Comparing Boundary Handling Techniques of CMA-ES on the bbob and sbox-cost Test Suites

Dimo Brockhoff

▶ **To cite this version:**

Dimo Brockhoff. Comparing Boundary Handling Techniques of CMA-ES on the bbob and sbox-cost Test Suites. GECCO '23 Companion: Companion Conference on Genetic and Evolutionary Computation, Jul 2023, Lisbon, Portugal. pp.2318-2325, 10.1145/3583133.3596413 . hal-04403658

HAL Id: hal-04403658

https://inria.hal.science/hal-04403658

Submitted on 18 Jan 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Comparing Boundary Handling Techniques of CMA-ES on the bbob and sbox-cost Test Suites

Dimo Brockhoff
Inria, CMAP, CNRS, Ecole Polytechnique, Institut Polytechnique de Paris
91120 Palaiseau, France

ABSTRACT

Bound constraints on the variables are the most basic constraints in an optimization problem formulation and, thus, among the most common. It is therefore essential to understand the impact of different boundary handling techniques on algorithm performance. Equally, it is important to understand the practical impact of using bound constraint handling in an algorithm on principally unbounded problems but where the user has a good indication of the domain of the (sought) optimum. Both questions will be investigated in this paper on the newly introduced box-constrained version `sbox-cost` of the well-known, unconstrained test suite `bbob` and for the example of the two boundary handling techniques, implemented in the CMA-ES python module `pycma`. The numerical experiments performed with the COCO platform show that there is (i) only a minor difference in performance between the two test suites and (ii) a slight performance reduction for the (default) `BoundTransform` boundary handling compared to the `BoundPenalty` version of CMA-ES.

CCS CONCEPTS

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

KEYWORDS

Benchmarking, black-box optimization, bound constraints

ACM Reference format:
Dimo Brockhoff. 2023. Comparing Boundary Handling Techniques of CMA-ES on the `bbob` and `sbox-cost` Test Suites. In *Proceedings of Genetic and Evolutionary Computation Conference Companion, Lisbon, Portugal, July 15–19, 2023 (GECCO '23 Companion),* 8 pages.
DOI: 10.1145/3583133.3596413

1 INTRODUCTION

Very often, real-world optimization problems have bound constraints on their variables, i.e., the variables are restricted to stay within (partly open) intervals. For those cases, optimization algorithms have to provide boundary handling options that restrict the sampling of solutions to the feasible domain. The well-known CMA-ES [7], implemented in the python module `pycma`, has two of such options: the default `BoundTransform` and the `BoundPenalty` option.

In this paper, we investigate the performance of those two boundary handling techniques on the recently introduced `sbox-cost` test suite—a slightly modified version of the well-known, unconstrained `bbob` suite that returns infinity as function value outside of the region $[-5,5]^n$ with n being the search space dimension and thus imitating bound constraints on the variables. The `sbox-cost` test suite furthermore changes the instances of the original `bbob` suite by placing the optima (pseudo-)randomly within the entire feasible space, compared to the interval $[-4,4]^n$ for the `bbob` suite. The only exception is the linear slope function f_5 in which the optimum within $[-5,5]^n$ stays on one of the corners in both suites. [1] Given that the optima lie within the search domain $[-5,5]^n$ (with the exception of the mentioned linear slope function), we do not expect any (large) performance difference between algorithms as long as the search is local and does not sample outside of the feasible domain.

Numerical experiments on the COCO implementation of the test suites [4] support this. However, some unexpected performance differences between the two test suites and between the two boundary handling methods within CMA-ES can be observed on a few test functions as we will see later on—ultimately resulting in the need to further investigate the reason for those differences in future work.

2 SBOX-COST VS. BBOB

We already mentioned the main differences between the original `bbob` test suite and its bound constrained `sbox-cost` counterpart: unconstrained vs. bound constrained to $[-5,5]^n$ and the optima being sampled within $[-4,4]^n$ vs. within $[-5,5]^n$ (with the exception of the linear slope function where there is no difference within $[-5,5]^n$). Both suites are implemented within the COCO platform [4] and results can be easily compared with its postprocessing module. When doing so, we use 15 instances and the default 51 target precisions $100,\ldots,10^{-8}$ prescribed for both the `bbob` and the `sbox-cost` test suite.

3 ALGORITHM PRESENTATION

The Covariance Matrix Adaptation Evolution Strategy (CMA-ES, [7]) is a state-of-the-art solver for unconstrained, nonlinear black-box optimization. It samples, in each iteration, λ solutions from a multivariate Normal distribution which is adapted throughout

©Dimo Brockhoff 2023. This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in GECCO'23 Companion, http://dx.doi.org/10.1145/3583133.3596413.
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
GECCO '23 Companion, Lisbon, Portugal
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
979-8-4007-0120-7/23/07…$15.00
DOI: 10.1145/3583133.3596413

[1] Note that with the unboundedness of the `bbob` suite, the optimal function value can furthermore be reached in a subspace of $\mathbb{R}^n$ outside the box $[-5,5]^n$ in `bbob` (with a flat objective function value) while the single search point with optimal function value lies in a corner of the bounded search space for `sbox-cost`.

the run in terms of its mean, stepsize, and covariance matrix, hence the name. Its python implementation offers two different boundary handling techniques, the default `BoundTransform` and the `BoundPenalty`. In the following, we assume that both lower and upper bounds on the variables are given by the (feasibility) interval $[\mathrm{lb}, \mathrm{ub}]$ for each variable.[2]

`BoundTransform`. The `BoundTransform` is CMA-ES' default technique to handle bound constraints, mapping an arbitrary variable value x_i, periodically back to the feasible interval $[\mathrm{lb}, \mathrm{ub}]$.

The transformation is thereby the identity in about 90% of the originally feasible interval (i.e., within $[\mathrm{lb} + \mathrm{al}, \mathrm{ub} - \mathrm{au}]$ for a choice of al and au that, in practice, depend on the absolute values of lb and ub). The transformation is quadratic in the intervals $[\mathrm{lb} - 3 \cdot \mathrm{al}, \mathrm{lb} + \mathrm{al}]$ and $[\mathrm{ub} - \mathrm{au}, \mathrm{ub} + 3 \cdot \mathrm{au}]$ and "is periodically expanded beyond the limits [...] with a period of $2 \cdot (\mathrm{ub} - \mathrm{lb} + \mathrm{al} + \mathrm{au})$" (see the pycma API documentation here and here). This construction allows for a smooth transformation close to the domain boundary and a robust handling of "arbitrary" bound values.

In the remainder of this paper, we denote the version of CMA-ES, using the `BoundTransform` boundary handling, as `CMA-bt`.

`BoundPenalty`. With this boundary handling method, CMA-ES evaluates an infeasible solution x as

$$f(x^{\mathrm{feas}}) + \sum w_i \cdot (x_i - x_i^{\mathrm{feas}})^2 \tag{1}$$

where x^{feas} is the feasible (in-bounds) solution that is closest to x (in Euclidean space) and the weights w_i for each variable i are updated in each iteration. For more details we refer to section IV.B in [6].

In the following, we denote this version of CMA-ES as `CMA-bp`.

4 EXPERIMENTAL PROCEDURE

We compare both `CMA-bt` and `CMA-bp` on the two test suites `bbob` and `sbox-cost` as mentioned. The experiments are performed with COCO [4], in its version from the `suite-sbox-cost` branch of the `sbox-cost` organization at Github in its version from the 13th of February 2023. The plots are produced with an adapted version of COCO 2.6.3.[3] To distinguish the experiments on the two test suites from each other, the name of the test suite is appended to the algorithm name of the data sets in the plots. For better readability, the prefix "cma-" is removed in the tables. For the CMA-ES algorithm, we used the python module `pycma`, version 3.3.0.

Initial values for the CMA-ES variants (besides defaults) are an initial step size of 2, the search space origin as initial mean in the first run and a maximum of 9 restarts. The initial mean for a restart is chosen according to the `initial_solution_proposal` functionality of COCO. All output from CMA-ES is suppressed for speed (and lower environmental impact of the experiment) and the bounds are chosen as -5 and $+5$ respectively for each variable. The length of a single experiment is bounded above by 10^4 times dimension many function evaluations and stopped as soon as this amount of function evaluations is reached or if COCO detected a function value within a precision of 10^{-8} from the optimal function value and, in addition, any of the many (default) CMA-ES-internal stopping criteria is reached.

The **expected runtime (**ERT**)**, used in the figures and tables below, 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 [3, 9]. **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. For a detailed explanation of the COCO plots, please refer to the COCO documentation [4].

Parameter Tuning. No parameter tuning has been performed before the experiments.

Environmental Impact. Including all preliminary experiments (for smaller budgets but with essentially the same setup), we used less than 64 CPU hours of computing time overall. With a maximal power consumption of 120W for the used processor type, this results in less than 8kWh (plus cooling).

5 CPU TIMING

In order to evaluate the CPU timing of the algorithm, we have run the different CMA-ES versions with maximally nine restarts on the entire `bbob` [5] and `sbox-cost` test suites for $10^4 n$ function evaluations according to [8]. The Python code was run on a Linux machine with 32 Intel Xeon E5-2683 v4 processors with 2.10GHz and python 3.7.4 but without using the batch parallelization code of COCO. The (average) time per function evaluation for dimensions 2, 3, 5, 10, and 20 on the `bbob` suite equaled 0.14, 0.15, 0.16, 0.17, and 0.19 milliseconds respectively for the CMA-ES with default `BoundTransform` and 0.15, 0.16, 0.18, 0.21, and 0.26 milliseconds for the version with `BoundPenalty` boundary handling. For the `sbox-cost` experiments, the timings were slightly faster: 0.14, 0.14, 0.15, 0.17, and 0.19 milliseconds for the CMA-ES with `BoundTransform` and 0.13, 0.13, 0.16, 0.19, and 0.23 milliseconds for the `BoundPenalty` version.

6 RESULTS

Results from experiments according to [8] and [2] on the benchmark functions given in [1] are presented in Figures 1, 2, 3, and 4 and Tables 1 and 2. In addition to the displayed results in this paper, we provide all postprocessed results together with the original data, in particular for different dimensions, at the supplementary material webpage[4].

With the two algorithm variants, ran on two test suites, also two main investigations can be made: on the impact of the different instance sampling on problem difficulty and on the impact on algorithm performance of the boundary handling itself.

[2] Of course, the code allows to specify different bounds and to also omit one of the two or both bounds independently for each variable.

[3] Originally, the two test suites could not be postprocessed together although they are, in principle, compatible with each other.

[4] see https://brockho.github.io/pycmaonsboxcost/

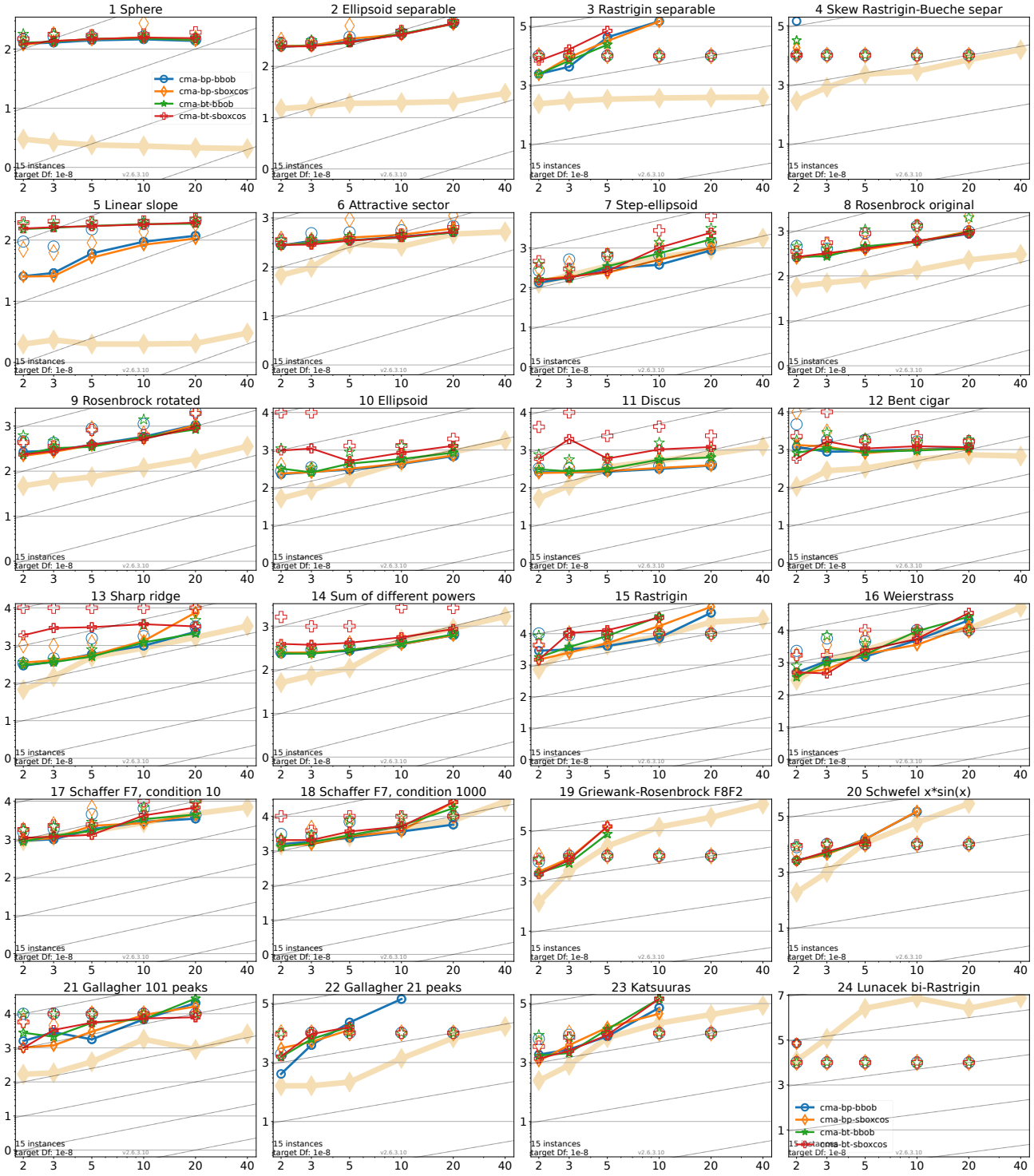

Figure 1: Expected running time (ERT in number of f-evaluations as $\log_{10}$ value), divided by dimension for target function value 10^{-8} versus dimension. Slanted grid lines indicate quadratic scaling with the dimension. Different symbols correspond to different algorithms given in the legend of f_1 and f_{24}. Light symbols give the maximum number of evaluations from the longest trial divided by dimension. Black stars (if present) indicate a statistically better result compared to all other algorithms with $p < 0.01$ and Bonferroni correction number of dimensions (six). The ERT of the (artificial) best algorithm from BBOB 2009 is shown as thick line with diamond markers. Legend: ○: cma-bp-sboxcost-Brockhoff, ◇: cma-bt-sboxcost-Brockhoff, ⋆: cma-bt-bbob, +: cma-bt-sboxcost.

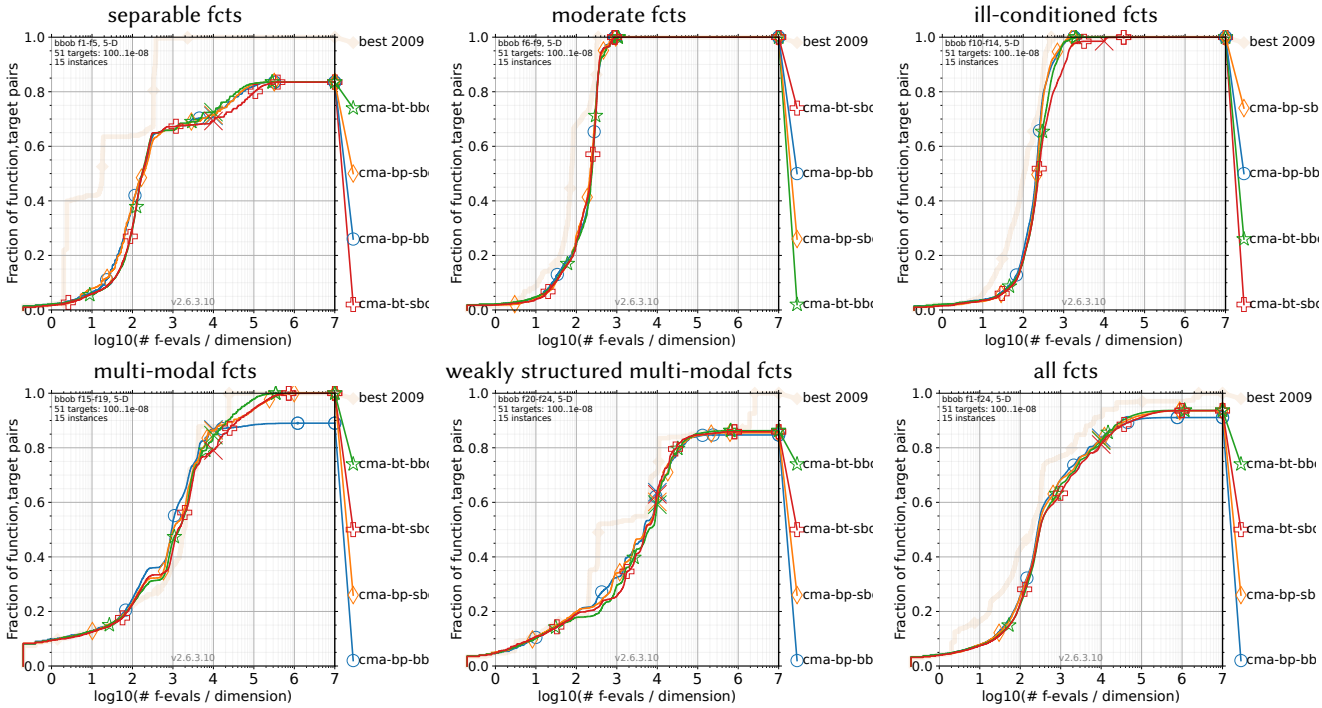

Figure 2: Bootstrapped empirical cumulative distribution of the number of f-evaluations divided by dimension (FEvals/DIM) for 51 targets with target precision in $10^{[-8..2]}$ for all functions and subgroups in 5-D. As reference algorithm, the best algorithm from BBOB 2009 is shown as light thick line with diamond markers.

6.1 Impact of bound constraints: bbob vs. sbox-cost

Let us first have a look at the question whether the addition of bound constraints and the changed optima sampling makes the test problems harder to solve. Of course, here, this question can only be answered with respect to the CMA-ES but preliminary experiments showed globally the same observation for a simple random search in $[-5, 5]^n$. Over all functions, we see only slight differences for each CMA-ES version between the two test suites with a difference of maximally 50% for budgets larger than $500n$ function evaluations in 5-D (Figure 2) and with slightly larger gaps for the larger budgets in 20-D (Figure 3). For the easiest targets and thus the smallest budgets (for example below $100n$ function evaluations in 20-D) no difference in the ECDF plot is visible.

When looking at the single functions, we observe that (significant) differences appear only on few functions. The largest differences can be observed for `CMA-bt` on the Rastrigin (in low dimension), Sharp Ridge, non-separable Ellipsoid, Discus, Rosenbrock, Gallagher, and Weierstrass functions, and to a lesser extend on the Step Ellipsoid and for `CMA-bp` on the non-separable Rastrigin function, Sharp Ridge, Rosenbrock (in the end) and on the multimodal Schaffer and Gallagher functions. Most of the time, the `bbob` versions are easier to solve but some examples with the opposite observation exists. The most consistent differences over all dimensions are observed on the non-separable Ellipsoid and Discus functions for `CMA-bt` and on the Sharp Ridge and Rastrigin functions for `CMA-bp`. Most of the observed differences are not statistically significant. The only exceptions are two target precisions on the sphere function for `CMA-bt` (one in dimension 10 and one in dimension 20) and the harder targets for the non-separable Ellipsoid and Discus functions in 20-D for `CMA-bp`—in all four cases in favor of the `bbob` function (for the results of the statistical tests, we refer to the supplementary material).

6.2 Impact of boundary handling in CMA-ES

A larger effect than the definition of the functions can be observed when the two boundary handling techniques are compared with each other. While aggregated over all functions also no large differences occur on the `bbob` test suite, the `BoundPenalty` shows slight advantages over the `BoundTransform` in particular for larger budgets and in higher dimension. The tendency towards a slightly better performance of `BoundPenalty` over `BoundTransform` is the same on the `sbox-cost` test suite but with less clarity in terms of a dependency on budget and/or dimension.

On the single functions, no visual differences occur on 11 out of the 24 functions (in 10-D and 20-D). On the other functions, it is mostly the `BoundPenalty` version outperforming the variant with `BoundTransform`, but exceptions (in some dimensions) exist. Worth to mention are the differences on the linear slope function (statistically significantly better performance for `BoundPenalty` on both test suites for the harder targets in all dimensions). The differences are more pronounced in smaller dimension with a factor of larger than 5 in dimension 2 (and, in comparison, a factor

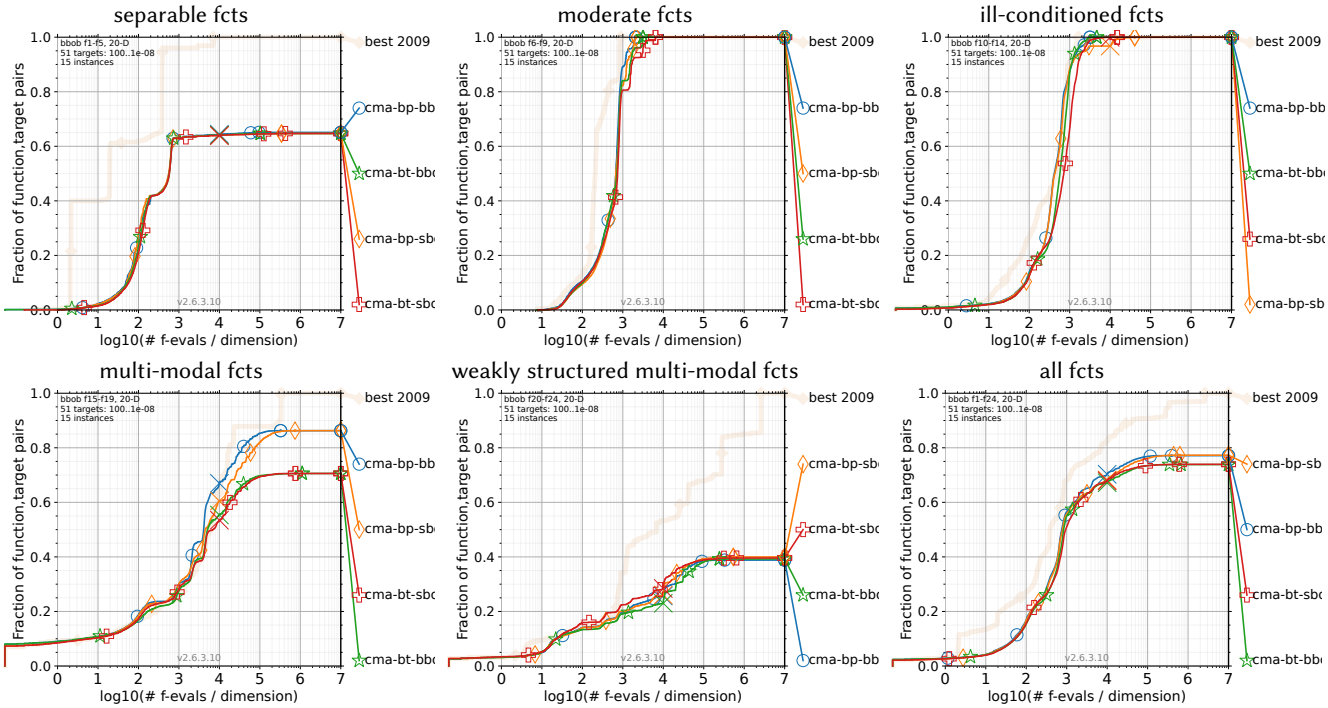

Figure 3: Bootstrapped empirical cumulative distribution of the number of f-evaluations divided by dimension (FEvals/DIM) for 51 targets with target precision in $10^{[-8..2]}$ for all functions and subgroups in 20-D. As reference algorithm, the best algorithm from BBOB 2009 is shown as light thick line with diamond markers.

of around 1.5 in 20-D). Other functions with notable differences are the non-separable Ellipsoid and Discus functions. Here, the differences become more pronounced with larger dimension for the easier targets (the differences are only significant for the harder targets in 20-D for both functions and in 10-D for the Discus function). Other functions with similar behavior (statistically significant only in larger dimension and for the harder targets in favor of `CMA-bp`) are the Step Ellipsoid, non-separable Rastrigin, and the two Schaffer functions. Statistically significant differences in favor of `BoundTransform` cannot be observed. For the results of the statistical tests, we refer once again to the supplementary material.

7 CONCLUSIONS

We investigated the performance of two variants of CMA-ES handling bound constraints on both the original `bbob` and the newly introduced box-constrained `sbox-cost` test suite. Two main observations can be made. First, the performance differences (on both test suites) are larger between the two boundary handling methods compared to the observed differences for the same algorithm, when run on the two test suites. Second, on about half of the 24 test functions, the `BoundPenalty` boundary handling of CMA-ES performs (slightly) better than the default `BoundTransform` setting. The observed performance degradation for `CMA-bt` especially on the Ellipsoid and Discus functions and on some additional functions for the hardest targets suggest to revisit the algorithm in future work.

ACKNOWLEDGMENTS

This research was conducted with support of the consortium in Applied Mathematics CIROQUO, gathering partners in technological research and academia in the development of advanced methods for Computer Experiments.

REFERENCES

[1] S. Finck, N. Hansen, R. Ros, and A. Auger. 2009. *Real-Parameter Black-Box Optimization Benchmarking 2009: Presentation of the Noiseless Functions.* Technical Report 2009/20. Research Center PPE. http://coco.lri.fr/downloads/download15.03/bbobdocfunctions.pdf Updated February 2010.

[2] Nikolaus Hansen, Anne Auger, Dimo Brockhoff, and Tea Tušar. 2022. Anytime Performance Assessment in Blackbox Optimization Benchmarking. *IEEE Transactions on Evolutionary Computation* 26, 6 (2022), 1293–1305.

[3] N. Hansen, A. Auger, S. Finck, and R. Ros. 2012. *Real-Parameter Black-Box Optimization Benchmarking 2012: Experimental Setup.* Technical Report. INRIA. http://numbbo.github.io/gforge/bbob2012-downloads

[4] 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. https://doi.org/10.1080/10556788.2020.1808977

[5] N. Hansen, S. Finck, R. Ros, and A. Auger. 2009. *Real-Parameter Black-Box Optimization Benchmarking 2009: Noiseless Functions Definitions.* Technical Report RR-6829. INRIA. http://hal.inria.fr/inria-00362633/en/

[6] N. Hansen, S.P.N. Niederberger, L. Guzzella, and P. Koumoutsakos. 2009. A Method for Handling Uncertainty in Evolutionary Optimization with an Application to Feedback Control of Combustion. *IEEE Transactions on Evolutionary Computation* 13, 1 (2009), 180–197.

[7] N. Hansen and A. Ostermeier. 2001. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation* 9, 2 (2001), 159–195.

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

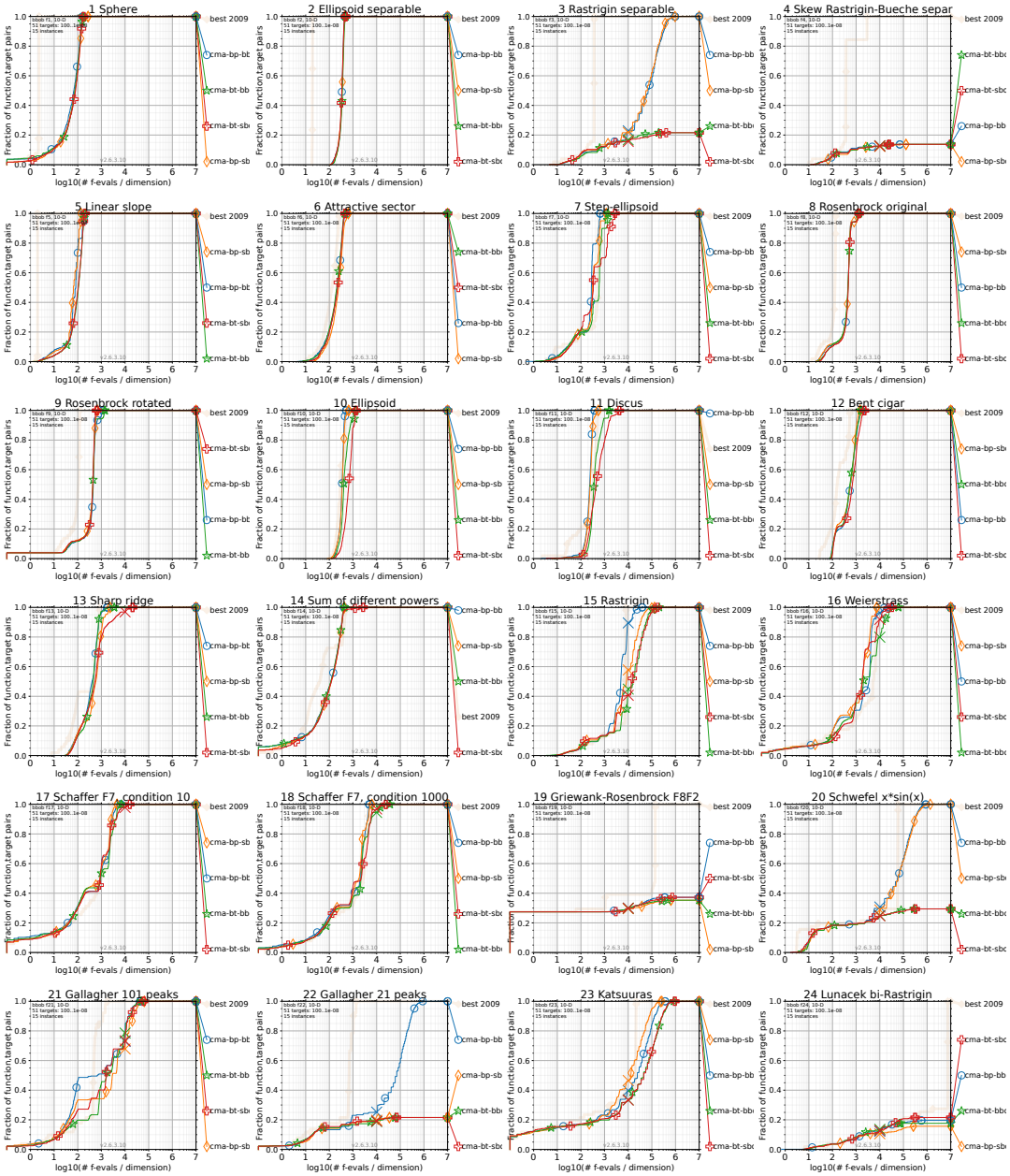

Figure 4: Empirical cumulative distribution of simulated (bootstrapped) runtimes, measured in number of f-evaluations, divided by dimension (FEvals/DIM) for the 51 targets $10^{[-8..2]}$ in dimension 10.

[9] 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. https://doi.org/10.1109/ICEC.1997.592287

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f1	11	12	12	12	12	12	12	15/15
bp-bbob	2.0(2)	**8.5**(3)	**14**(3)	**21**(4)	**27**(5)	**39**(5)	**53**(5)	15/15
bp-sbox	2.1(1)	9.3(4)	15(4)	22(4)	29(5)	42(3)	54(5)	15/15
bt-bbob	**1.9**(2)	10(4)	16(5)	24(6)	31(7)	42(7)	55(5)	15/15
bt-sbox	2.3(2)	9.1(4)	16(5)	23(6)	30(5)	42(5)	56(6)	15/15
f2	83	87	88	89	90	92	94	15/15
bp-bbob	7.3(3)	9.4(3)	11(2)	12(1)	13(2)	14(2)	15(2)	15/15
bp-sbox	9.1(5)	11(5)	13(5)	13(5)	14(5)	16(5)	17(6)	15/15
bt-bbob	**7.3**(2)	9.0(1)	**10**(2)	**11**(2)	**12**(2)	**13**(2)	**14**(2)	15/15
bt-sbox	7.8(2)	**9.0**(2)	10(1)	11(1)	12(1)	13(1)	15(1)	15/15
f3	716	1622	1637	1642	1646	1650	1654	15/15
bp-bbob	**0.99**(1)	**14**(20)	96(143)	96(99)	96(87)	96(121)	129(196)	3/15
bp-sbox	1.3(2)	21(22)	96(82)	96(137)	96(89)	96(131)	96(108)	4/15
bt-bbob	1.1(1)	15(16)	**71**(72)	**71**(79)	**71**(63)	**72**(65)	**72**(109)	5/15
bt-sbox	1.2(1)	29(41)	213(295)	212(222)	212(177)	212(231)	212(296)	2/15
f4	809	1633	1688	1758	1817	1886	1903	15/15
bp-bbob	1.9(1)	∞	∞	∞	∞	∞	∞ *5e4*	0/15
bp-sbox	**1.5**(1)	∞	∞	∞	∞	∞	∞ *5e4*	0/15
bt-bbob	1.8(3)	∞	∞	∞	∞	∞	∞ *5e4*	0/15
bt-sbox	1.7(2)	∞	∞	∞	∞	∞	∞ *5e4*	0/15
f5	10	10	10	10	10	10	10	15/15
bp-bbob	7.4(4)	24(19)	29(21)	30(19)	30(19)	30(19)	30(19)	15/15
bp-sbox	**5.2**(3)	**21**(13)	**26**(16)	**26**(16)	**26**(16)	**26**(16)	**26**(16)	15/15
bt-bbob	12(7)	23(9)	32(10)	40(10)	47(8)	63(10)	78(10)	15/15
bt-sbox	11(4)	23(7)	31(8)	37(8)	45(7)	60(9)	75(10)	15/15
f6	114	214	281	404	580	1038	1332	15/15
bp-bbob	**1.5**(0.7)	**1.7**(0.5)	**1.9**(0.3)	1.7(0.2)	1.5(0.2)	1.2(0.1)	1.2(0.1)	15/15
bp-sbox	2.1(1)	2.0(1)	2.2(0.9)	2.0(0.8)	1.8(0.7)	1.3(0.4)	1.3(0.4)	15/15
bt-bbob	1.8(0.7)	1.8(0.5)	2.0(0.4)	1.8(0.3)	1.6(0.3)	1.2(0.2)	1.2(0.1)	15/15
bt-sbox	1.5(0.8)	1.7(0.5)	1.9(0.4)	**1.7**(0.2)	**1.5**(0.2)	**1.2**(0.2)	**1.2**(0.1)	15/15
f7	24	324	1171	1451	1572	1572	1597	15/15
bp-bbob	**3.3**(2)	1.2(1)	0.70(0.4)	0.74(0.3)	0.76(0.5)	0.76(0.5)	0.85(0.6)	15/15
bp-sbox	4.3(2)	1.3(0.8)	**0.65**(0.4)	**0.65**(0.4)$_{\uparrow 1}$	**0.63**(0.3)$_{\uparrow 1}$	**0.63**(0.3)$_{\uparrow 1}$	**0.64**(0.3)$_{\uparrow 1}$	15/15
bt-bbob	3.9(2)	1.0(1)	0.91(0.8)	0.83(0.7)	0.90(0.7)	0.90(0.7)	0.97(0.7)	15/15
bt-sbox	4.2(2)	**0.94**(0.5)	0.82(0.5)	0.78(0.5)	0.74(0.4)	0.74(0.4)	0.77(0.4)	15/15
f8	73	273	336	372	391	410	422	15/15
bp-bbob	**2.4**(0.8)	3.5(1)	3.9(1)	4.1(1)	4.2(1)	4.5(1)	4.7(1.0)	15/15
bp-sbox	3.0(2)	**3.0**(2)	**3.7**(1)	**3.8**(1)	**3.9**(1)	**4.1**(1)	**4.4**(1)	15/15
bt-bbob	3.0(1)	4.2(4)	4.5(3)	4.6(3)	4.8(3)	5.0(3)	5.2(3)	15/15
bt-sbox	3.4(2)	3.3(1)	3.9(0.7)	4.1(0.9)	4.2(0.9)	4.4(0.9)	4.6(0.9)	15/15
f9	35	127	214	263	300	335	369	15/15
bp-bbob	3.9(2)	6.4(2)	5.7(1)	5.2(1)	4.9(0.9)	5.0(0.7)	4.9(0.6)	15/15
bp-sbox	**3.6**(1)	6.5(2)	5.7(1)	5.3(1.0)	5.0(0.9)	4.9(0.8)	4.9(0.8)	15/15
bt-bbob	4.6(2)	**5.0**(1)	**4.7**(0.7)	**4.6**(0.6)	**4.4**(0.6)	**4.5**(0.6)	**4.5**(0.5)	15/15
bt-sbox	4.4(1)	6.7(4)	5.8(2)	5.3(2)	5.1(1)	5.1(1)	5.1(1.0)	15/15
f10	349	500	574	607	626	829	880	15/15
bp-bbob	**1.7**(0.5)	**1.7**(0.4)	**1.7**(0.2)	**1.7**(0.2)	**1.8**(0.2)	**1.5**(0.2)	**1.6**(0.2)	15/15
bp-sbox	2.0(0.5)	1.8(0.4)	1.8(0.4)	1.8(0.2)	1.9(0.2)	1.6(0.1)	1.7(0.2)	15/15
bt-bbob	3.1(2)	3.1(3)	2.8(2)	2.8(2)	2.9(2)	2.3(2)	2.4(1)	15/15
bt-sbox	2.1(0.6)	2.4(0.9)	2.9(3)	3.1(3)	3.3(3)	2.8(2)	2.8(2)	15/15
f11	143	202	763	977	1177	1467	1673	15/15
bp-bbob	**3.5**(1)	**3.3**(1)	**1.0**(0.2)	**0.89**(0.2)	**0.80**(0.1)$_{\uparrow 0}$	**0.75**(0.1)$_{\uparrow 0}$	**0.75**(0.1)$_{\uparrow 0}$	15/15
bp-sbox	3.9(1)	3.5(0.7)	1.1(0.2)	0.92(0.2)	0.84(0.1)$_{\uparrow 1}$	0.77(0.1)$_{\uparrow 1}$	0.77(0.1)$_{\uparrow 1}$	15/15
bt-bbob	5.4(4)	4.5(3)	1.3(0.8)	1.1(0.6)	0.99(0.5)	0.90(0.4)	0.89(0.3)	15/15
bt-sbox	4.4(2)	4.0(1)	1.4(0.3)	1.2(0.5)	1.6(1)	1.7(2)	1.7(2)	15/15
f12	108	268	371	413	461	1303	1494	15/15
bp-bbob	**4.6**(0.7)	3.9(4)	4.3(4)	4.9(4)	5.5(4)	2.7(2)	2.9(2)	15/15
bp-sbox	5.2(3)	**3.6**(3)	**4.2**(4)	**4.8**(3)	**5.2**(3)	**2.5**(1)	**2.5**(1)	15/15
bt-bbob	7.6(5)	5.4(7)	5.6(5)	6.0(4)	6.2(4)	2.8(2)	2.8(2)	15/15
bt-sbox	4.7(0.6)	4.7(5)	6.8(5)	7.9(6)	8.0(6)	3.5(2)	3.5(2)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f13	132	195	250	319	1310	1752	2255	15/15
bp-bbob	**2.4**(0.5)	3.9(1)	4.1(1)	3.9(1.0)	1.1(0.2)	**1.1**(0.2)	1.1(0.1)	15/15
bp-sbox	2.9(1)	**3.4**(1)	**3.6**(1.0)	4.2(0.6)	1.2(0.2)	1.1(0.2)	1.1(0.1)	15/15
bt-bbob	2.8(1)	3.7(1)	4.0(1)	**3.8**(0.7)	**1.1**(0.2)	1.1(0.2)	**1.0**(0.2)	15/15
bt-sbox	3.2(1)	4.1(0.8)	4.7(3)	5.8(6)	2.0(2)	6.1(14)	6.7(11)	12/15
f14	10	41	58	90	139	251	476	15/15
bp-bbob	**0.84**(0.8)	**2.2**(0.8)	3.4(1)	3.6(0.8)	**3.3**(0.6)	**3.2**(0.5)	**2.5**(0.4)	15/15
bp-sbox	1.0(1)	2.8(2)	3.7(1)	3.7(0.7)	4.0(1)	3.6(0.8)	2.6(0.4)	15/15
bt-bbob	0.87(1)	2.4(1)	3.7(1)	**3.5**(1)	3.4(0.8)	3.4(0.5)	2.5(0.2)	15/15
bt-sbox	1.7(2)	2.5(1)	**3.3**(1)	3.8(0.6)	3.7(0.8)	3.5(0.7)	3.1(1)	15/15
f15	511	9310	19369	19743	20073	20769	21359	14/15
bp-bbob	**1.0**(0.5)	1.4(2)	**0.92**(0.8)	**0.92**(0.8)	**0.93**(0.8)	**0.93**(0.8)	**0.94**(0.8)	15/15
bp-sbox	1.6(2)	**0.92**(1)	1.1(0.8)	1.1(0.8)	1.2(0.8)	1.1(0.8)	1.2(0.8)	14/15
bt-bbob	1.7(2)	1.8(2)	2.0(2)	2.0(2)	2.0(3)	2.0(2)	2.0(2)	11/15
bt-sbox	1.8(2)	3.4(4)	3.2(3)	3.2(3)	3.1(3)	3.1(4)	3.0(5)	8/15
f16	120	612	2662	10163	10449	11644	12095	15/15
bp-bbob	**1.5**(1)	2.1(3)	1.7(3)	**0.50**(0.7)	**0.51**(0.7)	**0.49**(0.7)	**0.61**(0.8)	15/15
bp-sbox	1.6(1)	2.7(2)	2.2(3)	0.77(0.8)	0.77(0.7)	0.74(0.7)	0.75(0.7)	15/15
bt-bbob	1.7(2)	3.2(6)	2.0(2)	0.62(0.6)	0.75(0.8)	0.71(0.7)	0.72(0.7)	15/15
bt-sbox	2.0(4)	**1.9**(3)	**1.6**(3)	0.88(0.9)	0.98(0.9)	0.96(0.8)	0.95(0.8)	14/15
f17	5.0	215	899	2861	3669	6351	7934	15/15
bp-bbob	1.3(1)	**0.90**(0.3)	**0.70**(0.3)$_{\uparrow 1}$	**0.53**(0.5)	0.97(0.5)	0.96(0.6)	1.1(1.0)	15/15
bp-sbox	2.3(2)	0.97(0.7)	1.1(2)	0.61(0.6)	0.94(0.5)	0.93(0.5)	1.4(0.7)	15/15
bt-bbob	**0.92**(1)	1.7(0.4)	0.85(1)	0.55(0.5)	**0.72**(0.5)	0.92(0.5)	1.0(0.5)	15/15
bt-sbox	3.3(4)	2.4(0.7)	1.2(2)	0.91(0.7)	0.89(0.6)	**0.81**(0.1)$_{\uparrow 1}$	**0.80**(0.4)	15/15
f18	103	378	3968	8451	9280	10905	12469	15/15
bp-bbob	0.88(0.7)	1.0(0.7)	**0.59**(0.9)	**0.50**(0.4)$_{\uparrow 1}$	**0.51**(0.4)$_{\uparrow 1}$	**0.84**(0.6)	**0.93**(0.5)	15/15
bp-sbox	**0.77**(0.5)	3.0(6)	1.0(1)	0.67(0.7)	0.89(1)	1.1(0.8)	1.0(0.7)	15/15
bt-bbob	0.78(0.6)	**0.98**(0.6)	1.0(1)	0.93(0.7)	0.93(0.6)	0.96(0.5)	1.1(0.8)	15/15
bt-sbox	1.1(0.7)	4.3(5)	1.4(2)	1.1(0.9)	1.0(0.8)	1.0(0.5)	1.4(0.6)	14/15
f19	1	1	242	1.0e5	1.2e5	1.2e5	1.2e5	15/15
bp-bbob	**1**(0)	**1**(0)	145(226)	6.7(9)	∞	∞	∞ *5e4*	0/15
bp-sbox	**1**(0)	**1**(0)	144(170)	6.8(7)	5.9(6)	5.9(7)	5.8(5)	1/15
bt-bbob	**1**(0)	**1**(0)	**142**(158)	**1.3**(1)	**2.0**(1)	**3.0**(4)	**3.0**(3)	2/15
bt-sbox	**1**(0)	**1**(0)	154(263)	2.1(3)	6.1(5)	6.1(7)	6.0(5)	1/15
f20	16	851	38111	51362	54470	54861	55313	14/15
bp-bbob	1.9(1)	**3.7**(3)	1.6(2)	1.3(0.9)	1.3(0.9)	1.4(1)	1.4(2)	8/15
bp-sbox	2.0(2)	8.1(6)	1.6(1)	1.2(1)	1.2(0.8)	1.2(1)	1.3(1)	8/15
bt-bbob	1.9(2)	5.5(4)	**1.2**(0.9)	**1.0**(0.8)	**0.98**(1)	**0.99**(0.7)	**1.00**(0.7)	10/15
bt-sbox	**1.7**(2)	6.3(6)	1.3(1)	1.0(0.8)	0.98(0.8)	1.00(0.8)	1.1(1)	10/15
f21	41	1157	1674	1692	1705	1729	1757	14/15
bp-bbob	1.5(1)	**4.6**(3)	**4.6**(6)	**4.7**(6)	**4.7**(6)	**4.9**(7)	**5.0**(7)	15/15
bp-sbox	**1.4**(0.9)	6.1(6)	8.1(12)	8.1(12)	8.2(12)	8.3(12)	8.4(13)	14/15
bt-bbob	1.7(1)	5.2(5)	16(26)	16(27)	15(21)	15(20)	15(23)	11/15
bt-sbox	5.8(17)	11(9)	15(18)	15(18)	15(22)	15(17)	16(18)	12/15
f22	71	386	938	980	1008	1040	1068	14/15
bp-bbob	9.3(22)	21(25)	120(164)	115(95)	112(155)	109(158)	107(118)	5/15
bp-sbox	3.6(9)	**19**(16)	67(113)	64(75)	**63**(68)	**61**(65)	**59**(76)	7/15
bt-bbob	**1.2**(0.7)	23(45)	86(150)	82(103)	80(90)	78(123)	76(99)	6/15
bt-sbox	2.1(0.7)	19(17)	**58**(58)	**56**(106)	67(74)	76(88)	74(98)	6/15
f23	3.0	518	14249	27890	31654	33030	34256	15/15
bp-bbob	3.7(5)	**6.9**(5)	**1.8**(2)	**1.3**(2)	**1.2**(1)	**1.1**(1)	**1.1**(1)	12/15
bp-sbox	2.7(4)	7.4(7)	5.2(7)	2.7(3)	2.4(3)	2.3(4)	2.3(2)	7/15
bt-bbob	**2.5**(3)	7.0(7)	4.0(5)	2.3(2)	2.1(2)	2.0(1)	2.0(2)	8/15
bt-sbox	2.9(2)	9.3(7)	2.8(4)	1.5(1)	1.3(1.0)	1.3(1)	1.2(1.0)	11/15
f24	1622	2.2e5	6.4e6	9.6e6	9.6e6	1.3e7	1.3e7	3/15
bp-bbob	1.4(2)	**0.56**(0.5)	∞	∞	∞	∞	∞ *5e4*	0/15
bp-sbox	1.6(1)	3.4(3)	∞	∞	∞	∞	∞ *5e4*	0/15
bt-bbob	2.2(2)	3.4(3)	**0.12**(0.1)	∞	∞	∞	∞ *5e4*	0/15
bt-sbox	**1.4**(1)	3.4(3)	∞	∞	∞	∞	∞ *5e4*	0/15

Table 1: Expected runtime (ERT in number of f-evaluations) divided by the respective best ERT measured during BBOB-2009 (when finite) in dimension 5. This ERT ratio and, in braces as dispersion measure, the half difference between 10 and 90%-tile of bootstrapped run 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 ↑ signifies the number of trials that were worse than the ERT of the best algorithm from BBOB 2009 shown only when less than 10 percent were worse and the ERT was better.Best results are printed in bold. Data produced with COCO v2.6.3.10

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
bp-bbob	**6.5**(1)	**13**(1)	**19**(1)	**25**(1)	**32**(1)*	**45**(3)	**57**(4)	15/15
bp-sbox	7.1(1)	14(2)	20(2)	26(1)	33(1)	46(2)	59(4)	15/15
bt-bbob	8.0(2)	15(3)	21(3)	28(3)	34(3)	47(4)	60(4)	15/15
bt-sbox	8.4(2)	16(2)	23(4)	29(3)	37(5)	51(6)	64(8)	15/15
f2	385	386	387	388	390	391	393	15/15
bp-bbob	**20**(4)	25(3)	28(3)	30(2)	31(2)	33(1)	34(2)	15/15
bp-sbox	21(4)	**25**(4)	**27**(3)	**29**(3)	**30**(2)	**32**(2)	**33**(2)	15/15
bt-bbob	22(1)	26(3)	29(3)	30(2)	31(2)	32(2)	34(2)	15/15
bt-sbox	22(2)	25(2)	29(2)	30(2)	31(2)	33(2)	34(2)	15/15
f3	5066	7626	7635	7637	7643	7646	7651	15/15
bp-bbob	**21**(22)	∞	∞	∞	∞	∞	∞ *2e5*	0/15
bp-sbox	26(30)	∞	∞	∞	∞	∞	∞ *2e5*	0/15
bt-bbob	25(29)	∞	∞	∞	∞	∞	∞ *2e5*	0/15
bt-sbox	41(48)	∞	∞	∞	∞	∞	∞ *2e5*	0/15
f4	4722	7628	7666	7686	7700	7758	1.4e5	9/15
bp-bbob	∞	∞	∞	∞	∞	∞	∞ *2e5*	0/15
bp-sbox	∞	∞	∞	∞	∞	∞	∞ *2e5*	0/15
bt-bbob	∞	∞	∞	∞	∞	∞	∞ *2e5*	0/15
bt-sbox	∞	∞	∞	∞	∞	∞	∞ *2e5*	0/15
f5	41	41	41	41	41	41	41	15/15
bp-bbob	30(24)	51(33)	56(35)	57(35)	57(35)	57(35)	57(35)	15/15
bp-sbox	30(18)	45(21)	50(24)	52(24)	**52**(24)	**52**(24)	**52**(24)	15/15
bt-bbob	**22**(3)	**33**(8)	**41**(8)	**48**(8)	55(8)	69(9)	83(7)	15/15
bt-sbox	23(8)	37(9)	44(10)	52(9)	59(8)	73(9)	87(11)	15/15
f6	1296	2343	3413	4255	5220	6728	8409	15/15
bp-bbob	1.5(0.8)	1.2(0.6)	1.1(0.5)	**1.1**(0.4)	1.1(0.3)	1.1(0.3)	1.1(0.3)	15/15
bp-sbox	1.9(0.9)	1.5(0.6)	1.4(0.5)	1.4(0.5)	1.3(0.4)	1.3(0.4)	1.3(0.3)	15/15
bt-bbob	1.5(0.4)	1.2(0.2)	**1.1**(0.1)	1.1(0.1)	**1.1**(0.1)	**1.1**(0.1)	**1.1**(0.1)	15/15
bt-sbox	**1.4**(0.3)	**1.2**(0.2)	1.1(0.2)	1.1(0.1)	1.1(0.1)	1.1(0.2)	1.1(0.2)	15/15
f7	1351	4274	9503	16523	16524	16524	16969	15/15
bp-bbob	**0.72**(0.2)$_{\uparrow 1}$	**2.8**(2)	**1.6**(0.6)	**1.0**(0.4)	**1.0**(0.4)	**1.0**(0.4)	**1.0**(0.4)	15/15
bp-sbox	0.96(0.3)	2.8(2)	1.9(0.6)	1.2(0.4)	1.2(0.4)	1.2(0.4)	1.2(0.4)	15/15
bt-bbob	1.1(0.4)	4.6(3)	3.2(2)	2.0(1)	2.0(1)	2.0(1)	1.9(1)	15/15
bt-sbox	1.7(2)	5.4(2)	4.3(3)	2.8(2)	2.9(1)	2.9(1)	2.8(1)	15/15
f8	2039	3871	4040	4148	4219	4371	4484	15/15
bp-bbob	3.4(1.0)	**3.4**(0.5)	**3.6**(0.5)	**3.7**(0.5)	**3.7**(0.5)	**3.8**(0.4)	**3.8**(0.5)	15/15
bp-sbox	**3.4**(0.8)	3.9(2)	4.1(2)	4.2(2)	4.3(2)	4.3(2)	4.3(2)	15/15
bt-bbob	3.4(0.8)	3.6(0.6)	3.8(0.5)	3.9(0.5)	4.0(0.5)	4.0(0.5)	4.0(0.5)	15/15
bt-sbox	3.6(0.7)	3.4(0.3)	3.7(0.2)	3.8(0.2)	3.8(0.2)	3.8(0.2)	3.9(0.2)	15/15
f9	1716	3102	3277	3379	3455	3594	3727	15/15
bp-bbob	**3.8**(0.9)	5.3(4)	5.6(3)	5.6(3)	5.7(3)	5.6(3)	5.6(3)	15/15
bp-sbox	4.0(0.7)	5.2(3)	5.5(3)	5.6(3)	5.6(3)	5.6(3)	5.5(3)	15/15
bt-bbob	3.9(0.8)	**4.0**(0.4)	**4.3**(0.4)	**4.4**(0.4)	**4.4**(0.4)	**4.4**(0.4)	**4.4**(0.4)	15/15
bt-sbox	3.9(0.8)	4.5(0.6)	4.8(0.6)	4.9(0.6)	4.9(0.6)	4.9(0.6)	4.9(0.6)	15/15
f10	7413	8661	10735	13641	14920	17073	17476	15/15
bp-bbob	**1.1**(0.1)	**1.1**(0.1)	**1.0**(0.1)	**0.87**(0.1)$_{\uparrow 0}$	**0.82**(0.0)$_{\uparrow 0}$	**0.76**(0.0)$_{\uparrow 0}$	**0.77**(0.0)$_{\uparrow 0}$	15/15
bp-sbox	1.1(0.2)	1.2(0.1)	1.1(0.1)	0.90(0.1)$_{\uparrow 1}$	0.85(0.0)$_{\uparrow 0}$	0.79(0.0)$_{\uparrow 0}$	0.81(0.0)$_{\uparrow 0}$	15/15
bt-bbob	1.6(0.6)	1.6(0.5)	1.3(0.3)	1.1(0.3)	1.0(0.2)	0.95(0.2)	0.96(0.2)	15/15
bt-sbox	1.5(0.4)	1.9(1)	1.8(0.9)	1.6(0.7)	1.6(0.6)	1.5(0.5)	1.5(0.5)	15/15
f11	1002	2228	6278	8586	9762	12285	14831	15/15
bp-bbob	**4.2**(0.3)	**2.2**(0.2)	**0.85**(0.1)$_{\uparrow 0}$	**0.67**(0.0)$_{\uparrow 0}$	**0.62**(0.0)$_{\uparrow 0}$	**0.55**(0.0)$_{\uparrow 0}$	**0.50**(0.0)$_{\uparrow 0}$	15/15
bp-sbox	4.3(0.4)	2.2(0.1)	0.88(0.1)$_{\uparrow 0}$	0.69(0.0)$_{\uparrow 0}$	0.64(0.0)$_{\uparrow 0}$	0.57(0.0)$_{\uparrow 0}$	0.51(0.0)$_{\uparrow 0}$	15/15
bt-bbob	7.4(2)	4.4(2)	1.6(0.7)	1.3(0.5)	1.1(0.4)	0.96(0.3)	0.84(0.3)	15/15
bt-sbox	10(6)	5.5(3)	2.2(1.0)	2.0(1.0)	2.0(0.9)	1.8(1)	1.6(0.9)	15/15
f12	1042	1938	2740	3156	4140	12407	13827	15/15
bp-bbob	2.6(2)	3.5(3)	4.0(2)	4.2(2)	3.7(1)	1.5(0.5)	1.6(0.4)	15/15
bp-sbox	2.5(0.9)	**2.4**(2)	**2.9**(2)	**3.3**(1)	**3.1**(1)	**1.3**(0.3)	**1.4**(0.2)	15/15
bt-bbob	3.3(3)	3.3(3)	3.7(2)	3.9(2)	3.5(1)	1.5(0.3)	1.5(0.3)	15/15
bt-sbox	**2.2**(0.3)	2.5(3)	3.0(3)	3.8(2)	3.7(2)	1.6(0.6)	1.6(0.6)	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
bp-bbob	3.2(3)	3.3(2)	**3.6**(2)	**3.5**(2)	**0.72**(0.4)	1.3(0.9)	1.4(0.5)	15/15
bp-sbox	3.5(2)	12(8)	10(6)	14(29)	3.8(6)	4.2(5)	3.6(4)	10/15
bt-bbob	3.8(3)	**2.6**(2)	4.5(3)	4.4(2)	0.93(0.4)	**1.1**(0.7)	**1.2**(0.6)	15/15
bt-sbox	**3.1**(1)	3.0(2)	5.3(4)	5.1(4)	1.1(0.8)	1.7(1)	1.7(1)	14/15
f14	75	239	304	451	932	1648	15661	15/15
bp-bbob	**2.4**(1)	**2.3**(0.4)	**3.3**(0.4)	**3.7**(0.4)	**3.0**(0.2)	**3.8**(0.2)	0.66(0.0)$_{\uparrow 0}$	15/15
bp-sbox	3.0(1)	2.5(0.6)	3.3(0.3)	3.7(0.4)	3.1(0.2)	3.8(0.3)	**0.65**(0.1)$_{\uparrow 0}$	15/15
bt-bbob	3.0(1)	2.9(0.7)	3.7(0.4)	4.0(0.6)	3.3(0.2)	3.9(0.2)	0.69(0.0)$_{\uparrow 0}$	15/15
bt-sbox	3.8(1)	2.8(0.6)	3.6(0.3)	3.9(0.4)	3.2(0.3)	3.9(0.7)	0.73(0.1)$_{\uparrow 1}$	15/15
f15	30378	1.5e5	3.1e5	3.2e5	3.2e5	4.5e5	4.6e5	15/15
bp-bbob	0.98(0.5)	**1.3**(0.9)	**1.4**(1)	**1.4**(2)	**1.7**(2)	**2.0**(3)	**2.0**(2)	3/15
bp-sbox	1.3(0.5)	6.2(8)	4.5(4)	4.5(5)	4.4(5)	3.2(3)	3.1(3)	2/15
bt-bbob	1.2(0.8)	20(22)	∞	∞	∞	∞	∞ *2e5*	0/15
bt-sbox	**0.97**(0.8)	20(16)	∞	∞	∞	∞	∞ *2e5*	0/15
f16	1384	27265	77015	1.4e5	1.9e5	2.0e5	2.2e5	15/15
bp-bbob	**1.2**(0.3)	**0.30**(0.2)$_{\uparrow 0}$	0.66(0.5)	**0.74**(0.6)	**1.0**(2)	2.0(2)	1.8(2)	6/15
bp-sbox	1.3(0.3)	0.42(0.2)$_{\uparrow 1}$	**0.60**(0.6)	0.76(0.5)	1.2(1)	**1.3**(1)	**1.2**(0.7)	9/15
bt-bbob	4.1(3)	0.58(0.4)$_{\uparrow 1}$	1.2(1)	2.0(2)	2.1(2)	2.6(4)	2.4(2)	5/15
bt-sbox	3.6(3)	0.58(0.5)	0.96(1)	1.1(1)	1.4(1)	3.4(4)	3.1(3)	4/15
f17	63	1030	4005	12242	30677	56288	80472	15/15
bp-bbob	0.98(0.6)	**0.90**(0.3)	1.1(2)	**1.1**(0.7)	**0.76**(0.3)	**0.77**(0.0)$_{\uparrow 1}$	**0.78**(0.3)	15/15
bp-sbox	1.4(0.9)	1.0(0.4)	1.4(2)	1.2(0.7)	0.79(0.4)	1.1(0.4)	1.1(0.3)	14/15
bt-bbob	**0.97**(0.6)	1.1(0.2)	2.0(2)	1.6(0.5)	1.0(0.9)	1.1(0.4)	0.98(0.3)	15/15
bt-sbox	1.8(1)	0.95(0.2)	**1.1**(2)	1.7(1)	1.9(3)	1.6(2)	1.6(1)	13/15
f18	621	3972	19561	28555	67569	1.3e5	1.5e5	15/15
bp-bbob	**0.85**(0.3)	0.90(2)	0.87(0.8)	**1.3**(0.4)	**0.80**(0.3)	**0.72**(0.5)	**0.76**(0.4)	14/15
bp-sbox	0.88(0.2)	**0.46**(0.1)$_{\uparrow 0}$	**0.86**(0.6)	1.3(0.9)	1.8(2)	3.0(3)	3.5(5)	5/15
bt-bbob	0.96(0.2)	0.97(0.1)$_{\uparrow 1}$	1.2(1.0)	3.4(4)	2.3(2)	2.1(2)	2.0(2)	7/15
bt-sbox	1.1(0.3)	0.55(0.2)$_{\uparrow 1}$	0.89(0.5)	2.0(2)	2.6(3)	2.1(2)	2.9(2)	5/15
f19	1	1	3.4e5	4.7e6	6.2e6	6.7e6	6.7e6	15/15
bp-bbob	**1**(0)	**1**(0)	**2.8**(3)	∞	∞	∞	∞ *2e5*	0/15
bp-sbox	**1**(0)	**1**(0)	8.7(9)	∞	∞	∞	∞ *2e5*	0/15
bt-bbob	**1**(0)	**1**(0)	8.5(14)	∞	∞	∞	∞ *2e5*	0/15
bt-sbox	**1**(0)	**1**(0)	8.5(9)	∞	∞	∞	∞ *2e5*	0/15
f20	82	46150	3.1e6	5.5e6	5.5e6	5.6e6	5.6e6	14/15
bp-bbob	3.8(0.7)	**5.5**(4)	∞	∞	∞	∞	∞ *2e5*	0/15
bp-sbox	**3.5**(1)	11(13)	∞	∞	∞	∞	∞ *2e5*	0/15
bt-bbob	4.0(2)	6.2(7)	∞	∞	∞	∞	∞ *2e5*	0/15
bt-sbox	4.4(2)	14(14)	∞	∞	∞	∞	∞ *2e5*	0/15
f21	561	6541	14103	14318	14643	15567	17589	15/15
bp-bbob	**1.4**(1)	30(31)	29(25)	29(46)	28(51)	26(33)	23(40)	5/15
bp-sbox	5.4(11)	25(39)	23(39)	23(18)	22(33)	21(20)	19(32)	6/15
bt-bbob	7.9(24)	45(73)	40(50)	39(35)	38(48)	36(26)	32(34)	4/15
bt-sbox	4.1(5)	**16**(26)	**10**(10)	**10**(8)	**10**(7)	**9.0**(16)	**8.9**(13)	9/15
f22	467	5580	23491	24163	24948	26847	1.3e5	12/15
bp-bbob	16(28)	**42**(35)	∞	∞	∞	∞	∞ *2e5*	0/15
bp-sbox	13(30)	49(59)	∞	∞	∞	∞	∞ *2e5*	0/15
bt-bbob	33(6)	57(63)	∞	∞	∞	∞	∞ *2e5*	0/15
bt-sbox	**4.6**(0.8)	43(39)	∞	∞	∞	∞	∞ *2e5*	0/15
f23	3.0	1614	67457	3.7e5	4.9e5	8.1e5	8.4e5	15/15
bp-bbob	**1.6**(2)	**22**(20)	**1.8**(2)	∞	∞	∞	∞ *2e5*	0/15
bp-sbox	2.4(3)	56(67)	2.6(2)	∞	∞	∞	∞ *2e5*	0/15
bt-bbob	2.3(4)	90(119)	5.4(5)	∞	∞	∞	∞ *2e5*	0/15
bt-sbox	2.0(2)	179(252)	7.1(11)	∞	∞	∞	∞ *2e5*	0/15
f24	1.3e6	7.5e6	5.2e7	5.2e7	5.2e7	5.2e7	5.2e7	3/15
bp-bbob	0.50(0.5)	∞	∞	∞	∞	∞	∞ *2e5*	0/15
bp-sbox	0.48(0.4)	∞	∞	∞	∞	∞	∞ *2e5*	0/15
bt-bbob	0.37(0.3)	∞	∞	∞	∞	∞	∞ *2e5*	0/15
bt-sbox	**0.30**(0.2)	∞	∞	∞	∞	∞	∞ *2e5*	0/15

Table 2: Expected runtime (ERT in number of f-evaluations) divided by the respective best ERT measured during BBOB-2009 (when finite) in dimension 20. This ERT ratio and, in braces as dispersion measure, the half difference between 10 and 90%-tile of bootstrapped run 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 ↑ signifies the number of trials that were worse than the ERT of the best algorithm from BBOB 2009 shown only when less than 10 percent were worse and the ERT was better.Best results are printed in bold. Data produced with COCO v2.6.3.10