Benchmarking CMA-ES with Basic Integer Handling on a Mixed-Integer Test Problem Suite

Tristan Marty
Thales Research and Technology,
Inria and Institut Polytechnique

Yann Semet
Thales Research and Technology

Anne Auger
Inria and Institut Polytechnique

Sébastion Héron
Thales Research and Technology

Nikolaus Hansen
Inria and Institut Polytechnique

ABSTRACT

We compare the performances of one implementation of CMA-ES (pycma version 3.3.0) for optimizing functions with both continuous and integer variables. The implementation incorporates a lower bound on the variance along the integer coordinates to keep the optimization from stalling. This benchmark will serve as a baseline for further works on pycma. Results show substantial improvement since the last benchmarked version of pycma. Also this implementation is competitive with other mixed integer algorithms.

CCS CONCEPTS

•**Computing methodologies** → **Continuous space search;**

KEYWORDS

Benchmarking, Black-box optimization, Mixed-integer optimization

ACM Reference format:
Tristan Marty, Yann Semet, Anne Auger, Sébastion Héron, and Nikolaus Hansen. 2023. Benchmarking CMA-ES with Basic Integer Handling on a Mixed-Integer Test Problem Suite . In *Proceedings of The Genetic and Evolutionary Computation Conference 2023, Lisbon, Portugal, July 15–19, 2023 (GECCO '23),* 8 pages.
DOI: 10.1145/3583133.3596411

1 INTRODUCTION

Mixed optimization problems are of great importance in real-world applications. Typically, variables include real-valued degrees of freedom that correspond, directly or not, to physical measurements (e.g. angles, frequencies, forces, etc.) as well as discrete (integer or binary) settings that correspond to categorical choices (e.g. materials) or service activations (e.g. radar modes). Therefore, developing algorithmic variants that can handle several types of optimization variable at the same time without impairing the search process will be very useful in practice.

In the context of black box optimization for problems with continuous variables, evolutionary algorithms which update a covariance matrix, namely CMA-ES, havebeen studied for more than 20 years [10]. While the CMA-ES was originally not designed to handle integer variables, it can be employed with integer variables by using the integer part of a continuous variable in the fitness evaluation. This is however expected to lead to integer variables getting stuck in the resulting fitness plateaus, and several versions have been proposed to improve the algorithm in this situation [4, 5].

In order to study mixed integer problems within the COCO framework [9], the `bbob-mixint` testbed was proposed in [13]. This testbed is based on the 24 functions of the `bbob` testbed with 80% of the variables to be integer with an arity between 2 and 16.

In order to prevent variables getting stuck on the same integer level, for example in CMA-ES, a lower bound on the variance can be set such that there remains a reasonable probability for an integer variable to "jump" to another integer level, as for example in CMA-ES with margin (CMA-ESwM) [4]. The same principle is implemented in the pycma code with Integer Handling (`cma-IH` in the benchmark) with a more simplistic approach.

The purpose of this paper is to measure the performance of the current implementation of the widely distributed `cma-IH` code (version 3.3.0) [6] with different options and compare its performance with previous results and enhancements.

2 ALGORITHM PRESENTATION

The general idea behind `cma-IH` is to set a lower bound for the variance of the integer variables. It prevents integer variables from being trapped in one value. We formulate the bounding using the covariance matrix decomposition from the CMA-ES with diagonal decoding [1]. At each new iteration $t+1$, new candidate solutions, $x^{(t+1)}$, are sampled according to a multivariate normal distribution.

$$\begin{aligned} x^{(t+1)} &\sim \mathcal{N}(m,\, {\sigma^{(t)}}^2\, \mathbf{C}^{(t)}) \\ &\sim m + \sigma^{(t)}\, \sqrt{\mathbf{C}^{(t)}}\mathcal{N}(0, \mathbf{I}) \ , \end{aligned}$$

where $\mathbf{C}$ is a covariance matrix and σ is the step size. In order to bound the standard deviation of each integer variable, we introduce the diagonal scaling matrix $\mathbf{D}$,

$$x^{(t+1)} \sim m + \sigma^{(t)}\, \mathbf{D}^{(t)} \sqrt{\mathbf{C}^{(t)}}\mathcal{N}(0, \mathbf{I}) \ ,$$

where $\mathbf{D}^{(0)}$ is the identity. The update of σ and $\mathbf{C}$ are done as in the original CMA-ES algorithm. Then, the diagonal matrix $\mathbf{D}$ is updated based on the current standard deviation in each coordinate. The standard deviation of the i^{th}-th coordinate of x, $\sigma^{(t)}_{\text{std}(i)}$, is computed with the diagonal elements of $\mathbf{C}$ and $\mathbf{D}$:

GECCO '23, Lisbon, Portugal
 979-8-4007-0120-7/23/07…$15.00
DOI: 10.1145/3583133.3596411

```
import cocoex
import cma
suite = cocoex.Suite ('bbob-mixint', '', '')
for problem in suite :
   up_bound = list(problem.upper_bounds[
      :problem.number_of_integer_variables]) +
      [None]
   lo_bound = list(problem.lower_bounds[
      :problem.number_of_integer_variables]) +
      [None]
   cma.fmin2(problem , problem.initial_solution, 1,
             {'bounds': [lo_bound, up_bound]
              'CMA_stds ':(problem.upper_bounds
                           - problem.lower_bounds)/5,
              'integer_variables':list(range(
                 problem.number_of_integer_variables)),
             restarts =9)}
```

Figure 1: Calling code to benchmark `cma.fmin2`.

$$\sigma^{(t)}_{\mathrm{std}(i)} = \sigma^{(t)}\,\mathrm{D}^{(t)}_{i,i}\sqrt{\mathrm{C}^{(t)}_{i,i}}$$

This standard deviation has to be compared with a lower bound σ_{LB}. If the standard deviation in the i^{th} coordinate is smaller than the threshold value, $\sigma^{(t)}_{\mathrm{std}(i)} < \sigma_{LB}$, then the $\mathbf{D}$ matrix is scaled up in this coordinate.

$$\mathrm{D}^{(t+1)}_{i,i} \leftarrow \begin{cases} \frac{\sigma_{LB}}{\sigma^{(t)}\sqrt{\mathrm{C}^{(t)}_{i,i}}} & \text{if } 1 \le i \le N_{\mathrm{int}} \text{ and } \sigma^{(t)}_{\mathrm{std}(i)} \le \sigma_{LB} \\ \mathrm{D}^{(t)}_{i,i} & \text{otherwise} \end{cases} \tag{1}$$

where N_{int} is the number of integer variables. In the version 3.3.0 of pycma with integer handling (`cma-IH`), the lower bound is set to:

$$\sigma_{LB} = \frac{1}{2N_{\mathrm{int}} + 1} \tag{2}$$

3 EXPERIMENTAL PROCEDURE

Figure 1 shows the calling code and options used to run the benchmark for `cma-IH`. A maximum of nine restarts may be performed and each restart doubles the population size starting from the default (IPOP-CMA-ES) [2]. The initial step size is chosen to sample solutions mainly within the upper and the lower bound of the problem. The bounds are enforced on the integer variables only. The maximum number of function evaluations allowed per problem is set to 10^4 times the number of dimension of the problem.

This current implementation `cma-IH` is compared to performance data from the `bbob-mixint` COCO archive[1]. We show results of the algorithms given in Table 1.

The experiments were performed and the plots were produced with COCO [9], version 2.6.3.

4 CPU TIMING

In order to evaluate the CPU timing of the algorithm, we have run the `cma-IH` algorithm with restarts on the `bbob-mixint` test suite [13] with a maximum budget of $10^4 \times$ dimensions function evaluations according to [11].

Table 1: Compared algorithms

`cma-IH`	pycma version 3.3 (current) with the default integer handling option
`CMA-ES-pycma`	previous integer handling of pycma benchmarked in [13]
`CMA-ESwM`	CMA-ES with margin benchmarked in [3]
`cma-NoIH`	pycma version 3.3 (current) without any integer handling option
`cmaIH1e-1`	as `cma-IH` with a larger coordinate-wise lower bound on the sample standard deviation of $\sigma_{LB} = 10^{-1}$
`DE-scipy`	Differential Evolution implemented in scipy benchmarked in [13]
`RANDOMSEARCH`	Random Search algorithm for baseline

The Python code was run on a Linux machine with 32 cores, Intel® Xeon® E7 to T3 v4 processor. The time per function evaluation averaged over all functions in the test bed for dimensions 5, 10, 20, 40, 80 and 160 equals 3.7, 4.9, 5.1, 5.6, 6.4 and 8.4 $\times 10^{-4}$ seconds respectively.

5 RESULTS

We show results from experiments according to [11] and [8] on the benchmark functions given in [13]. Figures 2 and 3 show summary results for the different function groups. Figure 4 shows the empirical distribution of function evaluations to reach 51 target values on the single functions [9]. Tables 2 and 3 present expected runtimes and statistical tests.[2]

The **expected runtime (ERT)**, used in the tables, depends on a given target precision, $I_{\mathrm{target}} = f_{\mathrm{opt}} + \Delta f$, and is computed over all relevant trials as the number of function evaluations executed during each trial while the best function value did not reach f_t, summed over all trials and divided by the number of trials that actually reached f_t [11, 12]. **Statistical significance** is tested with the rank-sum test for a given target Δf_t using, for each trial, either the number of needed function evaluations to reach f_t (inverted and multiplied by -1), or, if the target was not reached, the best Δf-value achieved, measured only up to the smallest number of overall function evaluations for any unsuccessful trial under consideration.

Comparing two integer handling versions of `pycma`. We compare the performance of `cma-IH` (the current default implementation) versus `CMA-ES-pycma` (an older implementation). In comparison, the `cma-IH` performs increasingly better with increasing dimension and is overall about ten times faster than `CMA-ES-pycma` in dimension 80 (see Figure 3, lower right). The largest differences we observe on the ill-conditioned and multi-modal functions. The difference on the linear slope function is mainly due to the boundary handling which was applied also to the continuous variables in the `CMA-ES-pycma` experiment, as we confirmed in additional

[1] See https://numbbo.github.io/data-archive

[2] The entire data can be found at https://trmarty.github.io/MixedIntegerData/

experiments (not shown). The cma-IH performs statistically significantly better on 6, 8, 10, 16 and 15 functions in dimension 5, 10, 20, 40 and 80, respectively. The CMA-ES-pycma performs better on the separable ellipsoid function f2 in dimension 20, 40 and 80.

Comparing pycma with and without the integer handling. The cma-IH and cma-NoIH perform very similar overall. The only statistically significant difference we observed is on the separable ellipsoid function f2 in dimension 5 (not shown), where cma-IH is about three times faster than cma-NoIH, probably due to preventing additional restarts before to reach better targets. The default lower bound on the sample standard deviations for integer variables has overall little impact on the performance on this test suite.

A larger lower bound for the standard deviations. In order to understand the impact of the choice of the standard deviation lower bound value, we have run experiments with a few different bound values and show the results for the lower bound set to 10^{-1}. While in the smaller dimensions the effects are minor, in larger dimension the cmaIH1e-1 performs overall best and is three times faster than cma-IH on the ill-conditioned functions in larger dimension for budgets above 100 times dimension. On the 80-dimensional *sphere* function, cmaIH1e-1 solves the last target ten times quicker than cma-IH and also two times quicker than CMA-ESwM. The cmaIH1e-1 also solves the *Ellipsoid separable* whereas cma-IH gets stuck at around 20% of targets. The cmaIH1e-1 is slower than CMA-ESwM by a factor of approximately 3 on functions 5, 13 and 14, however it is significantly better than the current version cma-IH on functions 2, 5 or 12 while not being worse on all other functions. For this range of dimensions, bounding the standard deviation to 10^{-1} is a simple yet effective solution.

6 CONCLUSION

We have assessed the integer handling implemented in the current pycma module [7] (version 3.3.0). The performance with integer handling is overall superior to a previously benchmarked version, however it behaves quite similar to the version without integer handling. We attribute the latter observation to a suboptimal parameter setting: tests with a lower bound for the sample standard deviations of 1/10, which is considerably larger than the original setting, reveal a remarkably improved performance on a number of functions. In particular in larger dimension, this variant performs overall best.

REFERENCES

[1] Y. Akimoto and N. Hansen. 2020. Diagonal Acceleration for Covariance Matrix Adaptation Evolution Strategies. *Evolutionary Computation* 28, 3 (09 2020), 405–435. DOI: http://dx.doi.org/10.1162/evco_a_00260

[2] A. Auger and N. Hansen. 2005. A restart CMA evolution strategy with increasing population size. In *2005 IEEE Congress on Evolutionary Computation*, Vol. 2. 1769–1776 Vol. 2. DOI: http://dx.doi.org/10.1109/CEC.2005.1554902

[3] Ryoki Hamano, Shota Saito, Masahiro Nomura, and Shinichi Shirakawa. 2022. Benchmarking CMA-ES with Margin on the Bbob-Mixint Testbed. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '22)*. Association for Computing Machinery, New York, NY, USA, 1708–1716. DOI: http://dx.doi.org/10.1145/3520304.3534043

[4] Ryoki Hamano, Shota Saito, Masahiro Nomura, and Shinichi Shirakawa. 2022. CMA-ES with margin: lower-bounding marginal probability for mixed-integer black-box optimization. *Proceedings of the Genetic and Evolutionary Computation Conference* (2022).

[5] Nikolaus Hansen. 2011. *A CMA-ES for Mixed-Integer Nonlinear Optimization.* Research Report RR-7751. INRIA. https://hal.inria.fr/inria-00629689

[6] Nikolaus Hansen. 2022. CMA-ES/pycma 3.3.0. https://github.com/CMA-ES/pycma/tree/development. (2022). commit e762ac0.

[7] Nikolaus Hansen, Youhei Akimoto, and Petr Baudis. 2019. CMA-ES/pycma on Github. Zenodo, DOI:10.5281/zenodo.2559634. (Feb. 2019). DOI: http://dx.doi.org/10.5281/zenodo.2559634

[8] N. Hansen, A Auger, D. Brockhoff, D. Tušar, and T. Tušar. 2016. COCO: Performance Assessment. *ArXiv e-prints* arXiv:1605.03560 (2016).

[9] N. Hansen, A. Auger, R. Ros, O. Mersmann, T. Tušar, and D. Brockhoff. 2021. COCO: A Platform for Comparing Continuous Optimizers in a Black-Box Setting. *Optimization Methods and Software* 36 (2021), 114–144. Issue 1. DOI: http://dx.doi.org/10.1080/10556788.2020.1808977

[10] N. Hansen and A. Ostermeier. 1996. Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. In *Proceedings of IEEE International Conference on Evolutionary Computation*. 312–317. DOI: http://dx.doi.org/10.1109/ICEC.1996.542381

[11] N. Hansen, T. Tušar, O. Mersmann, A. Auger, and D. Brockhoff. 2016. COCO: The Experimental Procedure. *ArXiv e-prints* arXiv:1603.08776 (2016).

[12] Kenneth Price. 1997. Differential evolution vs. the functions of the second ICEO. In *Proceedings of the IEEE International Congress on Evolutionary Computation*. IEEE, Piscataway, NJ, USA, 153–157. DOI: http://dx.doi.org/10.1109/ICEC.1997.592287

[13] Tea Tušar, Dimo Brockhoff, and Nikolaus Hansen. 2019. Mixed-Integer Benchmark Problems for Single- and Bi-Objective Optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '19)*. Association for Computing Machinery, New York, NY, USA, 718–726. DOI: http://dx.doi.org/10.1145/3321707.3321868

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f1								
cma-IH	**818**(133)	2349(137)	7769(8446)	8453(8858)	9602(9016)	1.2e4(9556)	1.5e4(1e4)	15/15
CMA-ES-	1.5e4(2e4)	4.2e4(5e4)	4.2e4(6e4)	4.5e4(5e4)	5.3e4(5e4)	7.0e4(5e4)	8.7e4(5e4)	15/15
CMA-ESw	2239(849)	3993(2022)	4522(2228)	5079(2058)	5992(1916)	8350(1843)	1.1e4(1698)	15/15
DE-scip	8636(2819)	4.5e4(5327)	8.9e4(2e5)	9.2e4(2e5)	9.6e4(2e5)	1.0e5(2e5)	1.1e5(2e5)	13/15
cma-NoI	859(114)	**1405**(122)	8689(8511)	9428(8770)	1.1e4(8925)	1.4e4(9846)	1.7e4(1e4)	15/15
cmaIH1e	866(140)	1532(493)	**3957**(3556)	**4159**(3350)	**4477**(3065)	**5727**(2316)*	**6979**(1526)$^{\star 2}$	15/15
f2								
cma-IH	4.4e4(2e4)	2.4e5(7e4)	3.6e5(3e5)	5.6e5(3e5)	5.6e5(6e5)	7.2e5(5e5)	1.2e6(7e5)	2/15
CMA-ES-	1.1e4(1751)	6.4e4(2e4)	2.9e5(2e5)	4.5e5(3e5)	4.5e5(5e5)	4.5e5(4e5)	4.5e5(5e5)	8/15
CMA-ESw	3.5e4(2e4)	3.7e4(2e4)	**4.3e4**(2e4)	5.6e4(6e4)	5.8e4(6e4)	6.1e4(7e4)	6.6e4(6e4)	15/15
DE-scip	**1.0e4**(1948)	**1.7e4**(2474)	5.3e4(5931)	8.9e4(2e5)	9.4e4(2e5)	1.0e5(2e5)	1.1e5(2e5)	13/15
cma-NoI	4.2e4(2e4)	2.0e5(1e5)	3.7e5(8e4)	9.8e5(1e6)	1.2e6(8e5)	2.0e6(1e6)	6.0e6(7e6)	0/15
cmaIH1e	1.2e4(3940)	3.6e4(3e4)	5.1e4(3e4)	**5.4e4**(4e4)	**5.5e4**(4e4)	**5.7e4**(4e4)	**6.0e4**(4e4)	15/15
f3								
cma-IH	1623(444)	1.2e5(9e4)	5.7e6(8e6)	∞	∞	∞	∞ *4e5*	0/15
CMA-ES-	4467(4746)	6.0e5(5e5)	∞	∞	∞	∞	∞ *4e5*	0/15
CMA-ESw	6105(1148)	2.1e5(3e5)	∞	∞	∞	∞	∞ *4e5*	0/15
DE-scip	1.7e4(4340)	6.7e5(7e5)	5.7e6(6e6)	∞	∞	∞	∞ *4e5*	0/15
cma-NoI	3292(1500)	1.2e5(9e4)	**2.9e6**(4e6)	∞	∞	∞	∞ *4e5*	0/15
cmaIH1e	**1558**(431)	**2.5e4**(2e4)*	5.6e6(8e6)	∞	∞	∞	∞ *4e5*	0/15
ppdata/**f4**								
cma-IH	8499(1e4)	2.0e5(2e5)	∞	∞	∞	∞	∞ *4e5*	0/15
CMA-ES-	1.2e4(1e4)	5.9e6(5e6)	∞	∞	∞	∞	∞ *4e5*	0/15
CMA-ESw	2.5e4(2e4)	3.8e5(3e5)	∞	∞	∞	∞	∞ *4e5*	0/15
DE-scip	2.2e4(3909)	5.2e5(5e5)	∞	∞	∞	∞	∞ *4e5*	0/15
cma-NoI	1.2e4(1e4)	2.3e5(2e5)	∞	∞	∞	∞	∞ *4e5*	0/15
cmaIH1e	**2921**(654)	**1.6e5**(2e5)	∞	∞	∞	∞	∞ *4e5*	0/15
f5								
cma-IH	1270(184)	3429(2040)	3429(2040)	3429(2040)	3429(2040)	3429(2040)	3429(2040)	15/15
CMA-ES-	1.8e4(2e4)	7.5e4(8e4)	7.5e4(8e4)	7.6e4(8e4)	7.9e4(8e4)	8.8e4(6e4)	1.0e5(5e4)	15/15
CMA-ESw	**887**(370)$^{\star 2}$	**1451**(780)	**1458**(792)	**1458**(792)	**1458**(792)	**1458**(792)	**1458**(792)	15/15
DE-scip	4.2e4(7980)	1.8e5(2e5)	2.0e5(2e5)	2.2e5(2e5)	2.4e5(2e5)	2.8e5(2e5)	3.3e5(2e5)	12/15
cma-NoI	1429(230)	4171(4692)	4171(4692)	4171(4692)	4171(4692)	4171(4692)	4171(4692)	15/15
cmaIH1e	1318(226)	3429(4218)	3429(4218)	3429(4218)	3429(4218)	3429(4218)	3429(4218)	15/15
f6								
cma-IH	**1283**(816)	5.2e5(6e5)	∞	∞	∞	∞	∞ *4e5*	0/15
CMA-ES-	4.5e4(7e4)	2.9e6(4e6)	∞	∞	∞	∞	∞ *4e5*	0/15
CMA-ESw	9.6e4(2e5)	2.6e6(2e6)	∞	∞	∞	∞	∞ *4e5*	0/15
DE-scip	1.7e4(6864)	5.9e5(6e5)	∞	∞	∞	∞	∞ *4e5*	0/15
cma-NoI	1316(620)	**4.3e5**(8e5)	∞	∞	∞	∞	∞ *4e5*	0/15
cmaIH1e	4796(7773)	6.2e5(6e5)	∞	∞	∞	∞	∞ *4e5*	0/15
f7								
cma-IH	5.2e5(4e5)	8.2e5(6e5)	8.3e5(8e5)	8.4e5(1e6)	8.4e5(5e5)	8.4e5(8e5)	8.4e5(1e6)	6/15
CMA-ES-	4.8e5(4e5)	1.4e6(2e6)	1.4e6(1e6)	1.4e6(1e6)	1.4e6(1e6)	1.4e6(1e6)	1.4e6(2e6)	4/15
CMA-ESw	∞	∞	∞	∞	∞	∞	∞ *4e5*	0/15
DE-scip	∞	∞	∞	∞	∞	∞	∞ *4e5*	0/15
cma-NoI	4.0e5(4e5)	5.2e5(6e5)	6.0e5(4e5)	6.0e5(8e5)	6.0e5(7e5)	6.0e5(4e5)	6.0e5(6e5)	7/15
cmaIH1e	**4.0e5**(5e5)	**4.1e5**(3e5)	**4.2e5**(4e5)	**4.2e5**(4e5)	**4.2e5**(4e5)	**4.2e5**(4e5)	**4.2e5**(5e5)	9/15
f8								
cma-IH	2340(2820)	4.0e5(4e5)	∞	∞	∞	∞	∞ *4e5*	0/15
CMA-ES-	5.3e4(6e4)	1.9e5(3e5)	∞	∞	∞	∞	∞ *4e5*	0/15
CMA-ESw	6995(4116)	1.5e5(2e5)	∞	∞	∞	∞	∞ *4e5*	0/15
DE-scip	4.1e4(4962)	3.9e5(6e5)	**5.6e6**(9e6)	**5.7e6**(5e6)	**5.7e6**(8e6)	**5.8e6**(4e6)	**5.8e6**(7e6)	1/15
cma-NoI	3682(8742)	3.8e5(1e5)	∞	∞	∞	∞	∞ *4e5*	0/15
cmaIH1e	**1487**(282)	**6.0e4**(9e4)	∞	∞	∞	∞	∞ *4e5*	0/15
f9								
cma-IH	1.4e4(9322)	∞	∞	∞	∞	∞	∞ *4e5*	0/15
CMA-ES-	6.9e5(8e5)	∞	∞	∞	∞	∞	∞ *4e5*	0/15
CMA-ESw	4.3e4(4112)	∞	∞	∞	∞	∞	∞ *4e5*	0/15
DE-scip	8627(9834)	∞	∞	∞	∞	∞	∞ *4e5*	0/15
cma-NoI	**6811**(2e4)	∞	∞	∞	∞	∞	∞ *4e5*	0/15
f10								
cma-IH	**1.5e6**(2e6)	∞	∞	∞	∞	∞	∞ *4e5*	0/15
CMA-ES-	∞	∞	∞	∞	∞	∞	∞ *4e5*	0/15
CMA-ESw	∞	∞	∞	∞	∞	∞	∞ *4e5*	0/15
DE-scip	∞	∞	∞	∞	∞	∞	∞ *4e5*	0/15
cma-NoI	2.0e6(2e6)	∞	∞	∞	∞	∞	∞ *4e5*	0/15
cmaIH1e	1.5e6(2e6)	∞	∞	∞	∞	∞	∞ *4e5*	0/15
f11								
cma-IH	68(48)	5.8e6(7e6)	∞	∞	∞	∞	∞ *4e5*	0/15
CMA-ES-	76(96)	∞	∞	∞	∞	∞	∞ *4e5*	0/15
CMA-ESw	112(102)	∞	∞	∞	∞	∞	∞ *4e5*	0/15
DE-scip	177(226)	**1.3e5**(8e4)$^{\star 3}$	∞$^{\star 3}$	∞$^{\star 3}$	∞$^{\star 3}$	∞$^{\star 3}$	∞ *4e5*$^{\star 3}$	0/15
cma-NoI	77(48)	1.4e6(1e6)	∞	∞	∞	∞	∞ *4e5*	0/15
cmaIH1e	**64**(52)	1.1e6(9e5)	∞	∞	∞	∞	∞ *4e5*	0/15
f12								
cma-IH	1.1e4(1e4)	1.2e4(1e4)	1.3e4(1e4)	1.5e4(1e4)	1.6e4(1e4)	1.9e4(1e4)	2.2e4(1e4)	15/15
CMA-ES-	7.1e4(7e4)	7.8e4(7e4)	8.4e4(6e4)	9.1e4(6e4)	9.5e4(6e4)	1.1e5(6e4)	1.3e5(6e4)	15/15
CMA-ESw	3.7e4(3e4)	4.0e4(3e4)	4.3e4(3e4)	4.5e4(3e4)	4.6e4(3e4)	4.9e4(3e4)	5.2e4(3e4)	15/15
DE-scip	1.7e5(2e5)	1.7e5(3e5)	1.8e5(2e5)	1.9e5(2e5)	1.9e5(2e5)	2.0e5(2e5)	2.1e5(2e5)	11/15
cma-NoI	5842(9760)	6491(9902)	7584(1e4)	8730(1e4)	1.0e4(1e4)	1.3e4(1e4)	1.6e4(1e4)	15/15
cmaIH1e	**2875**(2073)	**3260**(1858)	**4092**(1654)	**4841**(1462)	**5596**(1310)	**7218**(1566)	**8826**(1664)$^{\star 2}$	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f13								
cma-IH	7429(8778)	2.2e4(2e4)	2.6e4(2e4)	2.9e4(2e4)	3.2e4(3e4)	3.9e4(3e4)	4.6e4(3e4)	15/15
CMA-ES-	7.1e4(7e4)	9.3e4(9e4)	1.1e5(9e4)	1.2e5(8e4)	1.4e5(8e4)	1.9e5(7e4)	∞ *4e5*	0/15
CMA-ESw	1.2e4(1e4)	1.4e4(1e4)	1.6e4(1e4)	1.9e4(1e4)	**2.1e4**(1e4)	**2.6e4**(1e4)	**3.2e4**(1e4)	15/15
DE-scip	8.9e4(2e5)	9.9e4(2e5)	1.1e5(2e5)	1.2e5(2e5)	1.3e5(2e5)	1.4e5(2e5)	1.6e5(2e5)	13/15
cma-NoI	1.0e4(1e4)	1.8e4(2e4)	2.1e4(3e4)	2.4e4(3e4)	2.8e4(3e4)	3.4e4(3e4)	4.1e4(3e4)	15/15
cmaIH1e	**3558**(3540)	**7104**(7991)	**1.2e4**(8631)	**1.7e4**(9380)	2.6e4(9920)	2.9e4(1e4)	3.3e4(1e4)	15/15
f14								
cma-IH	**578**(136)	7193(8184)	1.5e4(2e4)	2.3e4(2e4)	2.8e4(2e4)	3.6e4(2e4)	4.2e4(3e4)	15/15
CMA-ES-	588(200)	6.2e4(6e4)	6.8e4(7e4)	1.1e5(1e5)	1.2e5(1e5)	1.5e5(9e4)	1.9e5(9e4)	11/15
CMA-ESw	1643(629)	3018(658)	7015(1e4)	**9529**(1e4)	**1.2e4**(1e4)	**1.9e4**(1e4)	**2.5e4**(1e4)	15/15
DE-scip	4850(1211)	2.3e4(6259)	3.1e5(4e5)	4.0e5(5e5)	4.2e5(4e5)	4.8e5(4e5)	5.7e5(4e5)	8/15
cma-NoI	652(182)	5705(8030)	2.0e4(4e4)	2.7e4(4e4)	3.1e4(4e4)	3.8e4(5e4)	4.6e4(5e4)	15/15
cmaIH1e	620(134)	**1959**(938)	**5783**(4326)	1.3e4(1e4)	2.2e4(4e4)	4.9e4(4e4)	5.6e4(4e4)	15/15
f15								
cma-IH	2.3e5(6e4)	2.3e5(6e4)	2.3e5(6e4)	2.4e5(6e4)	2.5e5(6e4)	2.6e5(6e4)	2.7e5(7e4)	14/15
CMA-ES-	2.9e6(3e6)	2.9e6(4e6)	2.9e6(4e6)	2.9e6(3e6)	2.9e6(2e6)	2.9e6(3e6)	2.9e6(4e6)	2/15
CMA-ESw	∞	∞	∞	∞	∞	∞	∞ *4e5*	0/15
DE-scip	∞	∞	∞	∞	∞	∞	∞ *4e5*	0/15
cma-NoI	2.3e5(2e5)	2.5e5(2e5)	2.6e5(2e5)	2.6e5(2e5)	2.7e5(2e5)	2.7e5(2e5)	2.8e5(3e5)	12/15
cmaIH1e	**1.4e5**(6e4)	**1.4e5**(6e4)	**1.4e5**(6e4)	**1.5e5**(6e4)	**1.5e5**(6e4)	**1.6e5**(7e4)	**1.7e5**(7e4)	15/15
f16								
cma-IH	4.0e4(5e4)	∞	∞	∞	∞	∞	∞ *4e5*	0/15
CMA-ES-	**1.8e4**(2e4)	∞	∞	∞	∞	∞	∞ *4e5*	0/15
CMA-ESw	8.7e5(8e5)	∞	∞	∞	∞	∞	∞ *4e5*	0/15
DE-scip	∞	∞	∞	∞	∞	∞	∞ *4e5*	0/15
cma-NoI	3.0e4(4e4)	∞	∞	∞	∞	∞	∞ *4e5*	0/15
cmaIH1e	2.8e4(2e4)	∞	∞	∞	∞	∞	∞ *4e5*	0/15
f17								
cma-IH	9577(1e4)	**1.5e4**(3e4)	**1.7e4**(3e4)	**2.0e4**(3e4)	**2.4e4**(3e4)	**2.9e4**(3e4)	**3.6e4**(4e4)	15/15
CMA-ES-	1.0e5(1e5)	1.7e5(1e5)	2.0e5(1e5)	2.2e5(1e5)	2.4e5(1e5)	2.8e5(1e5)	∞ *4e5*	0/15
CMA-ESw	3.1e4(3e4)	1.1e5(2e5)	1.1e5(2e5)	1.2e5(2e5)	1.2e5(2e5)	1.2e5(2e5)	1.3e5(2e5)	13/15
DE-scip	7.0e4(3e4)	2.5e5(3e5)	2.6e5(4e5)	2.7e5(4e5)	2.8e5(3e5)	3.0e5(4e5)	3.2e5(2e5)	10/15
cma-NoI	1.4e4(1e4)	2.5e4(3e4)	2.8e4(3e4)	3.2e4(3e4)	3.6e4(3e4)	4.2e4(3e4)	5.1e4(3e4)	15/15
cmaIH1e	**6473**(8852)	2.5e4(2e4)	2.9e4(2e4)	3.1e4(2e4)	3.4e4(2e4)	3.8e4(2e4)	4.4e4(2e4)	15/15
f18								
cma-IH	1434(678)	**2.7e4**(3e4)	**4.7e4**(4e4)	**5.2e4**(4e4)	**5.6e4**(4e4)	**6.5e4**(4e4)	**7.5e4**(4e4)	15/15
CMA-ES-	1533(335)	4.0e5(3e5)	4.1e5(3e5)	4.3e5(3e5)	4.5e5(3e5)	4.9e5(3e5)	∞ *4e5*	0/15
CMA-ESw	2825(1204)	6.3e5(7e5)	8.2e5(1e6)	8.3e5(8e5)	8.3e5(1e6)	8.4e5(8e5)	8.4e5(1e6)	5/15
DE-scip	2.5e4(2e4)	1.7e6(2e6)	1.7e6(1e6)	1.7e6(2e6)	1.7e6(2e6)	1.7e6(3e6)	1.7e6(2e6)	3/15
cma-NoI	**1260**(378)	4.0e4(5e4)	5.2e4(6e4)	5.6e4(6e4)	6.0e4(6e4)	6.9e4(7e4)	7.7e4(7e4)	15/15
cmaIH1e	1451(568)	4.2e4(5e4)	5.2e4(6e4)	5.6e4(7e4)	6.0e4(7e4)	6.8e4(7e4)	7.7e4(8e4)	15/15
f19								
cma-IH	**2.9e6**(3e6)	**6.0e6**(5e6)	**6.0e6**(5e6)	**6.0e6**(9e6)	**6.0e6**(6e6)	**6.0e6**(7e6)	**6.0e6**(7e6)	0/15
CMA-ES-	∞	∞	∞	∞	∞	∞	∞ *4e5*	0/15
CMA-ESw	∞	∞	∞	∞	∞	∞	∞ *4e5*	0/15
DE-scip	∞	∞	∞	∞	∞	∞	∞ *4e5*	0/15
cma-NoI	∞	∞	∞	∞	∞	∞	∞ *4e5*	0/15
cmaIH1e	∞	∞	∞	∞	∞	∞	∞ *4e5*	0/15
f20								
cma-IH	**867**(340)	1012(274)	1.4e5(8e4)	∞	∞	∞	∞ *4e5*	0/15
CMA-ES-	979(370)	2.0e4(2e4)	4.4e5(4e5)	∞	∞	∞	∞ *4e5*	0/15
CMA-ESw	4501(6348)	4864(6336)	1.8e5(2e5)	∞	∞	∞	∞ *4e5*	0/15
DE-scip	4935(1332)	6037(2072)	1.6e5(4e5)	∞	∞	∞	∞ *4e5*	0/15
cma-NoI	872(187)	**994**(242)	2.5e5(1e5)	∞	∞	∞	∞ *4e5*	0/15
cmaIH1e	875(148)	997(238)	**9.3e4**(1e5)	∞	∞	∞	∞ *4e5*	0/15
f21								
cma-IH	4.4e4(4e4)	9.4e4(2e5)	1.3e5(2e5)	1.4e5(3e5)	1.9e5(2e5)	1.9e5(2e5)	1.9e5(3e5)	11/15
CMA-ES-	1.5e5(3e5)	3.4e5(4e5)	7.4e5(1e6)	7.4e5(6e5)	7.5e5(6e5)	7.5e5(7e5)	7.5e5(7e5)	6/15
CMA-ESw	1.4e5(2e5)	1.9e5(2e5)	2.0e5(2e5)	2.2e5(4e5)	2.2e5(3e5)	2.2e5(3e5)	2.2e5(3e5)	11/15
DE-scip	**3.6e4**(3882)	7.9e4(2e5)	6.3e5(6e5)	1.1e6(1e6)	1.1e6(8e5)	1.2e6(2e6)	1.2e6(1e6)	4/15
cma-NoI	1.0e5(2e5)	1.5e5(3e5)	2.3e5(3e5)	2.4e5(3e5)	2.4e5(3e5)	2.4e5(3e5)	2.5e5(2e5)	11/15
cmaIH1e	4.0e4(7e4)	**4.3e4**(7e4)	**5.9e4**(7e4)	**6.2e4**(8e4)	**6.4e4**(8e4)	**6.7e4**(8e4)	**7.1e4**(8e4)	15/15
f22								
cma-IH	∞	∞	∞	∞	∞	∞	∞ *4e5*	0/15
CMA-ES-	∞	∞	∞	∞	∞	∞	∞ *4e5*	0/15
CMA-ESw	∞	∞	∞	∞	∞	∞	∞ *4e5*	0/15
DE-scip	**5.6e6**(6e6)	∞	∞	∞	∞	∞	∞ *4e5*	0/15
cma-NoI	∞	∞	∞	∞	∞	∞	∞ *4e5*	0/15
cmaIH1e	∞	∞	∞	∞	∞	∞	∞ *4e5*	0/15
f23								
cma-IH	**2.8e5**(3e5)	∞	∞	∞	∞	∞	∞ *4e5*	0/15
CMA-ES-	∞	∞	∞	∞	∞	∞	∞ *4e5*	0/15
CMA-ESw	∞	∞	∞	∞	∞	∞	∞ *4e5*	0/15
DE-scip	∞	∞	∞	∞	∞	∞	∞ *4e5*	0/15
cma-NoI	2.8e5(3e5)	∞	∞	∞	∞	∞	∞ *4e5*	0/15
cmaIH1e	6.6e5(7e5)	∞	∞	∞	∞	∞	∞ *4e5*	0/15
f24								
cma-IH	∞	∞	∞	∞	∞	∞	∞ *4e5*	0/15
CMA-ES-	∞	∞	∞	∞	∞	∞	∞ *4e5*	0/15
CMA-ESw	∞	∞	∞	∞	∞	∞	∞ *4e5*	0/15
DE-scip	∞	∞	∞	∞	∞	∞	∞ *4e5*	0/15
cma-NoI	5.9e6(7e6)	5.9e6(8e6)	5.9e6(7e6)	5.9e6(7e6)	5.9e6(9e6)	5.9e6(5e6)	6.0e6(6e6)	1/15
cmaIH1e	**5.9e6**(5e6)	**5.9e6**(5e6)	**5.9e6**(9e6)	**5.9e6**(8e6)	**5.9e6**(4e6)	**5.9e6**(8e6)	**5.9e6**(6e6)	1/15

Table 2: Expected runtime (ERT) to reach given targets, measured in number of f-evaluations, in dimension 40. For each function, the ERT and, in braces as dispersion measure, the half difference between 10 and 90%-tile of (bootstrapped) runtimes is shown for the different target Δf-values as shown in the top row. #succ is the number of trials that reached the last target $f_{\text{opt}} + 10^{-8}$. The median number of conducted function evaluations is additionally given in *italics*, if the target in the last column was never reached. Entries, succeeded by a star, are statistically significantly better (according to the rank-sum test) when compared to all other algorithms of the table, with $p = 0.05$ or $p = 10^{-k}$ when the number k following the star is larger than 1, with Bonferroni correction by the number of functions (24). Best results are printed in bold. Data produced with COCO v2.6.3

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f1								
cma-IH	**1586**(212)	7261(2e4)	3.5e4(4e4)	3.8e4(4e4)	4.1e4(4e4)	4.8e4(5e4)	5.5e4(5e4)	15/15
CMA-ES-	4.8e4(9e4)	3.1e5(3e5)	3.1e5(3e5)	3.2e5(3e5)	3.4e5(3e5)	4.2e5(3e5)	5.2e5(5e5)	13/15
CMA-ESw	3832(729)	1.1e4(3711)	1.3e4(2772)	1.4e4(3119)	1.6e4(2645)	2.1e4(3882)	2.8e4(3508)	15/15
DE-scip	2.6e4(3658)	1.7e6(2e6)	1.1e7(2e7)	1.1e7(1e7)	1.1e7(2e7)	1.1e7(1e7)	1.1e7(1e7)	1/15
cma-NoI	1653(226)	1.3e4(2e4)	6.5e4(8e4)	6.8e4(8e4)	7.3e4(9e4)	8.3e4(9e4)	9.2e4(1e5)	15/15
cmaIH1e	1652(202)	**2963**(320)	**8453**(4542)	**8650**(3904)	**8988**(3262)	**1.1e4**(1943)$^{\star 3}$	**1.3e4**(956)$^{\star 3}$	15/15
f2								
cma-IH	3.7e5(2e5)	∞	∞	∞	∞	∞	∞ *8e5*	0/15
CMA-ES-	8.3e4(3e4)	3.3e5(1e5)	1.7e6(1e6)	3.8e6(4e6)	1.2e7(9e6)	1.2e7(2e7)	1.2e7(2e7)	1/15
CMA-ESw	2.3e5(2e5)	2.5e5(2e5)	2.6e5(2e5)	3.8e5(3e5)	3.9e5(3e5)	4.6e5(5e5)	4.7e5(5e5)	13/15
DE-scip	2.6e4(5618)	1.7e5(4e5)	7.7e5(1e6)	1.0e6(1e6)	1.0e6(1e6)	1.0e6(1e6)	1.0e6(1e6)	7/15
cma-NoI	3.1e5(1e5)	∞	∞	∞	∞	∞	∞ *8e5*	0/15
cmaIH1e	**2.2e4**(9422)	**3.9e4**(3e4)	**8.1e4**(9e4)	**1.0e5**(9e4)	**1.0e5**(9e4)	**1.0e5**(9e4)	**1.1e5**(9e4)*	15/15
f3								
cma-IH	4157(1210)	1.1e6(8e5)	∞	∞	∞	∞	∞ *8e5*	0/15
CMA-ES-	2.5e4(4e4)	1.2e7(9e6)	∞	∞	∞	∞	∞ *8e5*	0/15
CMA-ESw	8267(2116)	3.6e6(4e6)	∞	∞	∞	∞	∞ *8e5*	0/15
DE-scip	5.7e4(1e4)	∞	∞	∞	∞	∞	∞ *8e5*	0/15
cma-NoI	4107(1556)	1.3e6(1e6)	∞	∞	∞	∞	∞ *8e5*	0/15
cmaIH1e	**4024**(749)	**1.7e5**(2e5)$^{\star 2}$	**1.2e7**(1e7)	∞	∞	∞	∞ *8e5*	0/15
f4								
cma-IH	3.1e4(2e4)	2.4e6(1e6)	∞	∞	∞	∞	∞ *8e5*	0/15
CMA-ES-	1.7e5(2e5)	∞	∞	∞	∞	∞	∞ *8e5*	0/15
CMA-ESw	1.5e5(1e5)	5.7e6(5e6)	∞	∞	∞	∞	∞ *8e5*	0/15
DE-scip	6.6e4(9802)	∞	∞	∞	∞	∞	∞ *8e5*	0/15
cma-NoI	3.6e4(2e4)	6.0e6(8e6)	∞	∞	∞	∞	∞ *8e5*	0/15
cmaIH1e	**7833**(2775)	**8.4e5**(1e6)	∞	∞	∞	∞	∞ *8e5*	0/15
f5								
cma-IH	2535(202)	4.1e4(2e4)	4.1e4(2e4)	4.1e4(2e4)	4.1e4(2e4)	4.1e4(2e4)	4.1e4(2e4)	15/15
CMA-ES-	1.3e5(1e5)	2.6e6(4e6)	2.6e6(2e6)	2.6e6(3e6)	2.6e6(2e6)	2.6e6(3e6)	2.6e6(4e6)	4/15
CMA-ESw	**1363**(376)$^{\star 4}$	**2790**(1038)*	**2833**(1096)*	**2847**(1104)*	**2847**(1104)*	**2847**(1104)*	**2847**(1104)*	15/15
DE-scip	1.7e5(5e4)	∞	∞	∞	∞	∞	∞ *8e5*	0/15
cma-NoI	2543(220)	8.8e4(2e5)	8.8e4(2e5)	8.8e4(2e5)	8.8e4(2e5)	8.8e4(2e5)	8.8e4(2e5)	15/15
cmaIH1e	2530(282)	4704(3582)	4731(3600)	4741(3590)	4741(3590)	4741(3590)	4741(3590)	15/15
f6								
cma-IH	6187(1424)	1.1e7(1e7)	∞	∞	∞	∞	∞ *8e5*	0/15
CMA-ES-	9.0e4(2e5)	∞	∞	∞	∞	∞	∞ *8e5*	0/15
CMA-ESw	2.7e5(4e5)	∞	∞	∞	∞	∞	∞ *8e5*	0/15
DE-scip	5.0e4(2e4)	∞	∞	∞	∞	∞	∞ *8e5*	0/15
cma-NoI	1.2e4(2e4)	5.5e6(7e6)	∞	∞	∞	∞	∞ *8e5*	0/15
cmaIH1e	**5491**(2723)	**3.6e6**(3e6)	∞	∞	∞	∞	∞ *8e5*	0/15
f7								
cma-IH	∞	∞	∞	∞	∞	∞	∞ *8e5*	0/15
CMA-ES-	∞	∞	∞	∞	∞	∞	∞ *8e5*	0/15
CMA-ESw	∞	∞	∞	∞	∞	∞	∞ *8e5*	0/15
DE-scip	∞	∞	∞	∞	∞	∞	∞ *8e5*	0/15
cma-NoI	∞	∞	∞	∞	∞	∞	∞ *8e5*	0/15
cmaIH1e	∞	∞	∞	∞	∞	∞	∞ *8e5*	0/15
f8								
cma-IH	2.1e4(2e4)	2.5e6(3e6)	∞	∞	∞	∞	∞ *8e5*	0/15
CMA-ES-	2.1e5(3e5)	3.6e6(4e6)	∞	∞	∞	∞	∞ *8e5*	0/15
CMA-ESw	3.5e4(2e4)	1.4e6(2e6)	∞	∞	∞	∞	∞ *8e5*	0/15
DE-scip	1.7e5(4e5)	5.8e6(5e6)	∞	∞	∞	∞	∞ *8e5*	0/15
cma-NoI	7996(2e4)	2.5e6(3e6)	∞	∞	∞	∞	∞ *8e5*	0/15
cmaIH1e	**2789**(492)	**5.4e5**(6e5)	∞	∞	∞	∞	∞ *8e5*	0/15
f9								
cma-IH	3.2e4(4e4)	∞	∞	∞	∞	∞	∞ *8e5*	0/15
CMA-ES-	1.4e6(1e6)	∞	∞	∞	∞	∞	∞ *8e5*	0/15
CMA-ESw	4.8e5(4e5)	∞	∞	∞	∞	∞	∞ *8e5*	0/15
DE-scip	3.3e4(5e4)	∞	∞	∞	∞	∞	∞ *8e5*	0/15
cma-NoI	**2.6e4**(4e4)	∞	∞	∞	∞	∞	∞ *8e5*	0/15
f10								
cma-IH	∞	∞	∞	∞	∞	∞	∞ *8e5*	0/15
CMA-ES-	∞	∞	∞	∞	∞	∞	∞ *8e5*	0/15
CMA-ESw	∞	∞	∞	∞	∞	∞	∞ *8e5*	0/15
DE-scip	∞	∞	∞	∞	∞	∞	∞ *8e5*	0/15
cma-NoI	∞	∞	∞	∞	∞	∞	∞ *8e5*	0/15
cmaIH1e	∞	∞	∞	∞	∞	∞	∞ *8e5*	0/15
f11								
cma-IH	6954(1e4)	5.7e6(5e6)	∞	∞	∞	∞	∞ *8e5*	0/15
CMA-ES-	**751**(646)	3.7e6(5e6)	∞	∞	∞	∞	∞ *8e5*	0/15
CMA-ESw	921(738)	∞	∞	∞	∞	∞	∞ *8e5*	0/15
DE-scip	4009(4052)	∞	∞	∞	∞	∞	∞ *8e5*	0/15
cma-NoI	7208(1e4)	**2.7e6**(3e6)	∞	∞	∞	∞	∞ *8e5*	0/15
cmaIH1e	1.8e4(2e4)	5.7e6(6e6)	∞	∞	∞	∞	∞ *8e5*	0/15
f12								
cma-IH	3.2e4(2e4)	2.0e5(3e5)	2.1e5(3e5)	2.1e5(3e5)	2.2e5(3e5)	2.4e5(4e5)	2.7e5(4e5)	13/15
CMA-ES-	4.1e5(4e5)	4.2e5(5e5)	4.4e5(5e5)	4.7e5(5e5)	5.4e5(4e5)	6.7e5(5e5)	7.7e5(4e5)	11/15
CMA-ESw	2.5e5(1e5)	2.6e5(1e5)	2.9e5(2e5)	2.9e5(2e5)	2.9e5(2e5)	3.0e5(2e5)	3.1e5(2e5)	14/15
DE-scip	7.0e5(8e5)	1.1e6(1e6)	1.2e6(2e6)	1.2e6(1e6)	1.2e6(1e6)	1.5e6(1e6)	1.5e6(2e6)	6/15
cma-NoI	2.9e4(2e4)	8.2e4(1e5)	8.7e4(1e5)	9.1e4(1e5)	9.6e4(1e5)	1.0e5(1e5)	1.1e5(1e5)	15/15
cmaIH1e	**6361**(2626)	**8935**(4294)*	**1.2e4**(7724)$^{\star 2}$	**2.4e4**(1e4)$^{\star 2}$	**2.7e4**(2e4)$^{\star 2}$	**4.3e4**(9e4)*	**4.7e4**(9e4)*	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f13								
cma-IH	1.3e5(3e5)	2.3e5(3e5)	2.4e5(3e5)	2.6e5(3e5)	2.8e5(4e5)	3.5e5(4e5)	4.0e5(4e5)	11/15
CMA-ES-	6.7e5(5e5)	7.8e5(7e5)	8.1e5(9e5)	8.4e5(5e5)	9.1e5(7e5)	1.6e6(1e6)	∞ *8e5*	0/15
CMA-ESw	2.9e4(2e4)	4.3e4(2e4)	**4.7e4**(2e4)	**5.3e4**(2e4)	**5.9e4**(2e4)	**7.1e4**(2e4)	**8.2e4**(2e4)	15/15
DE-scip	3.3e6(4e6)	5.5e6(6e6)	5.6e6(5e6)	5.6e6(4e6)	5.6e6(6e6)	5.7e6(8e6)	5.7e6(5e6)	2/15
cma-NoI	5.1e4(8e4)	1.8e5(3e5)	1.9e5(3e5)	2.1e5(3e5)	2.3e5(3e5)	2.7e5(4e5)	3.2e5(4e5)	13/15
cmaIH1e	**6941**(3740)	**4.1e4**(9e4)	8.6e4(1e5)	9.5e4(2e5)	1.0e5(2e5)	1.1e5(2e5)	1.2e5(2e5)	15/15
f14								
cma-IH	1018(256)	2.1e4(2e4)	2.0e5(3e5)	4.7e5(6e5)	4.9e5(7e5)	5.4e5(8e5)	6.9e5(8e5)	9/15
CMA-ES-	1049(213)	8.2e5(8e5)	8.4e5(8e5)	1.9e6(1e6)	1.9e6(2e6)	2.5e6(3e6)	2.7e6(3e6)	4/15
CMA-ESw	3074(863)	5768(832)	8547(1574)	**4.1e4**(3e4)	**4.6e4**(3e4)	**7.6e4**(4e4)	**1.1e5**(5e4)	15/15
DE-scip	9388(2766)	6.3e4(3e4)	1.8e6(3e6)	5.3e6(8e6)	5.3e6(5e6)	5.3e6(7e6)	5.4e6(3e6)	2/15
cma-NoI	**1000**(175)	1.1e5(3e5)	2.3e5(3e5)	6.1e5(9e5)	6.4e5(7e5)	7.0e5(7e5)	8.3e5(9e5)	9/15
cmaIH1e	1068(308)	**3381**(575)	**8056**(3217)	4.8e4(8e4)	6.0e4(9e4)	1.8e5(1e5)	2.5e5(1e5)	15/15
f15								
cma-IH	1.3e6(1e6)	1.8e6(1e6)	2.1e6(2e6)	2.7e6(3e6)	2.7e6(2e6)	3.6e6(5e6)	3.6e6(4e6)	3/15
CMA-ES-	∞	∞	∞	∞	∞	∞	∞ *8e5*	0/15
CMA-ESw	∞	∞	∞	∞	∞	∞	∞ *8e5*	0/15
DE-scip	∞	∞	∞	∞	∞	∞	∞ *8e5*	0/15
cma-NoI	1.9e6(2e6)	1.9e6(2e6)	2.9e6(3e6)	5.8e6(8e6)	1.2e7(1e7)	1.2e7(2e7)	1.2e7(1e7)	1/15
cmaIH1e	**9.5e5**(1e6)	**1.1e6**(1e6)	**1.1e6**(1e6)	**1.1e6**(1e6)	**1.5e6**(1e6)	**1.6e6**(2e6)	**1.6e6**(1e6)	6/15
f16								
cma-IH	8.2e4(9e4)	∞	∞	∞	∞	∞	∞ *8e5*	0/15
CMA-ES-	**3.0e4**(5e4)	∞	∞	∞	∞	∞	∞ *8e5*	0/15
CMA-ESw	∞	∞	∞	∞	∞	∞	∞ *8e5*	0/15
DE-scip	∞	∞	∞	∞	∞	∞	∞ *8e5*	0/15
cma-NoI	5.0e4(4e4)	∞	∞	∞	∞	∞	∞ *8e5*	0/15
cmaIH1e	9.6e4(1e5)	∞	∞	∞	∞	∞	∞ *8e5*	0/15
f17								
cma-IH	7.4e4(1e5)	4.5e5(4e5)	4.7e5(5e5)	4.8e5(5e5)	4.9e5(6e5)	5.1e5(5e5)	5.4e5(5e5)	12/15
CMA-ES-	8.2e5(2e6)	1.2e7(1e7)	∞	∞	∞	∞	∞ *8e5*	0/15
CMA-ESw	5.1e5(9e5)	2.1e6(3e6)	2.1e6(2e6)	2.1e6(2e6)	2.1e6(2e6)	2.1e6(3e6)	2.7e6(2e6)	3/15
DE-scip	3.5e6(4e6)	∞	∞	∞	∞	∞	∞ *8e5*	0/15
cma-NoI	3.5e4(3e4)	2.9e5(8e5)	3.0e5(4e5)	3.1e5(4e5)	3.2e5(4e5)	3.4e5(4e5)	3.6e5(4e5)	12/15
cmaIH1e	**3.1e4**(4e4)	**2.1e5**(3e5)	**2.2e5**(4e5)	**2.5e5**(4e5)	**2.8e5**(4e5)	**2.9e5**(5e5)	**3.1e5**(4e5)	13/15
f18								
cma-IH	3464(1066)	1.7e6(1e6)	1.7e6(1e6)	1.7e6(1e6)	1.7e6(2e6)	2.1e6(3e6)	2.6e6(4e6)	4/15
CMA-ES-	3972(1193)	∞	∞	∞	∞	∞	∞ *8e5*	0/15
CMA-ESw	7629(2337)	∞	∞	∞	∞	∞	∞ *8e5*	0/15
DE-scip	9.5e4(5e4)	∞	∞	∞	∞	∞	∞ *8e5*	0/15
cma-NoI	**3215**(840)	1.2e6(1e6)	1.6e6(2e6)	1.6e6(2e6)	1.6e6(2e6)	1.6e6(2e6)	1.7e6(2e6)	6/15
cmaIH1e	3249(1067)	**1.2e6**(2e6)	**1.5e6**(2e6)	**1.5e6**(1e6)	**1.6e6**(2e6)	**1.6e6**(3e6)	**1.6e6**(2e6)	6/15
f19								
cma-IH	**1.2e7**(1e7)	∞	∞	∞	∞	∞	∞ *8e5*	0/15
CMA-ES-	∞	∞	∞	∞	∞	∞	∞ *8e5*	0/15
CMA-ESw	∞	∞	∞	∞	∞	∞	∞ *8e5*	0/15
DE-scip	∞	∞	∞	∞	∞	∞	∞ *8e5*	0/15
cma-NoI	1.2e7(1e7)	∞	∞	∞	∞	∞	∞ *8e5*	0/15
cmaIH1e	∞	∞	∞	∞	∞	∞	∞ *8e5*	0/15
f20								
cma-IH	**2142**(583)	5990(8398)	∞	∞	∞	∞	∞ *8e5*	0/15
CMA-ES-	4095(846)	7.6e4(2e5)	1.2e7(1e7)	∞	∞	∞	∞ *8e5*	0/15
CMA-ESw	1.2e5(2e5)	1.3e5(2e5)	2.5e6(5e6)	∞	∞	∞	∞ *8e5*	0/15
DE-scip	1.5e4(3040)	1.9e4(4690)	∞	∞	∞	∞	∞ *8e5*	0/15
cma-NoI	2207(552)	**2525**(621)	1.2e7(2e7)	∞	∞	∞	∞ *8e5*	0/15
cmaIH1e	2205(238)	2712(1002)	**2.4e6**(4e6)	∞	∞	∞	∞ *8e5*	0/15
f21								
cma-IH	**6.6e4**(7e4)	**8.1e4**(7e4)	4.8e5(9e5)	7.3e5(1e6)	9.6e5(1e6)	1.4e6(2e6)	1.4e6(2e6)	6/15
CMA-ES-	2.9e5(8e5)	8.7e5(9e5)	3.6e6(3e6)	1.2e7(8e6)	∞	∞	∞ *8e5*	0/15
CMA-ESw	3.8e5(8e5)	5.0e5(7e5)	6.4e5(1e6)	7.5e5(9e5)	1.2e6(1e6)	1.8e6(2e6)	1.8e6(2e6)	5/15
DE-scip	∞	∞	∞	∞	∞	∞	∞ *8e5*	0/15
cma-NoI	1.5e5(4e5)	2.0e5(5e5)	2.4e5(5e5)	4.4e5(5e5)	9.0e5(9e5)	1.1e6(1e6)	1.1e6(2e6)	8/15
cmaIH1e	1.6e5(4e5)	1.6e5(4e5)	**1.9e5**(4e5)	**3.0e5**(5e5)	**7.1e5**(9e5)	**7.7e5**(9e5)	**7.9e5**(1e6)	10/15
f22								
cma-IH	∞	∞	∞	∞	∞	∞	∞ *8e5*	0/15
CMA-ES-	∞	∞	∞	∞	∞	∞	∞ *8e5*	0/15
CMA-ESw	∞	∞	∞	∞	∞	∞	∞ *8e5*	0/15
DE-scip	∞	∞	∞	∞	∞	∞	∞ *8e5*	0/15
cma-NoI	∞	∞	∞	∞	∞	∞	∞ *8e5*	0/15
cmaIH1e	∞	∞	∞	∞	∞	∞	∞ *8e5*	0/15
f23								
cma-IH	**1.9e5**(2e5)	∞	∞	∞	∞	∞	∞ *8e5*	0/15
CMA-ES-	1.2e7(1e7)	∞	∞	∞	∞	∞	∞ *8e5*	0/15
CMA-ESw	∞	∞	∞	∞	∞	∞	∞ *8e5*	0/15
DE-scip	∞	∞	∞	∞	∞	∞	∞ *8e5*	0/15
cma-NoI	2.3e5(2e5)	∞	∞	∞	∞	∞	∞ *8e5*	0/15
cmaIH1e	5.7e6(7e6)	∞	∞	∞	∞	∞	∞ *8e5*	0/15
f24								
cma-IH	∞	∞	∞	∞	∞	∞	∞ *8e5*	0/15
CMA-ES-	∞	∞	∞	∞	∞	∞	∞ *8e5*	0/15
CMA-ESw	∞	∞	∞	∞	∞	∞	∞ *8e5*	0/15
DE-scip	∞	∞	∞	∞	∞	∞	∞ *8e5*	0/15
cma-NoI	5.9e6(8e6)	∞	∞	∞	∞	∞	∞ *8e5*	0/15
cmaIH1e	**3.8e6**(5e6)	∞	∞	∞	∞	∞	∞ *8e5*	0/15

Table 3: Expected runtime (ERT) to reach given targets, measured in number of f-evaluations, in dimension 80. For each function, the ERT and, in braces as dispersion measure, the half difference between 10 and 90%-tile of (bootstrapped) runtimes is shown for the different target Δf-values as shown in the top row. #succ is the number of trials that reached the last target $f_{\text{opt}} + 10^{-8}$. The median number of conducted function evaluations is additionally given in *italics*, if the target in the last column was never reached. Entries, succeeded by a star, are statistically significantly better (according to the rank-sum test) when compared to all other algorithms of the table, with $p = 0.05$ or $p = 10^{-k}$ when the number k following the star is larger than 1, with Bonferroni correction by the number of functions (24). Best results are printed in bold. Data produced with COCO v2.6.3

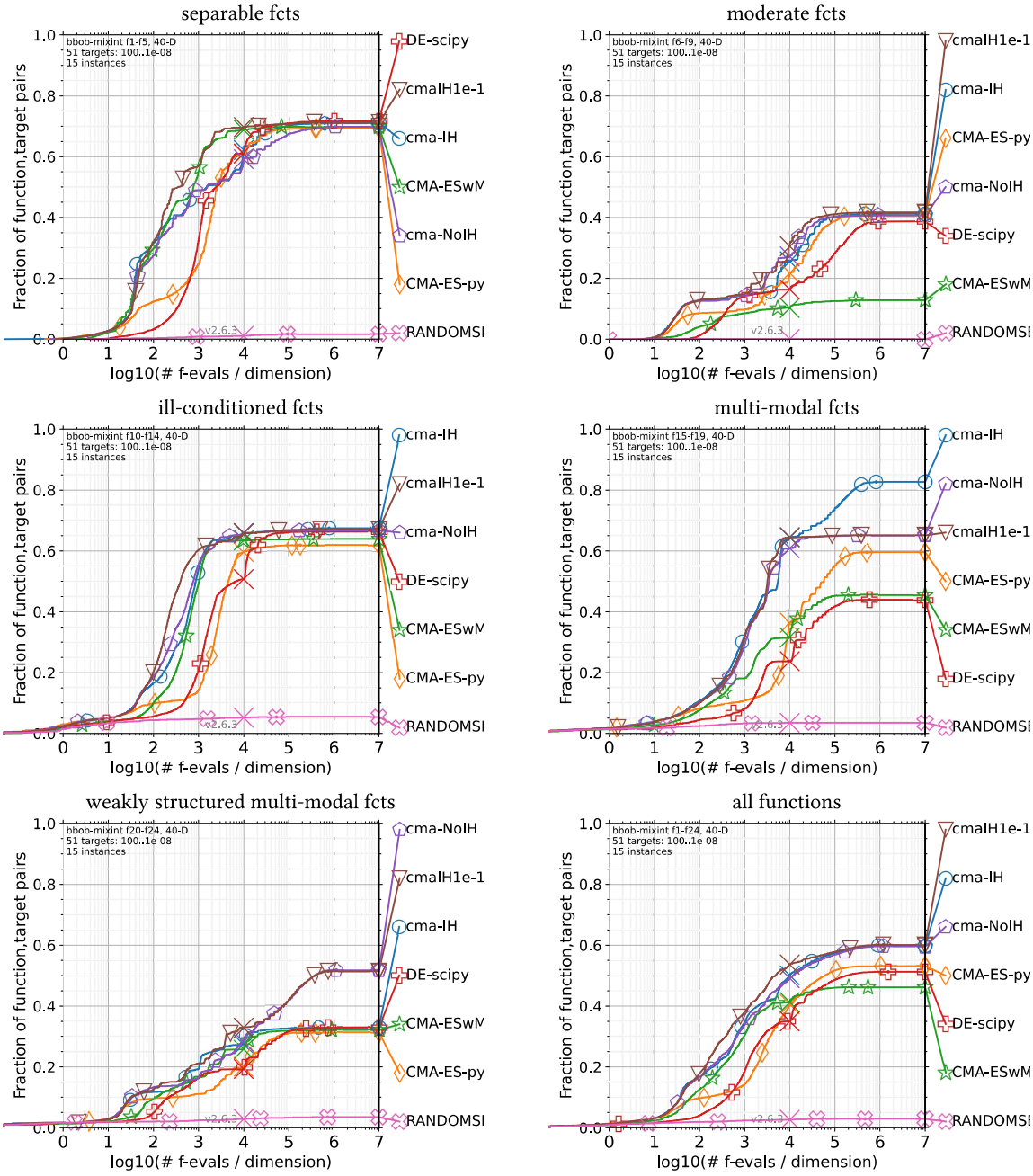

Figure 2: Bootstrapped empirical cumulative distribution of the number of f-evaluations divided by dimension (FEvals/DIM) for 51 targets with target precision in $10^{[-8..2]}$ for all functions and subgroups in 40-D.

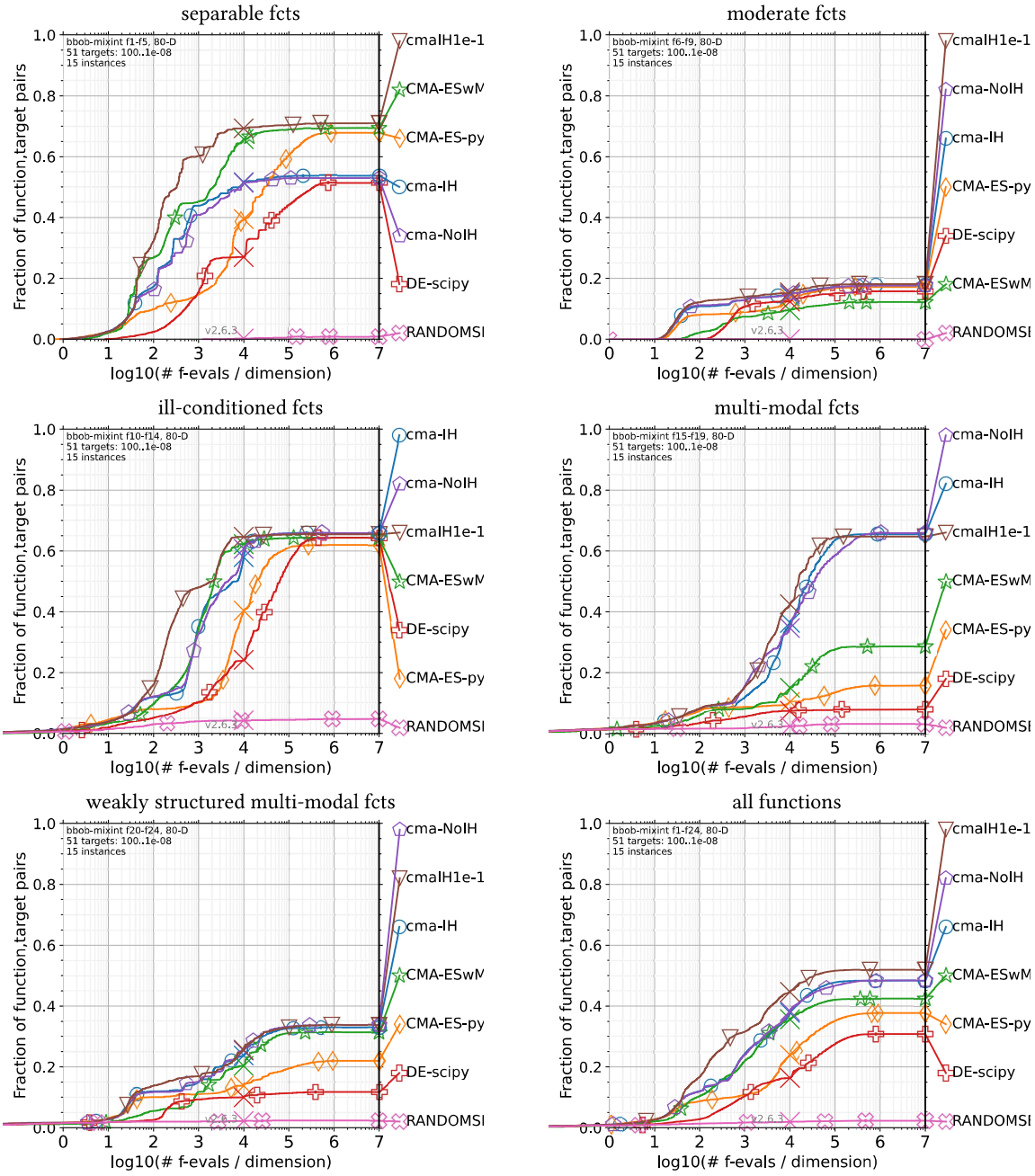

Figure 3: Bootstrapped empirical cumulative distribution of the number of f-evaluations divided by dimension (FEvals/DIM) for 51 targets with target precision in $10^{[-8..2]}$ for all functions and subgroups in 80-D.

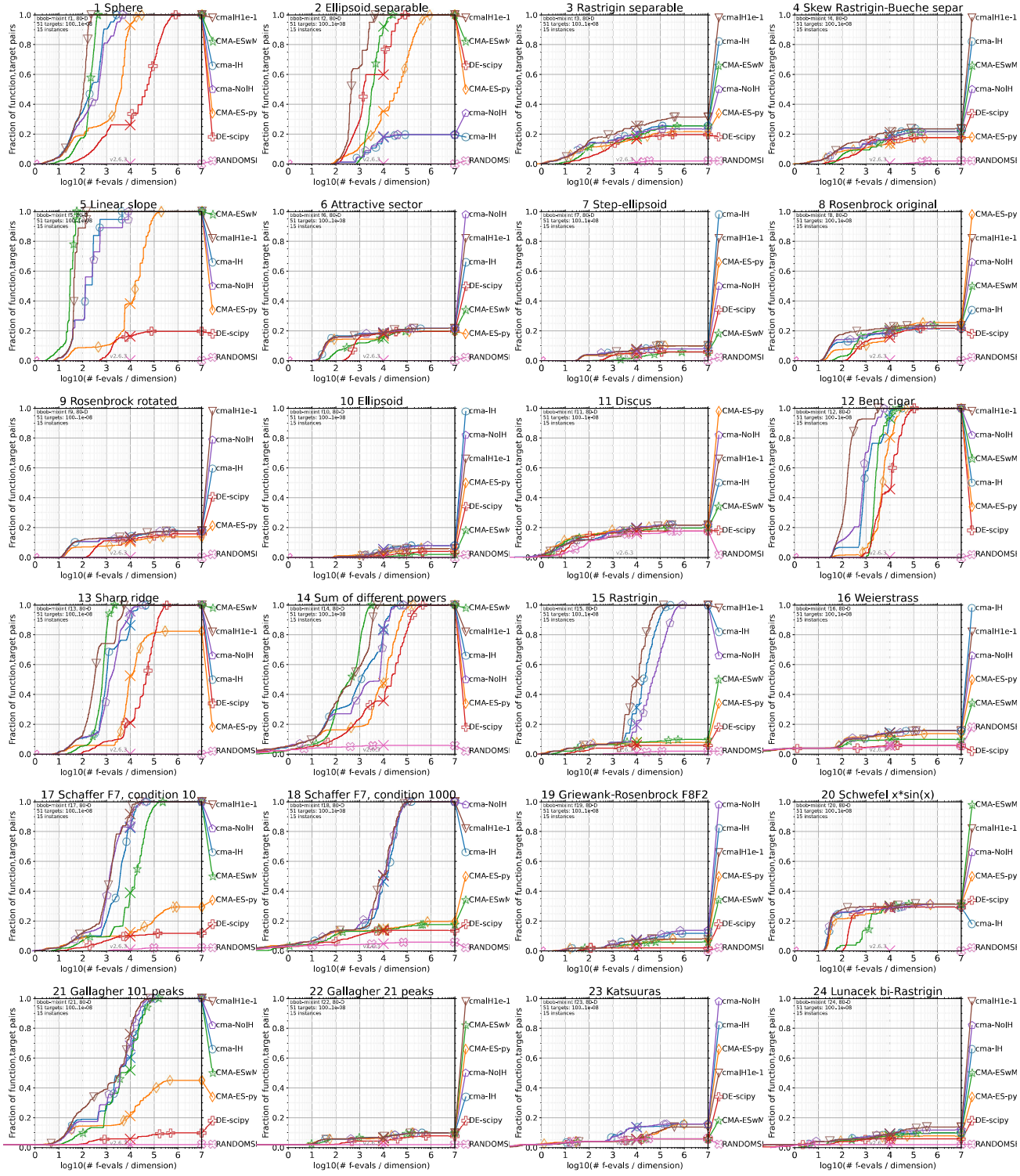

Figure 4: Empirical cumulative distribution of simulated (bootstrapped) runtimes, measured in number of f-evaluations, divided by dimension (FEvals/DIM) for the 51 targets $10^{[-8..2]}$ in dimension 80.