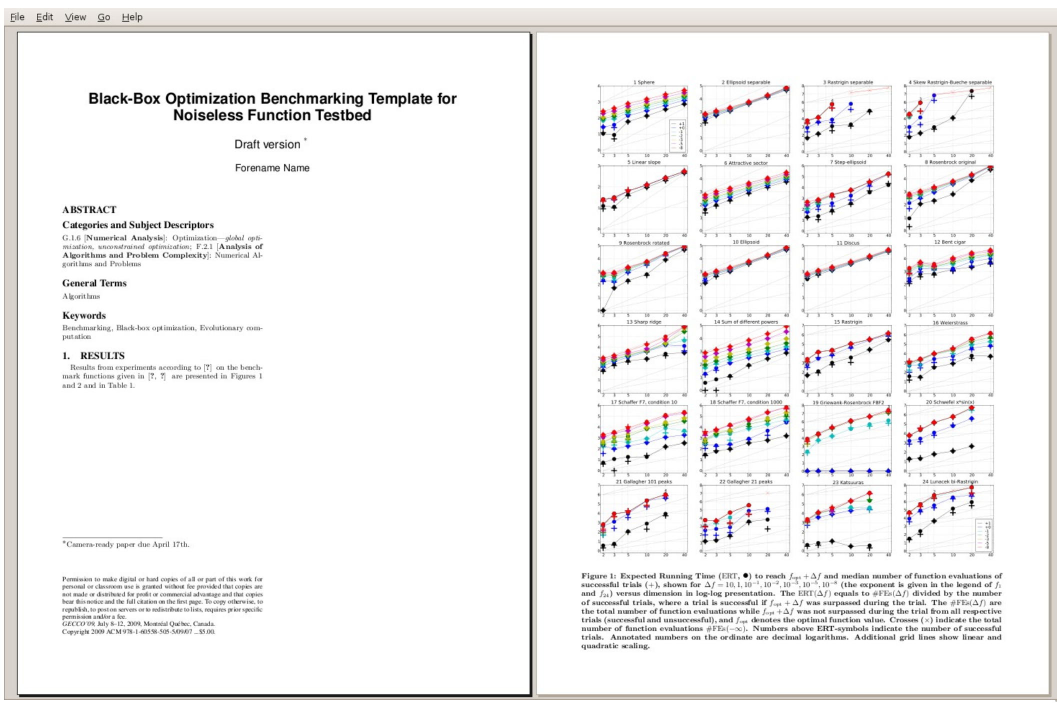
Name	Date Modified	Size	Kind
▼ bbob.v13.05	March 8, 2013 13:04	--	Folder
▶ c	March 5, 2013 23:04	--	Folder
▶ docs	March 6, 2013 13:56	--	Folder
▶ java	March 5, 2013 23:04	--	Folder
▶ latextemplates	March 5, 2013 23:04	--	Folder
▶ matlab	March 5, 2013 23:02	--	Folder
▼ python	Today, 20:08	--	Folder
▶ bbob_pproc	March 5, 2013 23:03	--	Folder
bbobbenchmarks.py	November 12, 2012 16:56	74 KB	Python script
benchmarkinfos.txt	February 9, 2009 16:29	4 KB	Gedit ...ument
exampleexperiment.py	February 22, 2013 14:26	4 KB	Python script
exampletiming.py	November 12, 2012 16:56	4 KB	Python script
fgeneric.py	March 3, 2013 19:33	25 KB	Python script
LICENSE.txt	February 1, 2012 20:23	4 KB	Gedit ...ument
README.txt	November 12, 2012 16:56	4 KB	Gedit ...ument
▶ r	March 5, 2013 23:05	--	Folder

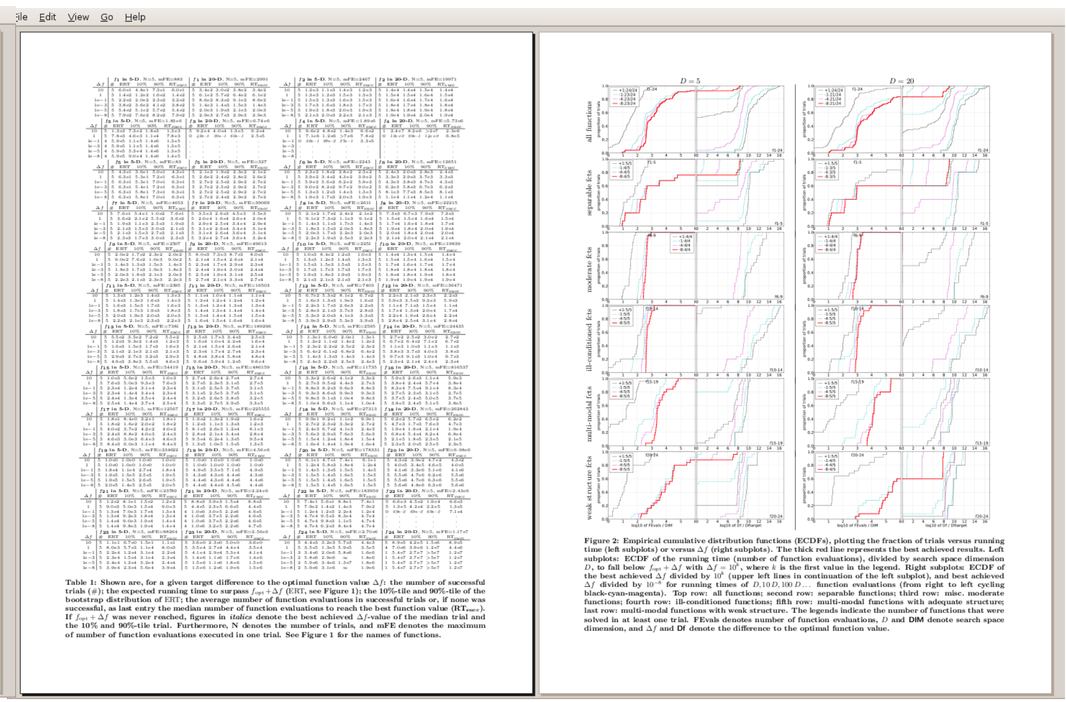
Macintosh HD ▸ Users ▸ hansen ▸ Downloads ▸ bbob.v13.05 ▸ python ▸ bbob_pproc

BBOB in practice

Post-processing at the OS shell:

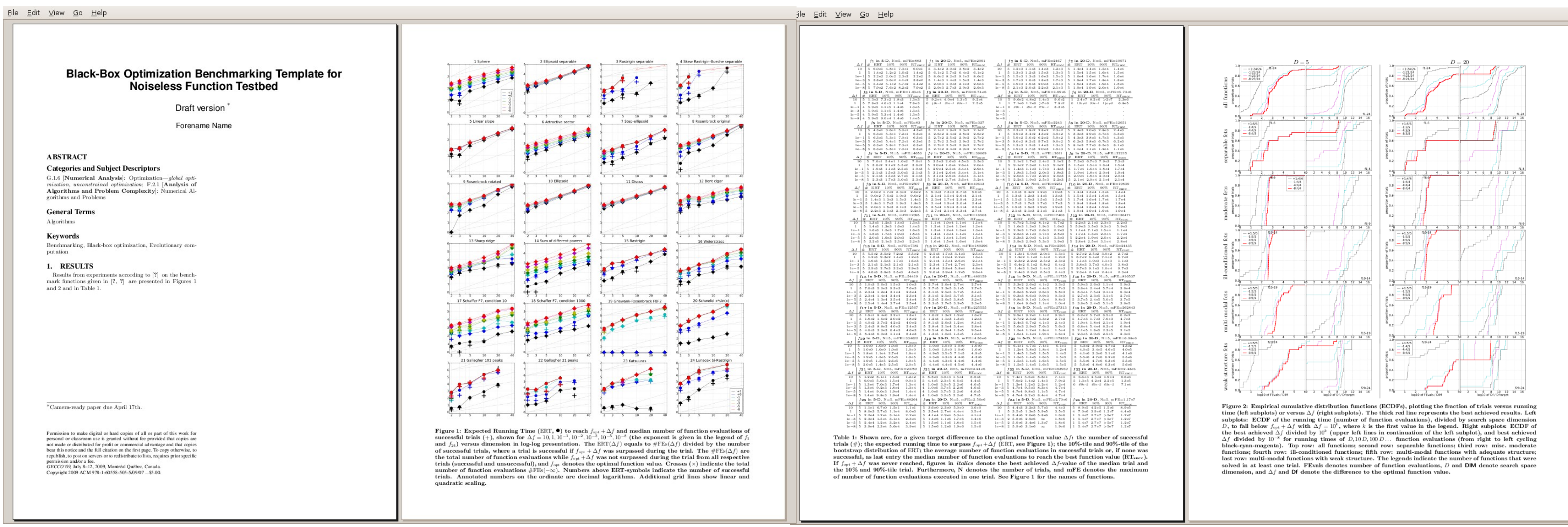
```
$ python codepath/bbob_pproc/rungeneric.py datapath
[...]
$ pdflatex templateACMarticle.tex
[...]
```





Post-processing at the OS shell:

```
$ python codepath/bbob_pproc/rungeneric.py datapath
[...]
$ pdflatex templateACMarticle.tex
[...]
```



Black-Box Optimization Benchmarking Template for Noiseless Function Testbed

Draft version *

Forename Name

ABSTRACT

Categories and Subject Descriptors

G.1.6 [**Numerical Analysis**]: Optimization—*global optimization, unconstrained optimization*; F.2.1 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems

General Terms

Algorithms

Keywords

Benchmarking, Black-box optimization, Evolutionary computation

1. RESULTS

Results from experiments according to [?] on the benchmark functions given in [?, ?] are presented in Figures 1 and 2 and in Table 1.

*Camera-ready paper due April 17th.

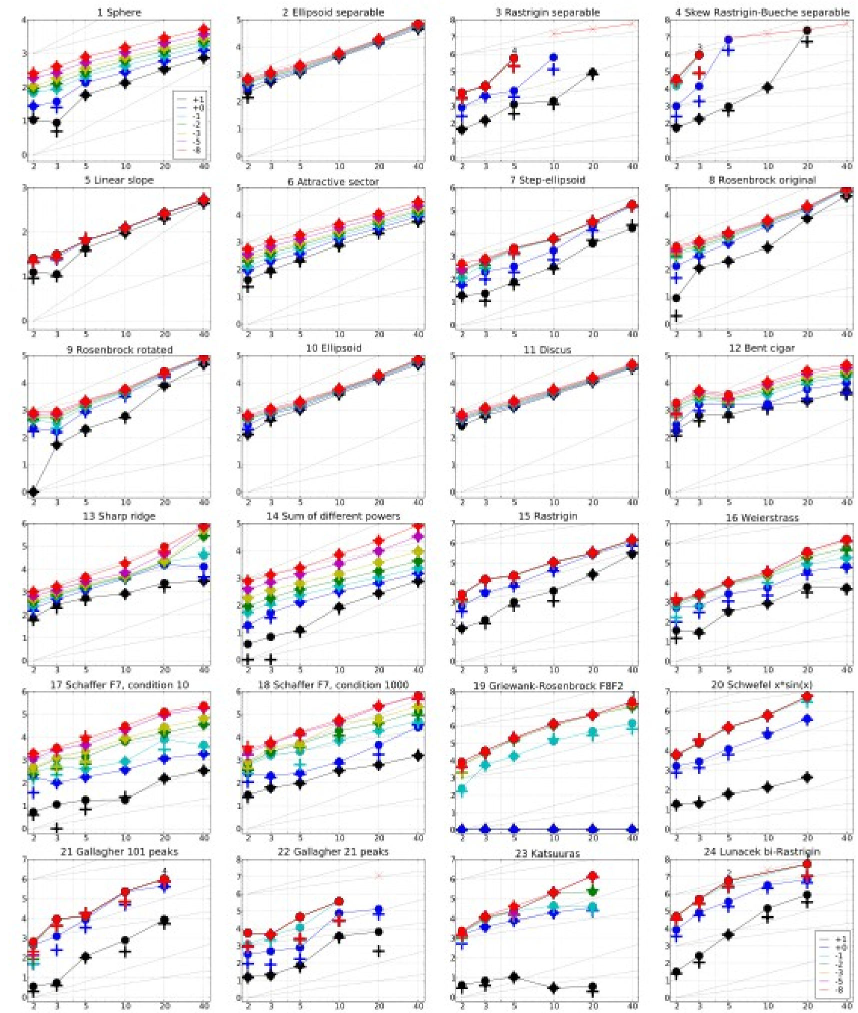


Figure 1: Expected Running Time (ERT, ●) to reach $f_{\rm opt}+\Delta f$ and median number of function evaluations of successful trials (+), shown for $\Delta f = 10, 1, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-5}, 10^{-8}$ (the exponent is given in the legend of f_1 and f_{24}) versus dimension in log-log presentation. The ERT(Δf) equals to #FEs(Δf) divided by the number of successful trials, where a trial is successful if $f_{\rm opt}+\Delta f$ was surpassed during the trial. The #FEs(Δf) are the total number of function evaluations while $f_{\rm opt}+\Delta f$ was not surpassed during the trial from all respective trials (successful and unsuccessful), and $f_{\rm opt}$ denotes the optimal function value. Crosses (×) indicate the total number of function evaluations #FEs($-\infty$). Numbers above ERT-symbols indicate the number of successful trials. Annotated numbers on the ordinate are decimal logarithms. Additional grid lines show linear and quadratic scaling.

Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
	f_1 in 5-D, N=5, mFE=883					**f_1 in 20-D, N=5, mFE=2991**				
10	5	6.0e1	4.8e1	7.3e1	6.0e1	5	3.4e2	3.0e2	3.8e2	3.4e2
1	5	1.4e2	1.2e2	1.6e2	1.4e2	5	6.1e2	5.7e2	6.4e2	6.1e2
1e−1	5	2.2e2	2.0e2	2.3e2	2.2e2	5	8.6e2	8.2e2	9.1e2	8.6e2
1e−3	5	3.8e2	3.6e2	4.1e2	3.8e2	5	1.4e3	1.4e3	1.5e3	1.4e3
1e−5	5	5.4e2	5.1e2	5.7e2	5.4e2	5	2.0e3	1.9e3	2.1e3	2.0e3
1e−8	5	7.9e2	7.6e2	8.2e2	7.9e2	5	2.9e3	2.7e3	2.9e3	2.9e3

Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
	f_2 in 5-D, N=5, mFE=2467					**f_2 in 20-D, N=5, mFE=19971**				
10	5	1.2e3	1.1e3	1.4e3	1.2e3	5	1.4e4	1.4e4	1.5e4	1.4e4
1	5	1.3e3	1.2e3	1.5e3	1.3e3	5	1.5e4	1.5e4	1.6e4	1.5e4
1e−1	5	1.5e3	1.3e3	1.6e3	1.5e3	5	1.6e4	1.6e4	1.7e4	1.6e4
1e−3	5	1.7e3	1.6e3	1.8e3	1.7e3	5	1.8e4	1.7e4	1.8e4	1.8e4
1e−5	5	1.9e3	1.8e3	2.0e3	1.9e3	5	1.8e4	1.8e4	1.9e4	1.8e4
1e−8	5	2.1e3	2.0e3	2.2e3	2.1e3	5	1.9e4	1.9e4	2.0e4	1.9e4

Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
	f_3 in 5-D, N=5, mFE=1.83e6					**f_3 in 20-D, N=5, mFE=6.74e6**				
10	5	1.3e3	7.3e2	1.8e3	1.3e3	5	9.2e4	4.0e4	1.3e5	9.2e4
1	5	7.8e3	4.6e3	1.1e4	7.8e3	0	*40e−1*	*30e−1*	*60e−1*	2.5e5
1e−1	4	5.9e5	1.1e5	1.4e6	1.3e5	.	.	.	.	.
1e−3	4	5.9e5	1.1e5	1.4e6	1.3e5	.	.	.	.	.
1e−5	4	5.9e5	5.3e4	1.4e6	1.3e5	.	.	.	.	.
1e−8	4	5.9e5	9.0e4	1.4e6	1.4e5	.	.	.	.	.

Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
	f_4 in 5-D, N=5, mFE=1.89e6					**f_4 in 20-D, N=5, mFE=5.73e6**				
10	5	9.6e2	4.8e2	1.4e3	9.6e2	1	2.4e7	8.2e6	>2e7	2.3e6
1	1	7.1e6	1.2e6	>7e6	7.8e2	0	*12e+0*	*99e−1*	*14e+0*	6.8e5
1e−1	0	*20e−1*	*99e−2*	*27e−1*	3.3e5	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
	f_5 in 5-D, N=5, mFE=83					**f_5 in 20-D, N=5, mFE=327**				
10	5	4.3e1	3.6e1	5.0e1	4.3e1	5	2.1e2	1.9e2	2.3e2	2.1e2
1	5	6.3e1	5.3e1	7.2e1	6.3e1	5	2.6e2	2.4e2	2.8e2	2.6e2
1e−1	5	6.3e1	5.3e1	7.0e1	6.3e1	5	2.7e2	2.5e2	2.9e2	2.7e2
1e−3	5	6.3e1	5.4e1	7.2e1	6.3e1	5	2.7e2	2.5e2	2.9e2	2.7e2
1e−5	5	6.3e1	5.8e1	7.3e1	6.3e1	5	2.7e2	2.5e2	2.9e2	2.7e2
1e−8	5	6.3e1	5.8e1	7.0e1	6.3e1	5	2.7e2	2.4e2	2.9e2	2.7e2

Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
	f_6 in 5-D, N=5, mFE=2243					**f_6 in 20-D, N=5, mFE=12651**				
10	5	2.3e2	1.8e2	2.8e2	2.3e2	5	2.4e3	2.0e3	2.8e3	2.4e3
1	5	3.9e2	3.4e2	4.3e2	3.9e2	5	3.3e3	2.9e3	3.7e3	3.3e3
1e−1	5	5.9e2	5.6e2	6.2e2	5.9e2	5	4.3e3	3.8e3	4.7e3	4.3e3
1e−3	5	9.0e2	8.2e2	9.7e2	9.0e2	5	6.2e3	5.8e3	6.7e3	6.2e3
1e−5	5	1.3e3	1.2e3	1.4e3	1.3e3	5	8.1e3	7.7e3	8.5e3	8.1e3
1e−8	5	1.9e3	1.7e3	2.0e3	1.9e3	5	1.1e4	1.1e4	1.2e4	1.1e4

Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
	f_7 in 5-D, N=5, mFE=4653					**f_7 in 20-D, N=5, mFE=39069**				
10	5	7.6e1	5.4e1	1.0e2	7.6e1	5	3.5e3	2.6e3	4.5e3	3.5e3
1	5	3.6e2	2.1e2	5.5e2	3.6e2	5	2.0e4	1.6e4	2.6e4	2.0e4
1e−1	5	1.9e3	1.1e3	2.5e3	1.9e3	5	2.9e4	2.5e4	3.4e4	2.9e4
1e−3	5	2.1e3	1.5e3	3.0e3	2.1e3	5	3.1e4	2.6e4	3.4e4	3.1e4
1e−5	5	2.1e3	1.5e3	2.7e3	2.1e3	5	3.1e4	2.6e4	3.6e4	3.1e4
1e−8	5	2.3e3	1.7e3	3.0e3	2.3e3	5	3.2e4	2.7e4	3.6e4	3.2e4

Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
	f_8 in 5-D, N=5, mFE=2611					**f_8 in 20-D, N=5, mFE=22215**				
10	5	2.1e2	1.7e2	2.4e2	2.1e2	5	7.3e3	6.7e3	7.9e3	7.3e3
1	5	9.1e2	7.3e2	1.1e3	9.1e2	5	1.5e4	1.5e4	1.6e4	1.5e4
1e−1	5	1.4e3	1.1e3	1.7e3	1.4e3	5	1.7e4	1.6e4	1.8e4	1.7e4
1e−3	5	1.8e3	1.5e3	2.0e3	1.8e3	5	1.9e4	1.8e4	2.0e4	1.9e4
1e−5	5	2.0e3	1.7e3	2.2e3	2.0e3	5	2.0e4	1.8e4	2.0e4	2.0e4
1e−8	5	2.2e3	1.9e3	2.5e3	2.2e3	5	2.1e4	2.0e4	2.1e4	2.1e4

Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
	f_9 in 5-D, N=5, mFE=2507					**f_9 in 20-D, N=5, mFE=49613**				
10	5	2.0e2	1.7e2	2.3e2	2.0e2	5	8.0e3	7.3e3	8.7e3	8.0e3
1	5	9.0e2	7.6e2	1.0e3	9.0e2	5	2.1e4	1.5e4	2.6e4	2.1e4
1e−1	5	1.4e3	1.3e3	1.5e3	1.4e3	5	2.3e4	1.7e4	2.9e4	2.3e4
1e−3	5	1.8e3	1.7e3	1.9e3	1.8e3	5	2.4e4	1.9e4	3.0e4	2.4e4
1e−5	5	2.0e3	1.8e3	2.1e3	2.0e3	5	2.5e4	1.9e4	3.1e4	2.5e4
1e−8	5	2.2e3	2.1e3	2.3e3	2.2e3	5	2.7e4	2.1e4	3.3e4	2.7e4

Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
	f_{10} in 5-D, N=5, mFE=2251					**f_{10} in 20-D, N=5, mFE=19839**				
10	5	1.0e3	8.4e2	1.2e3	1.0e3	5	1.4e4	1.3e4	1.5e4	1.4e4
1	5	1.3e3	1.2e3	1.4e3	1.3e3	5	1.5e4	1.5e4	1.6e4	1.5e4
1e−1	5	1.5e3	1.5e3	1.5e3	1.5e3	5	1.7e4	1.6e4	1.7e4	1.7e4
1e−3	5	1.7e3	1.7e3	1.7e3	1.7e3	5	1.8e4	1.8e4	1.8e4	1.8e4
1e−5	5	1.9e3	1.8e3	1.9e3	1.9e3	5	1.8e4	1.8e4	1.9e4	1.8e4
1e−8	5	2.1e3	2.1e3	2.1e3	2.1e3	5	1.9e4	1.9e4	1.9e4	1.9e4

Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
	f_{11} in 5-D, N=5, mFE=2395					**f_{11} in 20-D, N=5, mFE=16503**				
10	5	1.3e3	1.2e3	1.4e3	1.3e3	5	1.1e4	1.0e4	1.1e4	1.1e4
1	5	1.4e3	1.3e3	1.6e3	1.4e3	5	1.2e4	1.2e4	1.2e4	1.2e4
1e−1	5	1.6e3	1.5e3	1.7e3	1.6e3	5	1.3e4	1.2e4	1.3e4	1.3e4
1e−3	5	1.8e3	1.7e3	1.9e3	1.8e3	5	1.4e4	1.3e4	1.4e4	1.4e4
1e−5	5	2.0e3	1.9e3	2.0e3	2.0e3	5	1.5e4	1.4e4	1.5e4	1.5e4
1e−8	5	2.2e3	2.1e3	2.3e3	2.2e3	5	1.6e4	1.5e4	1.6e4	1.6e4

Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
	f_{12} in 5-D, N=5, mFE=7403					**f_{12} in 20-D, N=5, mFE=36471**				
10	5	6.7e2	5.3e2	8.1e2	6.7e2	5	2.2e3	2.1e3	2.3e3	2.2e3
1	5	1.6e3	1.3e3	1.9e3	1.6e3	5	5.9e3	3.5e3	9.3e3	5.9e3
1e−1	5	2.2e3	1.7e3	2.6e3	2.2e3	5	1.1e4	7.1e3	1.5e4	1.1e4
1e−3	5	2.8e3	2.1e3	3.7e3	2.8e3	5	1.7e4	1.3e4	2.0e4	1.7e4
1e−5	5	3.3e3	2.0e3	4.1e3	3.3e3	5	2.2e4	1.9e4	2.6e4	2.2e4
1e−8	5	3.9e3	2.9e3	5.3e3	3.9e3	5	2.8e4	2.5e4	3.1e4	2.8e4

Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
	f_{13} in 5-D, N=5, mFE=7381					**f_{13} in 20-D, N=5, mFE=189296**				
10	5	5.5e2	3.5e2	7.2e2	5.5e2	5	2.5e3	1.7e3	3.4e3	2.5e3
1	5	1.2e3	9.3e2	1.4e3	1.2e3	5	1.6e4	1.0e4	2.2e4	1.6e4
1e−1	5	1.6e3	1.5e3	1.7e3	1.6e3	5	2.1e4	1.5e4	2.6e4	2.1e4
1e−3	5	2.1e3	2.1e3	2.1e3	2.1e3	5	2.3e4	1.7e4	2.7e4	2.3e4
1e−5	5	2.9e3	2.7e3	3.2e3	2.9e3	5	4.8e4	3.8e4	5.8e4	4.8e4
1e−8	5	4.6e3	3.8e3	5.5e3	4.6e3	5	9.6e4	5.9e4	1.2e5	9.6e4

Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
	f_{14} in 5-D, N=5, mFE=2595					**f_{14} in 20-D, N=5, mFE=24435**				
10	5	1.3e1	6.0e0	2.0e1	1.3e1	5	2.7e2	2.5e2	3.0e2	2.7e2
1	5	1.2e2	1.1e2	1.4e2	1.2e2	5	6.7e2	6.4e2	7.1e2	6.7e2
1e−1	5	2.3e2	2.2e2	2.5e2	2.3e2	5	1.1e3	1.0e3	1.1e3	1.1e3
1e−3	5	6.4e2	6.1e2	6.8e2	6.4e2	5	3.8e3	3.7e3	4.0e3	3.8e3
1e−5	5	1.4e3	1.3e3	1.4e3	1.4e3	5	9.7e3	9.1e3	1.0e4	9.7e3
1e−8	5	2.4e3	2.2e3	2.5e3	2.4e3	5	2.3e4	2.1e4	2.4e4	2.3e4

Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
	f_{15} in 5-D, N=5, mFE=54419					**f_{15} in 20-D, N=5, mFE=486159**				
10	5	1.0e3	5.6e2	1.5e3	1.0e3	5	2.7e4	2.6e4	2.7e4	2.7e4
1	5	7.6e3	5.0e3	9.3e3	7.6e3	5	2.7e5	2.3e5	3.1e5	2.7e5
1e−1	5	2.3e4	1.2e4	3.1e4	2.3e4	5	3.1e5	2.5e5	3.7e5	3.1e5
1e−3	5	2.3e4	1.4e4	3.4e4	2.3e4	5	3.1e5	2.5e5	3.7e5	3.1e5
1e−5	5	2.4e4	1.3e4	3.5e4	2.4e4	5	3.2e5	2.6e5	3.8e5	3.2e5
1e−8	5	2.5e4	1.4e4	3.7e4	2.5e4	5	3.3e5	2.7e5	3.9e5	3.3e5

Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
	f_{16} in 5-D, N=5, mFE=11735					**f_{16} in 20-D, N=5, mFE=816537**				
10	5	3.3e2	2.6e2	4.1e2	3.3e2	5	5.9e3	2.6e3	1.1e4	5.9e3
1	5	2.7e3	9.5e2	4.4e3	2.7e3	5	3.8e4	2.4e4	5.7e4	3.8e4
1e−1	5	8.8e3	8.2e3	9.6e3	8.8e3	5	8.3e4	7.5e4	9.1e4	8.3e4
1e−3	5	9.3e3	8.6e3	9.9e3	9.3e3	5	2.7e5	2.3e5	3.1e5	2.7e5
1e−5	5	9.8e3	9.1e3	1.0e4	9.8e3	5	3.7e5	2.4e5	5.0e5	3.7e5
1e−8	5	1.0e4	9.6e3	1.1e4	1.0e4	5	3.8e5	2.4e5	5.1e5	3.8e5

Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
	f_{17} in 5-D, N=5, mFE=12567					**f_{17} in 20-D, N=5, mFE=225555**				
10	5	1.8e1	8.4e0	3.2e1	1.8e1	5	1.6e2	1.3e2	1.9e2	1.6e2
1	5	1.8e2	1.6e2	2.0e2	1.8e2	5	1.2e3	1.1e3	1.3e3	1.2e3
1e−1	5	4.0e2	3.7e2	4.2e2	4.0e2	5	8.1e3	2.6e3	1.2e4	8.1e3
1e−3	5	2.4e3	8.8e2	4.0e3	2.4e3	5	2.8e4	2.1e4	3.4e4	2.8e4
1e−5	5	4.6e3	3.0e3	6.4e3	4.6e3	5	9.5e4	6.2e4	1.3e5	9.5e4
1e−8	5	8.4e3	6.0e3	1.1e4	8.4e3	5	1.3e5	1.0e5	1.5e5	1.3e5

Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
	f_{18} in 5-D, N=5, mFE=27313					**f_{18} in 20-D, N=5, mFE=262843**				
10	5	9.9e1	9.2e1	1.1e2	9.9e1	5	6.2e2	5.7e2	6.5e2	6.2e2
1	5	2.7e2	2.3e2	3.3e2	2.7e2	5	4.7e3	1.7e3	7.6e3	4.7e3
1e−1	5	2.4e3	6.7e2	4.1e3	2.4e3	5	1.9e4	1.8e4	2.1e4	1.9e4
1e−3	5	5.6e3	2.9e3	7.6e3	5.6e3	5	6.8e4	5.4e4	8.2e4	6.8e4
1e−5	5	1.5e4	1.2e4	1.8e4	1.5e4	5	2.1e5	1.8e5	2.3e5	2.1e5
1e−8	5	1.6e4	1.4e4	1.9e4	1.6e4	5	2.3e5	2.0e5	2.5e5	2.3e5

Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
	f_{19} in 5-D, N=5, mFE=334622					**f_{19} in 20-D, N=5, mFE=4.56e6**				
10	5	1.0e0	1.0e0	1.0e0	1.0e0	5	1.0e0	1.0e0	1.0e0	1.0e0
1	5	1.0e0	1.0e0	1.0e0	1.0e0	5	1.0e0	1.0e0	1.0e0	1.0e0
1e−1	5	1.8e4	1.1e4	2.7e4	1.8e4	5	4.9e5	3.5e5	7.1e5	4.9e5
1e−3	5	1.9e5	1.5e5	2.5e5	1.9e5	5	4.3e6	4.3e6	4.4e6	4.3e6
1e−5	5	1.9e5	1.5e5	2.6e5	1.9e5	5	4.4e6	4.3e6	4.4e6	4.4e6
1e−8	5	2.0e5	1.4e5	2.5e5	2.0e5	5	4.4e6	4.4e6	4.5e6	4.4e6

Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
	f_{20} in 5-D, N=5, mFE=178331					**f_{20} in 20-D, N=5, mFE=6.98e6**				
10	5	6.1e1	4.7e1	7.4e1	6.1e1	5	4.3e2	3.9e2	4.7e2	4.3e2
1	5	1.2e4	5.8e3	1.8e4	1.2e4	5	4.0e5	3.4e5	4.6e5	4.0e5
1e−1	5	1.4e5	1.3e5	1.5e5	1.4e5	5	4.1e6	3.3e6	5.1e6	4.1e6
1e−3	5	1.5e5	1.4e5	1.6e5	1.5e5	5	5.5e6	4.7e6	6.2e6	5.5e6
1e−5	5	1.5e5	1.4e5	1.6e5	1.5e5	5	5.5e6	4.7e6	6.3e6	5.5e6
1e−8	5	1.5e5	1.4e5	1.6e5	1.5e5	5	5.6e6	4.8e6	6.3e6	5.6e6

Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
	f_{21} in 5-D, N=5, mFE=23789					**f_{21} in 20-D, N=5, mFE=2.24e6**				
10	5	1.2e2	8.1e1	1.5e2	1.2e2	5	8.8e3	3.9e3	1.5e4	8.8e3
1	5	9.0e3	5.0e3	1.5e4	9.0e3	5	4.4e5	2.3e5	6.6e5	4.4e5
1e−1	5	1.3e4	7.0e3	1.7e4	1.3e4	4	1.0e6	3.0e5	2.2e6	4.6e5
1e−3	5	1.3e4	9.2e3	1.8e4	1.3e4	4	1.0e6	3.7e5	2.2e6	4.6e5
1e−5	5	1.4e4	9.0e3	1.9e4	1.4e4	4	1.0e6	3.7e5	2.2e6	4.6e5
1e−8	5	1.4e4	9.8e3	1.9e4	1.4e4	4	1.0e6	3.2e5	2.2e6	4.7e5

Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
	f_{22} in 5-D, N=5, mFE=183959					**f_{22} in 20-D, N=5, mFE=2.43e6**				
10	5	7.4e1	5.6e1	8.8e1	7.4e1	5	6.6e3	4.5e2	1.9e4	6.6e3
1	5	7.9e2	1.4e2	1.4e3	7.9e2	5	1.3e5	4.2e4	2.2e5	1.3e5
1e−1	5	1.2e4	1.2e3	2.2e4	1.2e4	0	*69e−2*	*69e−2*	*69e−2*	7.1e4
1e−3	5	4.7e4	9.5e3	8.3e4	4.7e4	.	.	.	.	.
1e−5	5	4.7e4	9.8e3	1.1e5	4.7e4	.	.	.	.	.
1e−8	5	4.7e4	6.2e3	8.4e4	4.7e4	.	.	.	.	.

Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
	f_{23} in 5-D, N=5, mFE=88264					**f_{23} in 20-D, N=5, mFE=2.56e6**				
10	5	1.1e1	6.7e0	1.5e1	1.1e1	5	3.6e0	2.3e0	5.0e0	3.6e0
1	5	8.0e3	5.7e3	1.1e4	8.0e3	5	3.5e4	2.7e4	4.4e4	3.5e4
1e−1	5	2.2e4	1.2e4	3.1e4	2.2e4	5	4.1e4	2.9e4	5.3e4	4.1e4
1e−3	5	2.3e4	1.5e4	3.1e4	2.3e4	5	1.4e6	1.1e6	1.7e6	1.4e6
1e−5	5	2.4e4	1.2e4	3.2e4	2.4e4	5	1.5e6	1.1e6	1.8e6	1.5e6
1e−8	5	3.9e4	2.3e4	5.6e4	3.9e4	5	1.5e6	1.2e6	1.9e6	1.5e6

Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
	f_{24} in 5-D, N=5, mFE=2.70e6					**f_{24} in 20-D, N=5, mFE=1.17e7**				
10	5	4.4e3	3.2e3	5.7e3	4.4e3	5	8.9e5	4.2e5	1.5e6	8.9e5
1	5	3.5e5	1.3e5	5.9e5	3.5e5	4	7.0e6	3.9e6	1.2e7	4.4e6
1e−1	3	3.4e6	2.0e6	5.8e6	1.6e6	1	5.4e7	2.7e7	>5e7	1.2e7
1e−3	2	5.8e6	2.9e6	∞	1.8e6	1	5.4e7	2.7e7	>5e7	1.2e7
1e−5	2	5.9e6	3.4e6	1.3e7	1.8e6	1	5.4e7	2.7e7	>5e7	1.2e7
1e−8	2	5.9e6	3.1e6	∞	1.9e6	1	5.4e7	2.7e7	>5e7	1.2e7

Table 1: Shown are, for a given target difference to the optimal function value Δf: the number of successful trials (#); the expected running time to surpass $f_{opt}+\Delta f$ (ERT, see Figure 1); the 10%-tile and 90%-tile of the bootstrap distribution of ERT; the average number of function evaluations in successful trials or, if none was successful, as last entry the median number of function evaluations to reach the best function value (RT_{succ}). If $f_{opt}+\Delta f$ was never reached, figures in *italics* denote the best achieved Δf-value of the median trial and the 10% and 90%-tile trial. Furthermore, N denotes the number of trials, and mFE denotes the maximum of number of function evaluations executed in one trial. See Figure 1 for the names of functions.

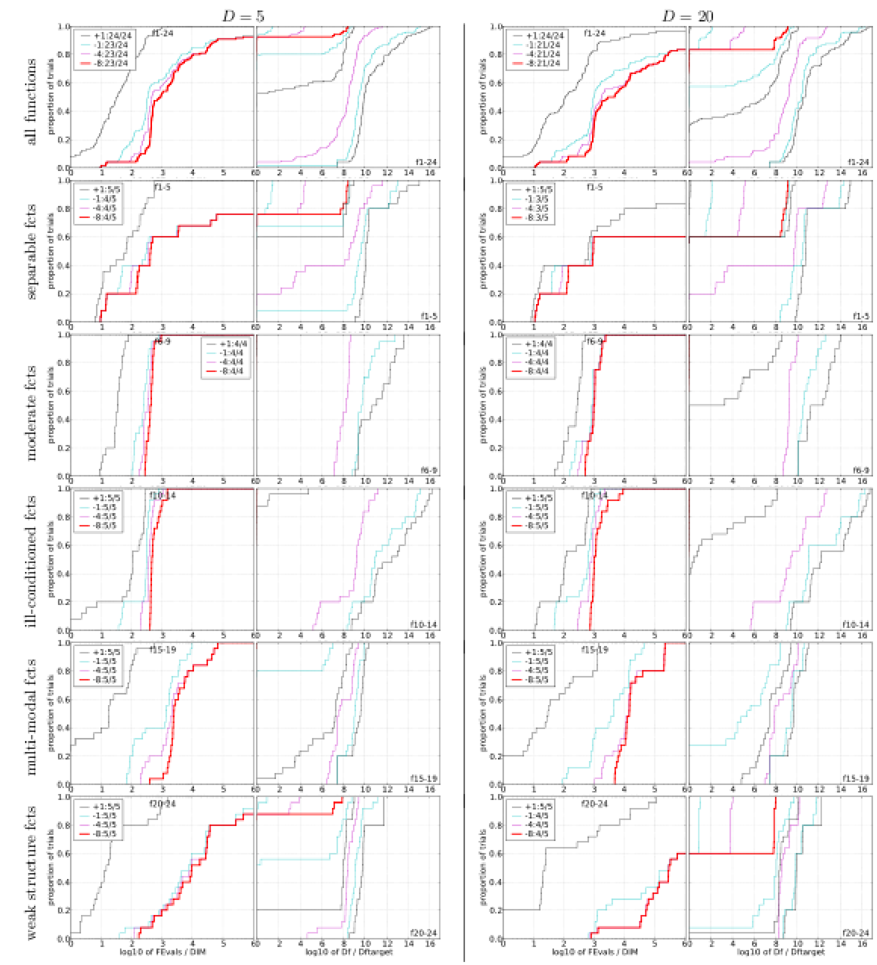


Figure 2: Empirical cumulative distribution functions (ECDFs), plotting the fraction of trials versus running time (left subplots) or versus Δf (right subplots). The thick red line represents the best achieved results. Left subplots: ECDF of the running time (number of function evaluations), divided by search space dimension D, to fall below $f_{opt}+\Delta f$ with $\Delta f = 10^k$, where k is the first value in the legend. Right subplots: ECDF of the best achieved Δf divided by 10^k (upper left lines in continuation of the left subplot), and best achieved Δf divided by 10^{-8} for running times of $D, 10\,D, 100\,D\ldots$ function evaluations (from right to left cycling black-cyan-magenta). Top row: all functions; second row: separable functions; third row: misc. moderate functions; fourth row: ill-conditioned functions; fifth row: multi-modal functions with adequate structure; last row: multi-modal functions with weak structure. The legends indicate the number of functions that were solved in at least one trial. FEvals denotes number of function evaluations, D and DIM denote search space dimension, and Δf and Df denote the difference to the optimal function value.

Test Functions

Test Functions

Test functions

- define the "scientific question"

the relevance can hardly be overestimated

- should represent "reality"
- are often too simple?

remind separability

- a number of testbeds are around

GECCO-BBOB Test Functions

Questions?