Evaluation of the impact of various modifications to CMA-ES that facilitate its theoretical analysis

Armand Gissler

Evaluation of the impact of various modifications to CMA-ES that facilitate its theoretical analysis

Armand Gissler
firstname.lastname@polytechnique.edu
Inria, CMAP, CNRS, École polytechnique, Institut Polytechnique de Paris
Palaiseau, France

ABSTRACT

In this paper we introduce modified versions of CMA-ES with the objective to help to prove convergence of CMA-ES. In order to ensure that the modifications do not alter the performances of the algorithm too much, we benchmark variants of the algorithm derived from them on problems of the bbob test suite. We observe that the main performances losses are observed on ill-conditioned problems, which is probably due to the absence of cumulation in the adaptation of the covariance matrix. However, the versions of CMA-ES presented in this paper have globally similar performances to the original CMA-ES.

CCS CONCEPTS

• **Mathematics of computing** → **Continuous functions**.

KEYWORDS

Benchmarking, Black-box optimization

ACM Reference Format:
Armand Gissler. 2023. Evaluation of the impact of various modifications to CMA-ES that facilitate its theoretical analysis. In *Genetic and Evolutionary Computation Conference Companion (GECCO '23 Companion), July 15–19, 2023, Lisbon, Portugal.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3583133.3596329

1 INTRODUCTION

In black-box optimization, the derivative-free algorithm CMA-ES (Covariance matrix adapation - Evolution strategies) [8] has shown good performances [5, 7] on many optimization problems. Although its theoretical foundations have progressed in the recent years, establishing a proof of convergence, even for very simple objective functions, remains very challenging.

In order to reduce the complexity of a theoretical analysis via the stability of a normalized Markov chain [10], we introduce in this paper small changes in the update equations. For instance, having continuously differentiable updates allows the use of verifiable conditions, in particular for the irreducibility of the forementioned normalized Markov chain [2]. So far, this approach was succesful to prove linear convergence for some algorithms in the ES class, e.g., $(1+1)$-ES [9] or $(\mu/\mu_w, \lambda)$-ES [11]. However, the adapation of a covariance matrix and cumulation increases the size of the state space of a normalized Markov chain, since it must include these parameters in addition to a normalized mean. Furthermore, a slightly different approach in the update of cumulative paths makes the algorithm that we know is affine-invariant and may extend the class of objective functions for which a proof of convergence exists.

In this paper, we examine modifications of the CMA-ES algorithm to address these different issues and benchmark variants of the CMA-ES algorithm with and without modifications. Our objective is to see whether and how much the performance of the original algorithm deteriorates.

All in all, we compare 10 variants of CMA-ES (including the original one) on the COCO platform [6], on 24 problems in the bbob suite.

This paper is organized as follows. In Section 2, we present the algorithm update equations as well as the considered modifications. Section 3 provides details on how the results were produced. We give the CPU timings of each variant in Section 4. In Section 5 we show the obtained results. We conclude in Section 6.

2 ALGORITHM PRESENTATION

The algorithm CMA-ES updates at each iteration $t \in \mathbb{N}$ the mean $m_t \in \mathbb{R}^d$, the stepsize $\sigma_t > 0$ and the covariance matrix $C_t \in \mathbb{S}^d_{++}$ –that is, C_t belongs to the set of positive definite symmetric matrices of size $d \times d$– of a multivariate normal distribution $\mathcal{N}(m_t, \sigma_t^2 C_t)$, as shown by Eqs. (2), (4) and (6) below. Besides, to better adapt the stepsize and the covariance matrix, CMA-ES relies on cumulation, that is it takes into account the favored directions in the previous iterations to update the stepsize and the covariance matrix. This is translated mathematically by the paths $p_t^\sigma \in \mathbb{R}^d$ and $p_t^c \in \mathbb{R}^d$, which are updated according to Eqs. (3) and (5).

Hence, at each iteration $t \in \mathbb{N}$, given current mean $m_t \in \mathbb{R}^d$, stepsize $\sigma_t > 0$, covariance matrix $C_t \in \mathbb{S}^d_{++}$, and cumulation paths p_t^c, p_t^σ, we generate a population of $\lambda \in \mathbb{N}$ i.i.d. candidate solutions $X_{t+1}^1, \dots, X_{t+1}^\lambda \sim \mathcal{N}(m_t, \sigma_t^2 C_t)$, and we rank them w.r.t. their f-values, i.e. we define indices $i{:}\lambda$ (for $i = 1, \dots, \lambda$) such that

$$f\left(X_{t+1}^{1:\lambda}\right) \leq f\left(X_{t+1}^{2:\lambda}\right) \leq \cdots \leq f\left(X_{t+1}^{\lambda:\lambda}\right). \tag{1}$$

We update then the algorithm parameters according to the following equations

$$m_{t+1} = \sum_{i=1}^{\mu} w_i X_{t+1}^{i:\lambda} \tag{2}$$

$$p_{t+1}^\sigma = (1-c_\sigma) p_t^\sigma + \sqrt{c_\sigma(2-c_\sigma)\mu_{\text{eff}}}\, \sigma_t^{-1} C_t^{-1/2} \left[m_{t+1} - m_t\right] \tag{3}$$

$$\sigma_{t+1} = \sigma_t \times \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|p_{t+1}^\sigma\|}{\mathbb{E}\|\mathcal{N}(0, I_d)\|} - 1\right)\right) \tag{4}$$

$$p_{t+1}^c = (1-c_c) p_t^c + \sqrt{c_c(2-c_c)\mu_{\text{eff}}}\, \sigma_t^{-1} \left[m_{t+1} - m_t\right] \tag{5}$$

GECCO '23 Companion, July 15–19, 2023, Lisbon, Portugal

$$C_{t+1} = (1 - c_1 - c_\mu \sum w_i) C_t + c_1 [p_{t+1}^c][p_{t+1}^c]^T + c_\mu \sigma_t^{-2} \sum_{i=1}^{\lambda} w_i \left[X_{t+1}^{i:\lambda} - m_t\right]\left[X_{t+1}^{i:\lambda} - m_t\right]^T \quad (6)$$

where $c_1, c_\mu \geq 0$ are such that $c_1 + c_\mu \leq 1$, $c_c, c_\sigma > 0$ and the weights $w_1 \geq w_2 \geq \ldots w_\mu \geq 0 \geq w_{\mu+1} \geq \cdots \geq w_\lambda$ are such that $\sum_{i=1}^{\mu} w_i = 1$ and $\sum_{i=1}^{\lambda} w_i \approx 0$. Moreover, the integer $\lambda \geqslant 1$ is called the population size, and $\mu \in \{1, \ldots, \lambda\}$ is the parent number.

First, a possible simplification of the algorithm is to remove cumulation on the covariance matrix and/or the stepsize, i.e. set $c_c = 1$ and/or $c_\sigma = 1$. Second, we can assume that the updates are continuously differentiable w.r.t. the current parameters. Only the update of the stepsize in Eq. (4) is concerned, which we replace then by

$$\sigma_{t+1} = \sigma_t \times \exp\left(\frac{c_\sigma}{2d_\sigma}\left(\frac{\|p_{t+1}^\sigma\|^2}{d} - 1\right)\right). \quad (7)$$

Note that this modification requires to change the value of d_σ. In the default algorithm d_σ is chosen proportional to $\sqrt{\mu_{\text{eff}}}$ when μ is large since $\|p_t^\sigma\|$ scales with $\sqrt{\mu_{\text{eff}}}$ on linear selection. Therefore, with this modification, we have to chose d_σ proportional to μ_{eff} when μ is large, while applying Eq. (7) instead of Eq. (4).

Last, we introduce a modification the replacement of the path p_t^σ by p_t^c, i.e., instead of Eq. (4) we have

$$\sigma_{t+1} = \sigma_t \times \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\left\|C_t^{-1/2} p_{t+1}^c\right\|}{\mathbb{E}\|\mathcal{N}(0, I_d)\|} - 1\right)\right). \quad (8)$$

This modification is indeed helpful in a theoretical context, as it reduces the dimension of the state space of the chain produced by the algorithm, since there would remain only one cumulation path (instead of two in the original algorithm). Besides, we know that this version of the algorithm is *affine-invariant* –we refer to [1] for a formal definition– and allow to extend in some contexts an analysis on e.g. spherical objective function to any (monotonic transformation of a) convex-quadratic objective function. Note that this introduces a bias on the log stepsize on uniformly random selection[1], whose expectation is 0 with the standard stepsize update.

Therefore, we analyze in total 10 variants of CMA-ES, as described here. We use the following labels to refer to them.

(a) The affine-invariant variant, i.e. when applying Eq. (8).
(s) The variant without cumulation on the stepsize, i.e. $c_\sigma = 1$.
(c) The variant without cumulation on the covariance matrix, i.e. $c_c = 1$.
(d) The continuously differentiable variant, i.e. when applying Eq. (7).

Moreover, we consider variants which are combinations of the previous ones, e.g., (ad) refers to the variant of CMA-ES applying Eqs. (7) and (8). Note that if we do not have cumulation, it is already affine-invariant, so we don't have to analyze variants that combine (a) with (s) and/or (c). Finally, the original CMA-ES is labeled (default).

[1] That is, choosing the indices $(i{:}\lambda)_{i=1,\ldots,\lambda}$ uniformly on the set of permutations of $\{1, \ldots, \lambda\}$

CPU timings per function evaluation (in 10^{-4}s)						
Dimensions	2	3	5	10	20	40
(default)-CMA-ES	2.8	2.4	2.0	2.3	2.6	3.0
a-CMA-ES	2.8	2.4	2.3	2.3	2.4	3.0
ad-CMA-ES	3.0	2.4	2.0	1.5	1.4	1.7
s-CMA-ES	2.4	2.2	1.9	1.5	1.4	1.7
c-CMA-ES	2.5	2.1	2.0	1.3	1.4	1.7
sc-CMA-ES	4.5	3.5	2.9	3.0	5.0	3.2
d-CMA-ES	3.1	2.6	2.1	1.6	1.5	1.8
sd-CMA-ES	3.0	2.4	2.1	1.5	1.4	1.8
cd-CMA-ES	2.9	2.3	1.8	1.4	1.4	1.7
scd-CMA-ES	2.6	2.4	1.9	1.5	1.3	1.8

Table 1: Average CPU timings for each variant

3 IMPLEMENTATION AND EXPERIMENTAL PROCEDURE

The code used for the benchmarking of the variants presented in this paper is available online [3]. This is based on the Python module of `cma`, and used options available in the `development – branch` which can also be found online [4]. To run the different variants we use respectively the following options in Python.

(a) `CSA_invariant_path = True`
(s) `es.adaptsigma.damps += 1 - es.adaptsigma.cs`
`es.adaptsigma.cs = 1`
(c) `es.sp.cc = 1`
(d) `CSA_squared = True`
`CSA_damp_mueff_exponent = 1`

The option `CSA_damp_mueff_exponent = 1` for the variant (d) is due to a wrong scaling of d_σ with $\mu_{\text{eff}} := \sum {w_i}^2$ when the stepsize update follows Eq. (7). We refer to e.g. Figure 1 where we see that the variant (d) with d_σ chosen as by default (labeled as (dm)) has lower performances, in particular on the Rastrigin function in higher dimensions. Note that the scaling of d_σ with μ_{eff} when using this update is not changed by default in v.3.3.0. of the `cma` module in Python [4]. The budget used for all experiments here was $2 \times 10^5 \times d$ function evaluations (where d is the dimension of the problem), with possibly up to 9 restarts with doubling the population size. We have used 15 instances, and analyzed 24 objective functions from `bbob` suite in dimensions 2, 3, 5, 10, 20, 40.

We performed the experiment using COCO [6] v.2.6.2, and post-processed the results (as shown in Section 5) with COCO v.2.6.3.

4 CPU TIMINGS

In order to evaluate the CPU timing of the algorithm, we have run each variant presented above with restarts on the entire `bbob` test suite for the entire budget. The Python code was run on a Linux machine with 32 cores, Intel ®Xeon ®E7 to E3 v4 processor, with Python 2.7.5. The time per function evaluation for dimensions 2, 3, 5, 10, 20, 40 for each variant is presented in Table 1.

5 RESULTS

Specific results for each algorithm variant presented in this paper are available online [3]. They notably include empirical cumulative distributions of the number of evaluations for a certain number of targets, expected runtimes measured in number of required function evaluations, and how they scale with dimension, on 24 objective functions of the bbob suite. We focus here on the main differences in the performances introduced by the modification presented in this paper w.r.t. the original CMA-ES.

Figure 5 shows how the expected runtime (measured in number of function evaluations) to reach given targets scales with the dimension of the problem when optimizing bbob functions with the default CMA-ES and the affine-invariant versions of CMA-ES (a) and (ad). We observe that (a) and (ad) performs well on many of these problems, in particular the sphere, Rastrigin, Weierstrass, Gallagher 101 peaks and Katsuuras. This is particularly significant for Katsuuras, where the affine-invariant variants solve up to three times more targets than the default, see Figure 3. Note that there is no remarkable performance loss for these variants, as on most objective functions, they perform about as fast as the default.

In Figure 5 are shown the scaling of the expected runtime with dimension for the modified CMA-ES algorithms without cumulation –(c), (s) and (sc)– compared to the default CMA-ES. The main differences are in favor of algorithms with cumulation on the covariance matrix especially on highly ill-conditioned problems, see Figure 4. This is particularly significant on the Bent cigar function, for which we show in Figure 2 the empirical cumulative distribution (ECDF) of the number of evaluations to reach several targets in dimension 40. We see there that variants with cumulation on the covariance matrix perform about twice faster in dimension 20 and thrice faster in dimension 40.

Figures 6 and 7 compare the algorithm variant (d), whose stepsize update is continuously differentiable, with the default CMA-ES. We can see that (d) performs slightly worse than the default on some functions, especially on high dimension, see e.g. Schwefel or Rastrigin separable, for which this is significant for some targets in larger dimensions. As explained in Section 3, Figure 1 shows a possible undesirable effect on the performance when we do not rescale the value of d_σ with μ_{eff} when using the variant (d).

All in all, when applying all modifications, the algorithm (scd) still performs well on most functions of the bbob suite, with the same observations that as for the variants (sc) and (d), with results presented in Figures 6 and 7 and Table 2. Note that, on most functions it performs as fast as the default CMA-ES. The highest differences of performances are, again, observed on ill-conditioned functions, see Figure 4, but it is never more than three times slower.

6 CONCLUSION / DISCUSSION

We have analyzed the effects of modifications of CMA-ES that allow for easier theoretical analysis. From the results obtained in this paper, we observe that these modifications do not deteriorate too much the performances. Specifically, it appears that cumulation on the covariance matrix helps significantly on ill-conditioned problems, and that the standard CSA update for the stepsize performs slightly better than the continuously differentiable alternative we introduce in this paper. We do not observe any major negative effects of the modification on the paths introduced here to make CMA-ES affine-invariant.

We also note that the value of d_σ, when the stepsize update follows Eq. (7), is not correctly chosen in the version of the cma Python package used when running the experiment (v.3.3.0), and it is currently required to change the scaling of d_σ with μ_{eff} when using this variant.

REFERENCES

[1] Anne Auger. Analysis of comparison-based stochastic continuous black-box optimization algorithms. *"Habilitation à diriger les recherches" dissertation, Université Paris-Sud*, 2015.

[2] Alexandre Chotard and Anne Auger. Verifiable conditions for the irreducibility and aperiodicity of markov chains by analyzing underlying deterministic models. *Bernoulli*, 25(1):112–147, 2019.

[3] Armand Gissler. Github repository. https://github.com/agissler/Benchmarking_proof_variants_CMA-ES/.

[4] Nikolaus Hansen. pycma. https://github.com/CMA-ES/pycma/.

[5] Nikolaus Hansen. The CMA evolution strategy: a comparing review. *Towards a new evolutionary computation: Advances in the estimation of distribution algorithms*, pages 75–102, 2006.

[6] Nikolaus Hansen, Anne Auger, Raymond Ros, Olaf Mersmann, Tea Tušar, and Dimo Brockhoff. COCO: A platform for comparing continuous optimizers in a black-box setting. *Optimization Methods and Software*, 36(1):114–144, 2021.

[7] Nikolaus Hansen, Sibylle D Müller, and Petros Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary computation*, 11(1):1–18, 2003.

[8] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–195, 2001.

[9] Mohamed Jebalia, Anne Auger, and Nikolaus Hansen. Log-linear convergence and divergence of the scale-invariant (1+1)-ES in noisy environments. *Algorithmica*, 59(3):425–460, 2011.

[10] Sean P Meyn and Richard L Tweedie. *Markov chains and stochastic stability*. Springer Science & Business Media, 2012.

[11] Cheikh Touré, Anne Auger, and Nikolaus Hansen. Global linear convergence of evolution strategies with recombination on scaling-invariant functions. *Journal of Global Optimization*, pages 1–41, 2022.

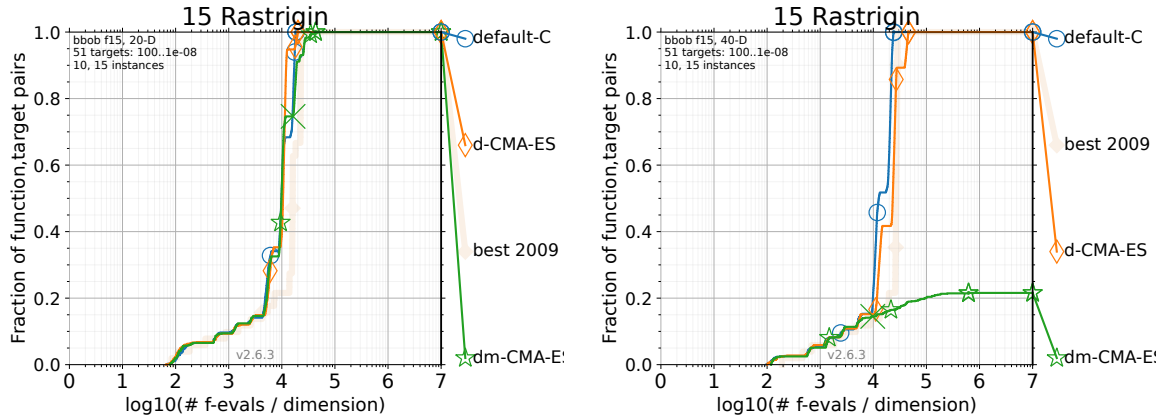

Figure 1: Empirical cumulative distribution of the number of evaluations for 51 targets on the Rastrigin (left) and the Bent cigar (right) objective function in dimensions 20 (left) and 40 (right). In both cases the budget used is $2 \times 10^5 \times$ dimension, and the variant (dm) does not use its full budget as it can diverge

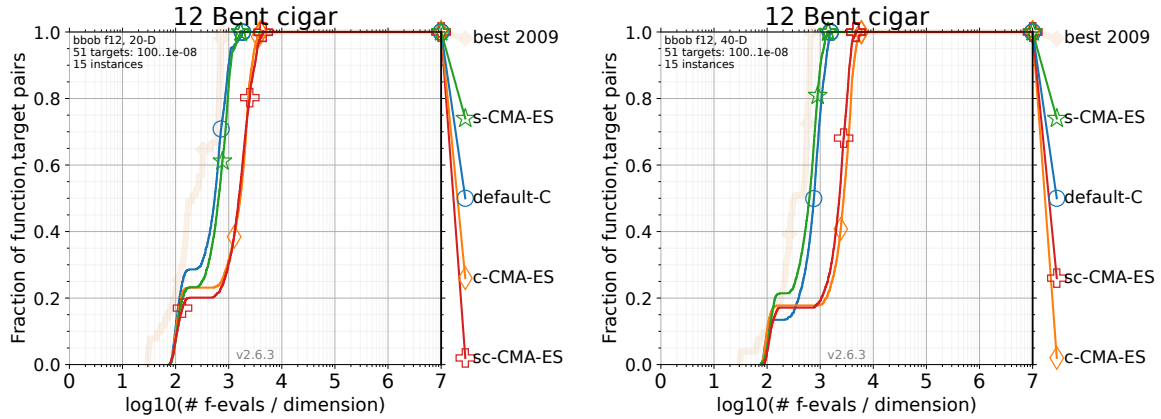

Figure 2: Empirical cumulative distribution of the number of evaluations for 51 targets on the Bent cigar objective function in dimensions 20 (left) and 40 (right). In both cases the budget used is $2 \times 10^5 \times$ dimension

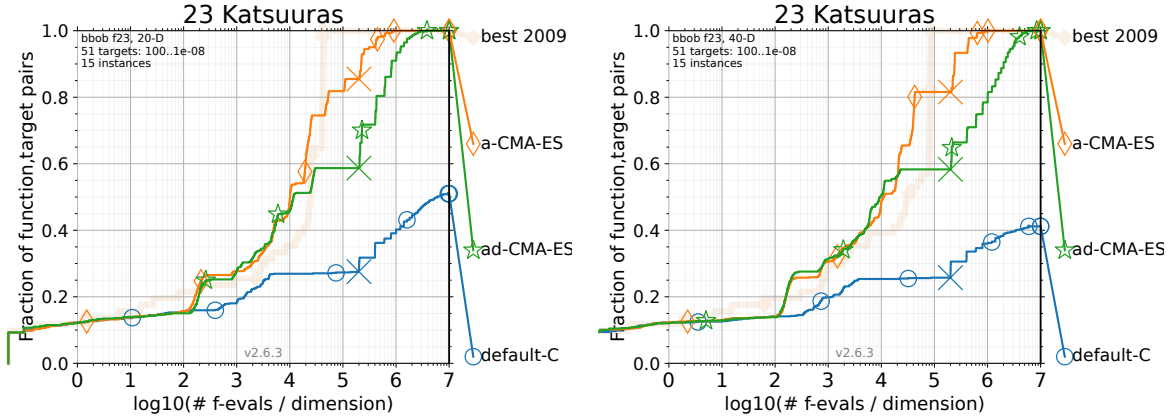

Figure 3: Empirical cumulative distribution of the number of evaluations for 51 targets on the Katsuuras objective function in dimensions 20 (left) and 40 (right). In both cases the budget used is $2 \times 10^5 \times$ dimension

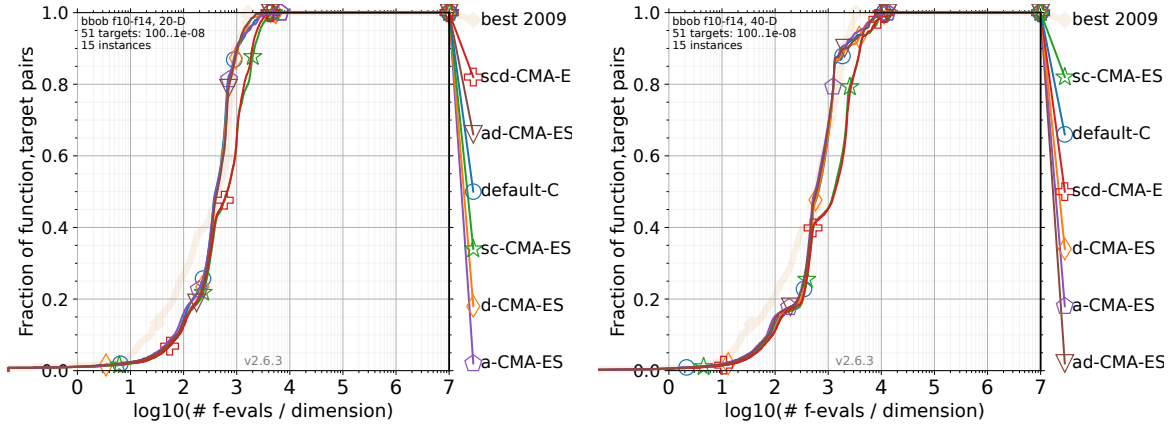

Figure 4: Empirical cumulative distribution of the number of evaluations for 51 targets on the highly ill-conditioned objective functions group in dimensions 20 (left) and 40 (right). In both cases the budget used is $2 \times 10^5 \times$ dimension

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f1	43	43	43	43	43	43	43	15/15
default	**6.8**(1.0)	**13**(1)	**19**(1)	**25**(2)	**32**(2)$^{\star 2}$	**45**(2)$^{\star 2}$	**57**(3)$^{\star 3}$	15/15
scd-CMA-ES	6.9(2)	15(5)	22(6)	30(5)	38(6)	53(7)	69(7)	15/15
f2	385	386	387	388	390	391	393	15/15
default	**21**(3)$^{\star 4}$	**25**(3)$^{\star 4}$	**28**(3)$^{\star 4}$	**30**(3)$^{\star 4}$	**31**(2)$^{\star 4}$	**32**(2)$^{\star 4}$	**34**(2)$^{\star 4}$	15/15
scd-CMA-ES	32(5)	39(6)	44(4)	47(4)	49(3)	51(4)	52(4)	15/15
f3	5066	7626	7635	7637	7643	7646	7651	15/15
default	**7.9**(8)	∞	∞	∞	∞	∞	∞ *3e6*	0/15
scd-CMA-ES	14(9)	∞	∞	∞	∞	∞	∞ *2e6*	0/15
f4	4722	7628	7666	7686	7700	7758	1.4e5	9/15
default	∞	∞	∞	∞	∞	∞	∞ *2e6*	0/15
scd-CMA-ES	∞	∞	∞	∞	∞	∞	∞ *2e6*	0/15
f5	41	41	41	41	41	41	41	15/15
default	**4.9**(1)	**5.6**(2)	**5.8**(1)	**5.8**(1)	**5.8**(1)	**5.8**(1)	**5.8**(1)	15/15
scd-CMA-ES	5.0(1)	6.3(2)	6.5(1)	6.5(1)	6.5(1)	6.5(1)	6.5(1)	15/15
f6	1296	2343	3413	4255	5220	6728	8409	15/15
default	1.4(0.2)	**1.2**(0.1)	**1.0**(0.1)	**1.0**(0.1)	**1.0**(0.1)	**1.0**(0.1)	**1.1**(0.1)	15/15
scd-CMA-ES	**1.4**(0.2)	1.2(0.1)	1.1(0.1)	1.1(0.1)	1.1(0.1)	1.1(0.1)	1.1(0.1)	15/15
f7	1351	4274	9503	16523	16524	16524	16969	15/15
default	1.0(1)	**2.5**(2)	**1.6**(0.6)*	**0.98**(0.3)$^{\star 2}$	**0.98**(0.3)$^{\star 2}$	**0.98**(0.3)$^{\star 2}$	**0.97**(0.3)$^{\star 2}$	15/15
scd-CMA-ES	**0.93**(0.3)	3.1(1)	2.2(0.6)	1.4(0.4)	1.4(0.4)	1.4(0.4)	1.4(0.4)	15/15
f8	2039	3871	4040	4148	4219	4371	4484	15/15
default	**3.3**(0.9)$^{\star 3}$	**4.3**(3)	**4.5**(3)	**4.5**(3)	**4.6**(3)	**4.6**(3)	**4.7**(3)	15/15
scd-CMA-ES	5.8(2)	7.4(4)	7.8(4)	8.1(4)	8.2(4)	8.2(4)	8.1(3)	15/15
f9	1716	3102	3277	3379	3455	3594	3727	15/15
default	**3.8**(0.9)$^{\star 3}$	**5.3**(4)	**5.6**(4)	**5.6**(3)	**5.7**(3)	**5.6**(3)	**5.6**(3)	15/15
scd-CMA-ES	6.0(2)	8.0(5)	8.6(5)	8.9(5)	9.0(5)	8.9(5)	8.8(4)	15/15
f10	7413	8661	10735	13641	14920	17073	17476	15/15
default	**1.1**(0.2)$^{\star 4}$	**1.1**(0.2)$^{\star 4}$	**1.00**(0.1)$^{\star 4}$	**0.83**(0.1)$^{\star 4}_{\uparrow 0}$	**0.79**(0.1)$^{\star 4}_{\uparrow 0}$	**0.73**(0.0)$^{\star 4}_{\uparrow 0}$	**0.75**(0.0)$^{\star 4}_{\uparrow 0}$	15/15
scd-CMA-ES	1.7(0.2)	1.8(0.2)	1.6(0.2)	1.4(0.1)	1.3(0.1)	1.2(0.1)	1.2(0.1)	15/15
f11	1002	2228	6278	8586	9762	12285	14831	15/15
default	**4.4**(0.4)	**2.2**(0.1)	**0.85**(0.1)$_{\uparrow 0}$	**0.67**(0.0)$_{\uparrow 0}$	**0.62**(0.0)$_{\uparrow 0}$	**0.55**(0.0)$_{\uparrow 0}$	**0.49**(0.0)$_{\uparrow 0}$	15/15
scd-CMA-ES	4.5(0.4)	2.3(0.2)	0.88(0.1)$_{\uparrow 1}$	0.69(0.0)$_{\uparrow 0}$	0.64(0.0)$_{\uparrow 0}$	0.56(0.0)$_{\uparrow 0}$	0.52(0.0)$_{\uparrow 0}$	15/15
f12	1042	1938	2740	3156	4140	12407	13827	15/15
default	**2.8**(3)	**2.2**(2)	**2.6**(2)	**2.9**(2)*	**2.8**(1)$^{\star 3}$	**1.2**(0.4)$^{\star 3}$	**1.3**(0.3)$^{\star 3}$	15/15
scd-CMA-ES	4.1(5)	5.7(7)	7.1(6)	7.7(6)	7.2(4)	3.1(1)	3.2(0.9)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f13	652	2021	2751	3507	18749	24455	30201	15/15
default	**2.3**(0.3)	**2.7**(2)	**4.0**(3)*	**4.0**(2)*	**1.0**(0.6)	**1.2**(0.6)	**1.4**(0.5)	15/15
scd-CMA-ES	4.4(7)	6.2(5)	7.6(4)	7.0(5)	1.5(0.6)	1.6(0.4)	1.6(0.5)	15/15
f14	75	239	304	451	932	1648	15661	15/15
default	**2.5**(1)	**2.3**(0.3)	**3.2**(0.5)	**3.6**(0.4)	**3.1**(0.4)	**3.7**(0.3)$^{\star 4}$	**0.67**(0.1)$^{\star 4}_{\uparrow 0}$	15/15
scd-CMA-ES	3.4(2)	2.7(0.6)	3.6(0.8)	4.0(0.7)	3.4(0.5)	4.8(0.4)	0.89(0.1)$_{\uparrow 0}$	15/15
f15	30378	1.5e5	3.1e5	3.2e5	3.2e5	4.5e5	4.6e5	15/15
default	**0.75**(0.6)*	**1.2**(0.4)	**0.71**(0.4)	**0.71**(0.4)	**0.72**(0.4)	**0.53**(0.3)$_{\uparrow 0}$	**0.54**(0.3)$_{\uparrow 0}$	15/15
scd-CMA-ES	1.3(0.5)	1.4(0.4)	0.77(0.4)	0.78(0.4)	0.79(0.4)	0.58(0.3)$_{\uparrow 0}$	0.59(0.3)$_{\uparrow 0}$	15/15
f16	1384	27265	77015	1.4e5	1.9e5	2.0e5	2.2e5	15/15
default	**1.9**(0.8)	**0.65**(0.6)	**0.65**(0.4)$_{\uparrow 1}$	0.81(0.8)	**1.2**(1)	**1.1**(1)	**1.1**(1)	15/15
scd-CMA-ES	2.2(0.9)	0.86(0.4)	0.94(0.6)	**0.78**(0.5)	1.4(1)	1.5(1)	1.4(1)	15/15
f17	63	1030	4005	12242	30677	56288	80472	15/15
default	**1.2**(1)	**0.96**(0.4)	**1.2**(2)	**0.98**(0.7)	**0.73**(0.3)	**0.88**(0.3)	**0.85**(0.2)	15/15
scd-CMA-ES	1.5(1)	1.0(0.2)	1.8(1)	1.1(0.6)	0.89(0.4)	0.93(0.4)	0.90(0.3)	15/15
f18	621	3972	19561	28555	67569	1.3e5	1.5e5	15/15
default	**0.86**(0.3)	**0.73**(0.1)$_{\uparrow 1}$	0.87(0.8)	1.3(0.8)	0.86(0.3)	0.82(0.4)	**0.79**(0.3)	15/15
scd-CMA-ES	0.97(0.3)	1.5(2)	**0.75**(0.4)	**1.3**(0.4)	**0.75**(0.3)	**0.74**(0.6)	0.89(0.6)	15/15
f19	1	1	3.4e5	4.7e6	6.2e6	6.7e6	6.7e6	15/15
default	**1**(0)	**1**(0)	**1.1**(0.9)	**0.48**(0.2)$^{\star 2}_{\uparrow 0}$	**0.54**(0.4)*	**0.59**(0.4)*	**0.59**(0.3)*	12/15
scd-CMA-ES	**1**(0)	**1**(0)	1.5(0.8)	1.2(0.8)	1.3(1)	1.8(2)	2.2(2)	3/15
f20	82	46150	3.1e6	5.5e6	5.5e6	5.6e6	5.6e6	14/15
default	**3.8**(0.9)	**4.0**(1)	**1.0**(1)$^{\star 3}$	**1.5**(2)	**1.5**(1)	**1.7**(2)	**1.7**(1)	5/15
scd-CMA-ES	5.0(1)	4.3(1)	∞	∞	∞	∞	∞ *3e6*	0/15
f21	561	6541	14103	14318	14643	15567	17589	15/15
default	**1.6**(4)	115(166)	**74**(74)	**73**(100)	**71**(80)	**67**(110)	**60**(108)	8/15
scd-CMA-ES	2.6(5)	**90**(100)	118(167)	116(137)	114(164)	107(126)	95(112)	6/15
f22	467	5580	23491	24163	24948	26847	1.3e5	12/15
default	171(24)	**141**(218)	∞	∞	∞	∞	∞ *1e6*	0/15
scd-CMA-ES	**8.4**(14)	141(216)	∞	∞	∞	∞	∞ *1e6*	0/15
f23	3.0	1614	67457	3.7e5	4.9e5	8.1e5	8.4e5	15/15
default	1.9(2)	1924(1872)	90(104)	**154**(175)	**115**(203)	∞	∞ *4e6*	0/15
scd-CMA-ES	**1.3**(1)	**391**(1240)	**53**(74)	∞	∞	∞	∞ *4e6*	0/15
f24	1.3e6	7.5e6	5.2e7	5.2e7	5.2e7	5.2e7	5.2e7	3/15
default	**4.5**(3)	7.6(12)	∞	∞	∞	∞	∞ *4e6*	0/15
scd-CMA-ES	7.1(6)	**2.4**(2)	∞	∞	∞	∞	∞ *4e6*	0/15

Table 2: Expected runtime (ERT in number of f-evaluations) divided by the respective best ERT measured during BBOB-2009 in dimension 20. This ERT ratio and, in braces as dispersion measure, the half difference between 10 and 90run lengths appear for each algorithm and target, the corresponding reference ERT in the first row. The different target Δf-values are shown in the top row. #succ is the number of trials that reached the (final) 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). A ↓ indicates the same tested against the best algorithm from BBOB 2009. Best results are printed in bold. Data produced with COCO v2.6.2

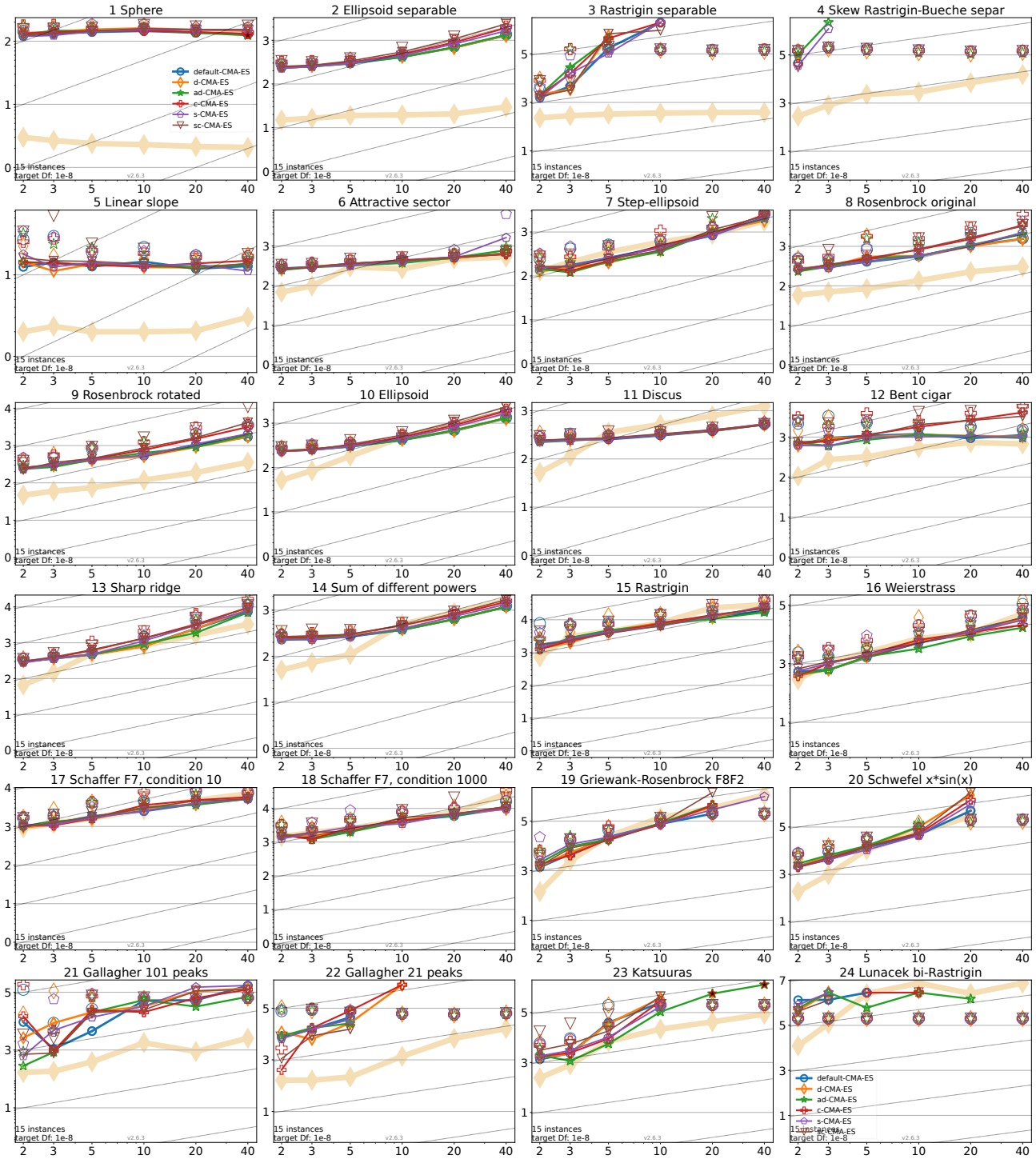

Figure 5: Scaling of expected runtime (measured in function evaluations) with dimension: on the x-axis the dimension of the problem, on the y-axis $\log_{10}(\#\{\text{function evaluations}\}/\text{dimension})$

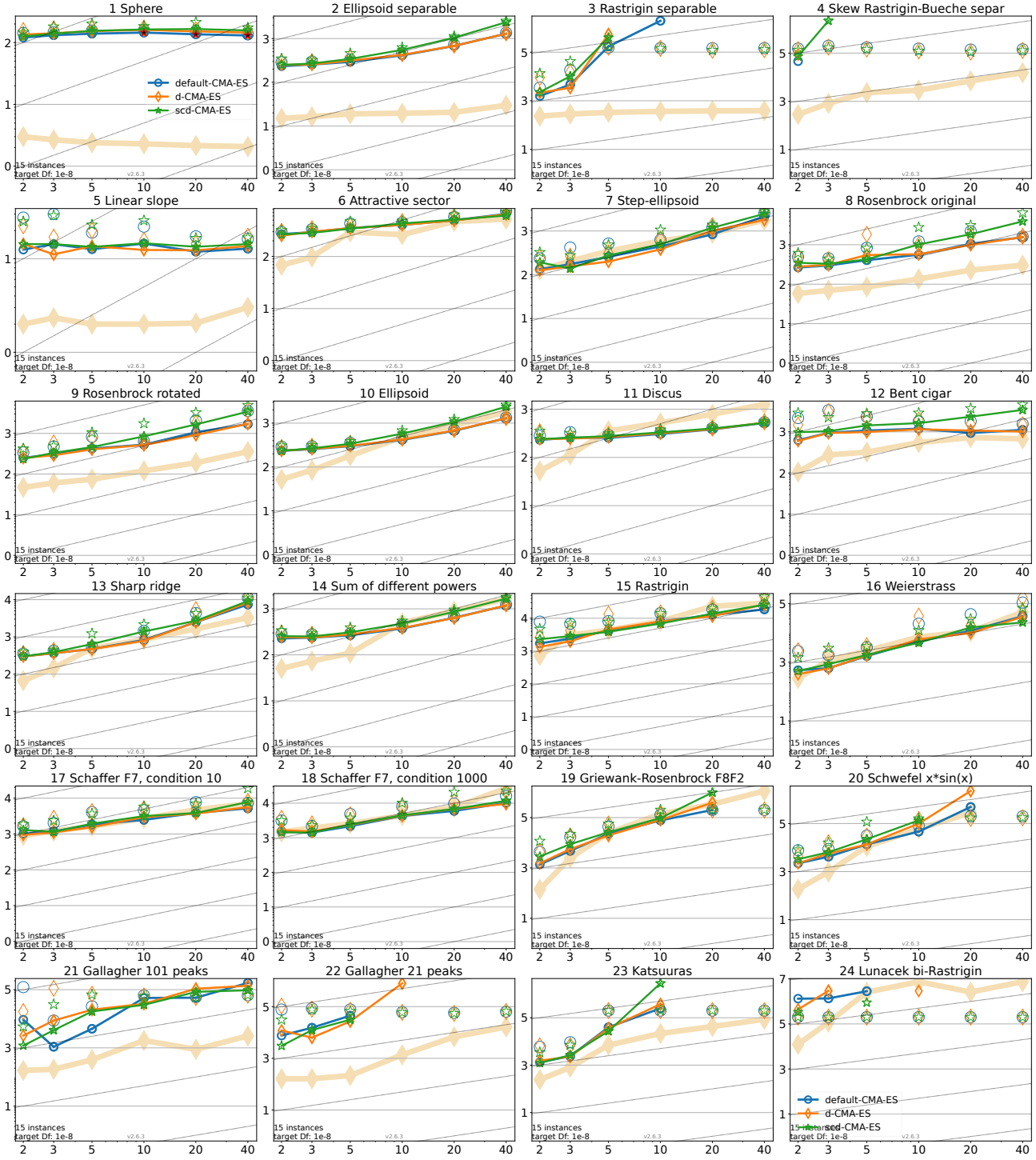

Figure 6: Scaling of expected runtime (measured in function evaluations) with dimension: on the x-axis the dimension of the problem, on the y-axis $\log_{10}(\#\{\text{function evaluations}\}/\text{dimension})$

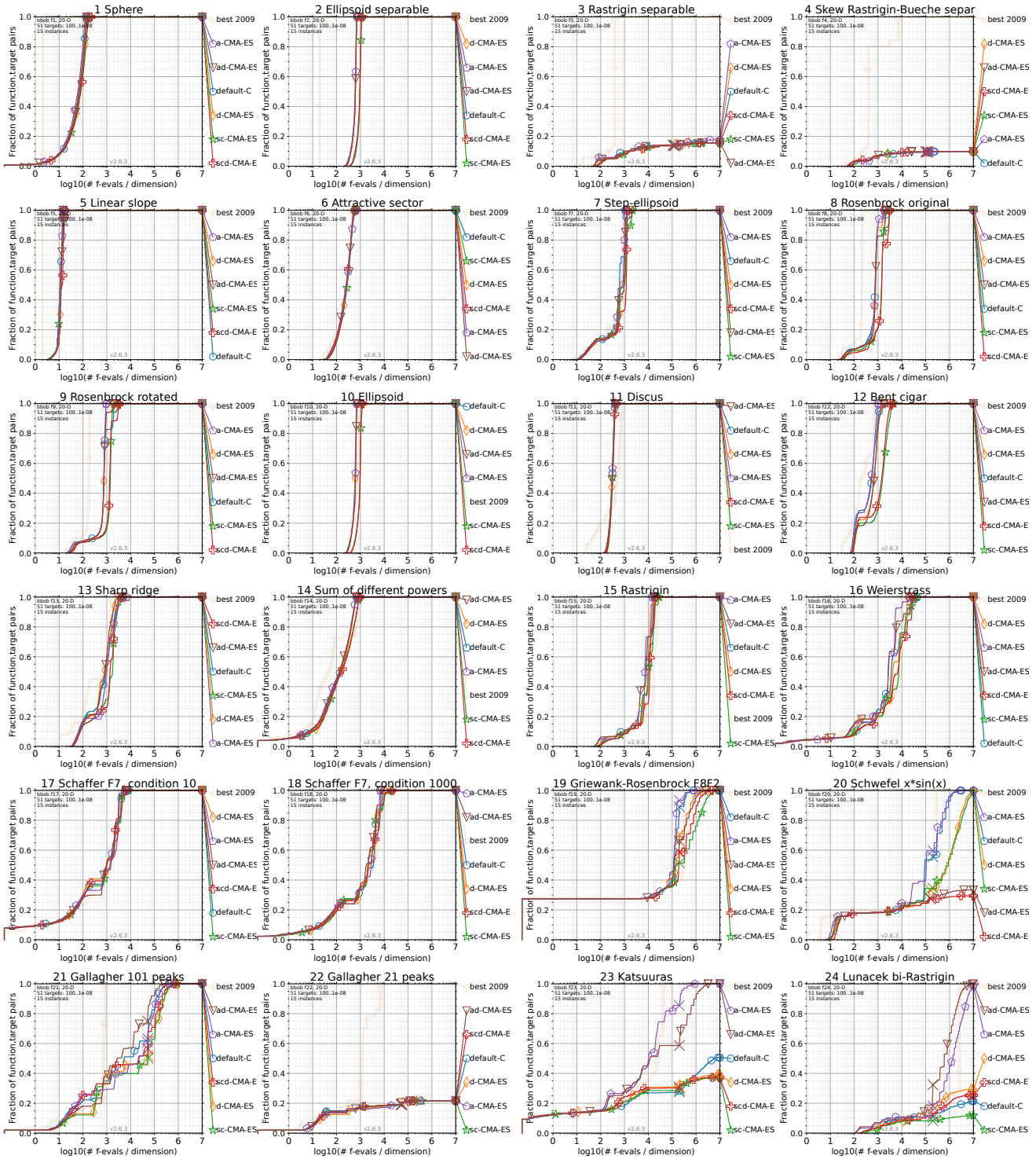

Figure 7: Empirical cumulative distribution of simulated (bootstrapped) runtimes, measured in number of f-evaluations, divided by dimension for 51 targets $10^{[-8..2]}$ in 20D.