Benchmarking MO-CMA-ES and COMO-CMA-ES on the Bi-objective bbob-biobj Testbed

Paul Dufossé
Thales DMS France, Inria, CMAP, Ecole Polytechnique, IP Paris
paul.dufosse@inria.fr

Cheikh Touré
Inria, CMAP, Ecole Polytechnique, IP Paris
cheikh.toure@polytechnique.edu

ABSTRACT

In this paper, we propose a comparative benchmark of MO-CMA-ES, COMO-CMA-ES (recently introduced in [12]) and NSGA-II, using the COCO framework for performance assessment and the Bi-objective test suite bbob-biobj. For a fixed number of points p, COMO-CMA-ES approximates an optimal p-distribution of the Hypervolume Indicator. While not designed to perform on archive-based assessment, *i.e.* with respect to all points evaluated so far by the algorithm, COMO-CMA-ES behaves well on the COCO platform. The experiments are done in a true Black-Blox spirit by using a minimal setting relative to the information shared by the 55 problems of the bbob-biobj Testbed.

CCS CONCEPTS

•Computing methodologies → Continuous space search;

KEYWORDS

Benchmarking, Black-box optimization, Bi-objective optimization

ACM Reference format:
Paul Dufossé and Cheikh Touré. 2019. Benchmarking MO-CMA-ES and COMO-CMA-ES on the Bi-objective bbob-biobj Testbed. In *Proceedings of Genetic and Evolutionary Computation Conference Companion, Prague, Czech Republic, July 13–17, 2019 (GECCO '19 Companion),* 8 pages.
DOI: 10.1145/3319619.3326892

1 INTRODUCTION

As the COCO platform assesses bi-objective algorithms by computing the covered Hypervolume of evaluated points in a black-box optimization framework, it is natural that well-performing benchmarked algorithms on COCO are designed to tackle these two issues (covering the front and black-box paradigm). Hence [9] uses a hybrid algorithm and parameter tuning to handle the black-box difficulty, and [11] develops an unbounded population size variant of MO-CMA-ES [8, 14] for the Pareto front covering issue.
MO-CMA-ES and COMO-CMA-ES, however, are non-hybrid algorithms with a population size p (number of kernels for COMO-CMA-ES) fixed *a priori*.

GECCO '19 Companion, Prague, Czech Republic

978-1-4503-6748-6/19/07…$15.00
DOI: 10.1145/3319619.3326892

2 ALGORITHMS DESCRIPTION

NSGA-II is a popular choice among Evolutionary Multi-Objective (EMO) algorithms [3], based on non dominated sorting of candidates and crowding distance comparisons. It has already been benchmarked on the bbob-biobj Testbed [1]. The archive data, *i.e.* the data for all points evaluated so far, is shown under the name NSGA-II along with the best2016 reference, as a baseline for the reader.

The elitist Multiobjective CMA-ES, MO-CMA-ES [8], is based on a two-way ranking: the Pareto ranking and the hypervolume contribution among individuals with the same Pareto rank. Each parent is not only a point in the search space but a 5-tuple of data representing a (1+1)-CMA-ES, which is a Single Objective (SO) optimizer defined in [8]. We use here the improved step-size adaptation designed in [14]. The algorithm also uses greedy mating, which means that parents are selected only among non-dominated solutions. We choose the default settings for the λ_{MO}-(1+1)-CMA-ES presented in [8]. In this paper, the term population size refers to the MO population size, namely the parameter λ_{MO}.

A recent EMO algorithm using the non-elitist CMA-ES [4] is the Comma Multiobjective CMA-ES (COMO-CMA-ES) defined in [12], and is designed to approximate the optimal p-distribution of the Hypervolume indicator. It is the instantiation of a wider framework called Sofomore, on the non-elitist CMA-ES [12]. We also take the default setup for the standard CMA-ES presented in [4]. The *Single-objective Optimization FOr Optimizing Multiobjective Optimization pRoblEms* framework (Sofomore) finds p points approximating the optimal p-distribution of the Hypervolume indicator, by iteratively maximizing an extended notion of the hypervolume contribution (hypervolume improvement) called Uncrowded Hypervolume Improvement UHVI [12]. As the two-way ranking in [8], UHVI unflattens the hypervolume improvement's level sets in dominated regions, but the novelty is inherent to the fact that this unflattening operation still preserves diversity among the solutions, by ensuring them to be uncrowded.

3 IMPLEMENTATION AND EXPERIMENTAL PROCEDURE

The code for MO-CMA-ES was written in Matlab, 2014, using routines from the Shark library for hypervolume computation. COMO-CMA-ES is implemented in Python and uses NumPy. Note that the hypervolume is computed in pure Python and does not call any NumPy method. Implementation details for NSGA-II can be found in [1].

For MO-CMA-ES, the algorithm starts with random starting point uniformly sampled in $I = [-5, 5]^n$ with an initial stepsize σ_0

Algorithm	p	2-D	3-D	5-D	10-D	20-D
MO-	10	2.8	3.3	3.0	3.6	3.3
CMA-ES	32	2.7	2.9	2.8	2.9	2.9
	100	3.0	3.2	3.5	3.4	3.0
COMO-	10	5.9	6.0	5.0	4.7	4.8
CMA-ES	32	5.8	5.2	5.8	4.1	4.9
	100	12	8.9	7.7	7.5	4.3

Table 1: CPU times per function evaluation of MO-CMA-ES and COMO-CMA-ES (in 10^{-4} seconds) for varying dimensions. p is the population size (or number of kernels).

equals to 1. From this starting point a population of p (1+1)-CMA-ES is evolved, with p the population size of the algorithm. Any (1+1)-CMA-ES can send a warning, meaning that it has somehow terminated. When any warning is met, the MO algorithm stops. Then a restart is performed with a random starting point in I, with the same settings. The population size is a core parameter fixed *a priori* for each run (including restarts), which is why MO-CMA-ES is benchmarked with different population sizes, namely 10, 32 and 100. The reference point used for computing the contributing hypervolume is iteratively chosen as the nadir point among the current population, and adding 1 to each coordinate.

COMO-CMA-ES also starts with a random point sampled in I, with an initial stepsize $\sigma_0 = \sqrt{n}$. No restart is performed and no parameter tuning is done on purpose to observe the behaviour of the algorithm with default settings. In the same spirit, the only stopping criteria is given by the allocated budget of function evaluations. As COMO-CMA-ES is designed to approximate the optimal p-distribution of the Hypervolume indicator, it is then necessary to fix one and for all a reference point for each run; which is done by setting an attribute of each COCO problem called `1argest_fvalues_of_interest` [7] as reference point.

We benchmark MO-CMA-ES (with population sizes equal to 10, 32 and 100), COMO-CMA-ES (with 3, 10, 32, 100, 316 and 1000 kernels), and NSGA-II. The latter is run with a population size of 100 as in [1]. COCO is run with the common settings for the `bbob-biobj` test suite; 10 instances with dimension $n \in \{2, 3, 5, 10, 20\}$. The allocated budget is $10^4 n$ function evaluations for each MO-CMA-ES, and $10^5 n$ for NSGA-II and each COMO-CMA-ES.

4 CPU TIMING

In order to evaluate the duration of each algorithm, we have reported here the timing results when running the full budget experiment. Both codes were run on a linux machine with 64 cores, Intel®Xeon ®E7 to E3 v4 processors. We are interested in CPU time per function evaluation for varying dimensions and population sizes. To account for Matlab internal parallelization we use the function **cputime** and not real timing. The results can be found in Table 1.

We are benchmarking algorithms written in different languages, therefore comparisons should be made carefully. COMO-CMA-ES computation per function evaluation takes 1.5 to 4 times longer than MO-CMA-ES, and this ratio is less important when the dimension increases.

5 RESULTS

Results from experiments according to [7], [5] and [2] on the benchmark functions given in [13] for the three algorithms called MO-*, NSGA-II or COMO-* (with * being the fixed population size) are presented in Figures 1, 2, 3 and 4. In Tables 2 and 3 we display results for each algorithm only with a population size of 100.

The **average runtime (aRT)** used in the tables depends on a given quality indicator value, $I_{\text{target}} = I_{\text{ref}} + \Delta I_{\text{HV}}^{\text{COCO}}$, and is computed over all relevant trials as the number of function evaluations executed during each trial while the best indicator value did not reach I_{target}, summed over all trials and divided by the number of trials that actually reached I_{target} [7]. **Statistical significance** is tested with the rank-sum test for a given target I_{target} using, for each trial, either the number of needed function evaluations to reach I_{target} (inverted and multiplied by -1), or, if the target was not reached, the best $\Delta I_{\text{HV}}^{\text{COCO}}$-value achieved, measured only up to the smallest number of overall function evaluations for any unsuccessful trial under consideration.

The experiments were performed with COCO [6], version 2.2, the plots were produced with version 2.3.1.

For either COMO-CMA-ES or MO-CMA-ES, one can compare the shapes of the empirical cumulative distribution functions (ECDFs) for different population sizes. Note that for each algorithm and each COCO problem, the function is evaluated first with a zero-vector, since we know this produces a point with better f-values than a random sampled point. This allows rigorous comparison of the different algorithms ECDFs.

A glance on the data allows to surmise the following: the larger the population size, the better the reached target precisions are in the long run. However for small budgets, algorithms with smaller population size reach better target precisions. For example in Figure 1 on function **f28**, MO-10 is the first to solve 35% of the targets, MO-32 is the first to solve 45% for a larger budget, and finally MO-100 has the best overall performance with more than 50% of the targets solved. This result is not new and a MO-CMA-ES adapting the population size has been studied for example in [10]. It is still interesting to note that this tendency appears for almost all functions in the test suite.

The same comment can be made on the same function **f28** for COMO-CMA-ES with 3, 10 and 32 kernels: among the COMO-CMA-ES variants, COMO-3 is the first one to solve 40% of the targets, COMO-10 the first to solve 55%, and COMO-32 the first to solve 70%. Note that COMO-32 performs well at a budget of $10^4 n$, compared to the other COMO-CMA-ES. And for a budget of $10^5 n$, COMO-100 and COMO-316 are systematically better. We can expect the variants with more kernels (1000 is proposed here) to solve more targets for longer runs, where the ones with smaller number of kernels will stagnate sooner.

There are some functions where MO-CMA-ES outperforms the best 2016 data for small budgets, say up to $10^2 n$, see **f7, f17, f27, f32** on Figure 1. In contrast the COMO-CMA-ES' ECDFs dynamics are different: compared to the MO-CMA-ES with the same population size, it solves less targets on small budgets; there is a longer-lasting starting phase. This is partly due to the elitist nature of the (1+1)-CMA-ES used by MO-CMA-ES and the non-elitist nature of the

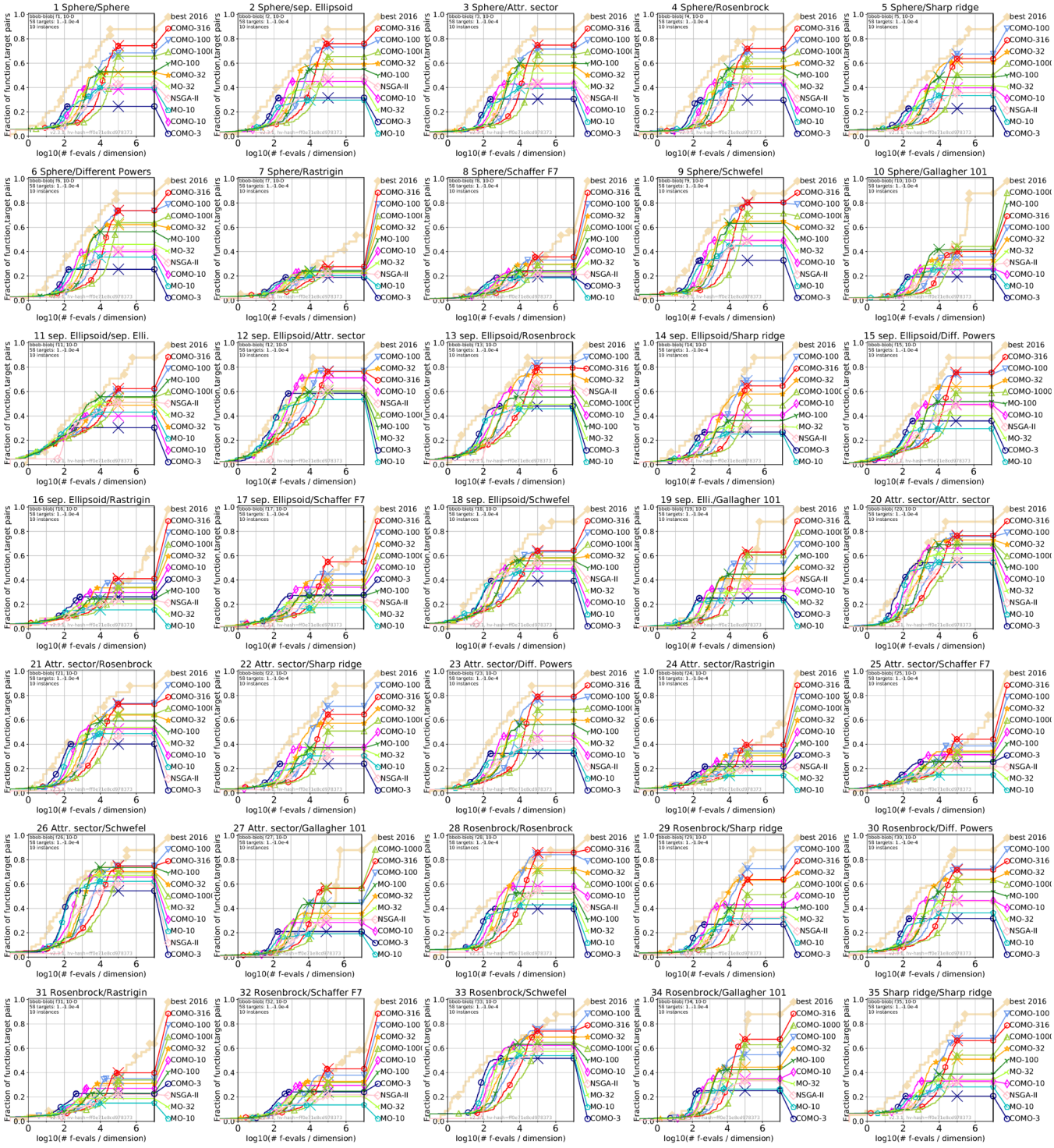

Figure 1: Empirical cumulative distribution of simulated (bootstrapped) runtimes, measured in number of objective function evaluations, divided by dimension (FEvals/DIM) for the 58 targets $\{-10^{-4}, -10^{-4.2}, -10^{-4.4}, -10^{-4.6}, -10^{-4.8}, -10^{-5}, 0, 10^{-5}, 10^{-4.9}, 10^{-4.8}, \ldots, 10^{-0.1}, 10^{0}\}$ in dimension 10.

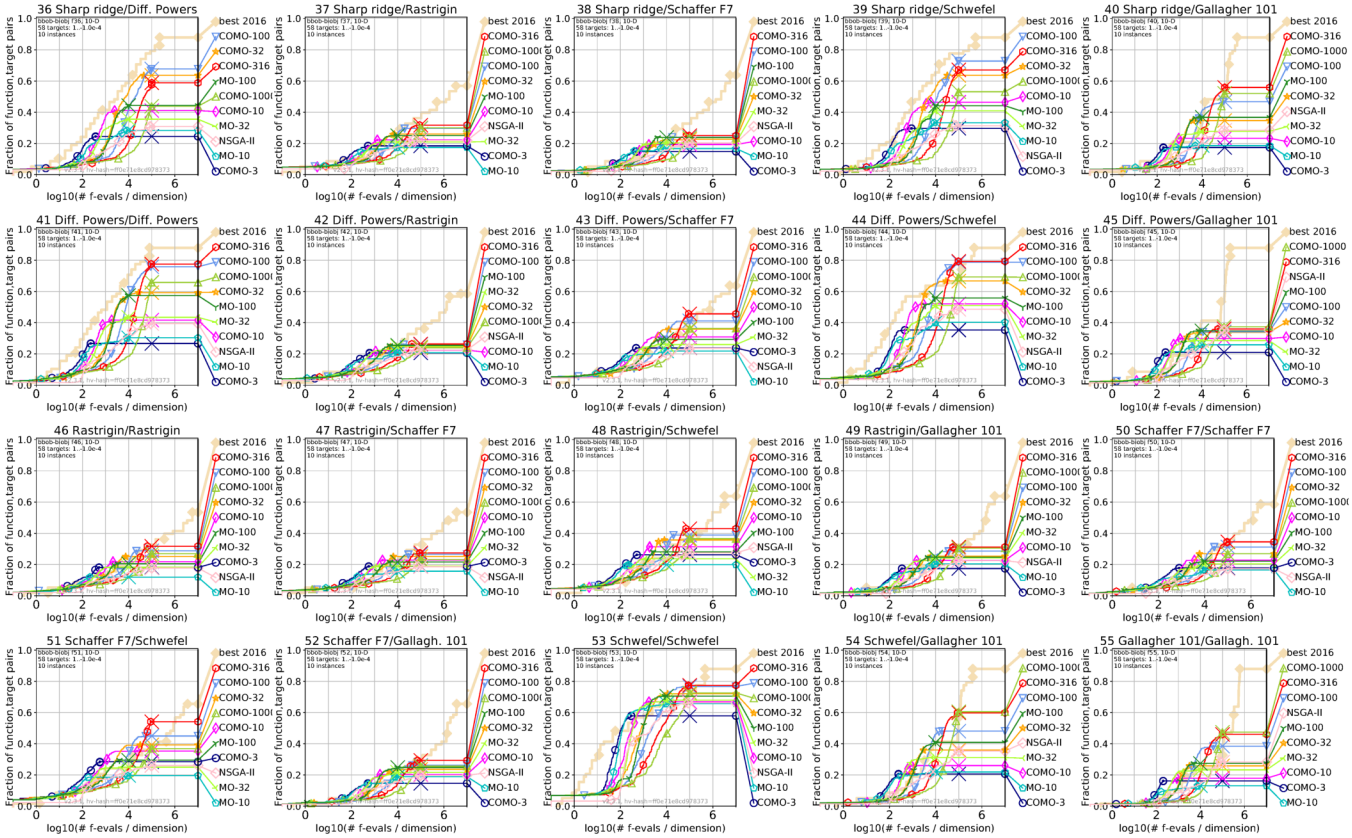

Figure 2: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) as in Fig. 1 but for functions f_{36} to f_{55} in 10-D.

standard CMA-ES used by COMO-CMA-ES. However COMO-CMA-ES performs better in the long run, for instance with a budget of 10^4n on functions **f2, f4, f19, f53** in Figures 1, 2. Note that this effect (starting phase duration and fraction of targets solved) increases with the number of kernels. Remark that the MO-CMA-ES algorithms perform particularly well on problems where one of the functions is the Gallagher 101 (**f40, f45, f49, f52, f54, f55**) where a MO-* systematically outperforms the best2016 reference.

In 5-D on Figure 3, if we look at the fraction of targets found for a budget of 10^4n, a global statement is that NSGA-II is ranking below either MO-100 or COMO-100 (which have the same population size), but it can be above the variants with a population size far from 100. In 20-D, as we can see on Figure 4, for a budget of 10^4n and considering all subgroups of functions, COMO-100 solves more targets than MO-100 which in turn performs better than NSGA-II. As an example, on **ill-cond, ill-cond** subgroup, COMO-100 solves 50% of the target precisions versus roughly 40% for MO-100 and 20% for the NSGA-II. This kind of gap appears in most subgroups with one ill conditioned objective function. COMO-3 and COMO-10 also perform better than best2016 on subgroups with one multimodal objective function for small budgets, between 10^2n and 10^4n.

6 DISCUSSION / CONCLUSION

We have benchmarked COMO-CMA-ES and MO-CMA-ES with different population sizes to particularly test the influence of this parameter on the performances. We also used NSGA-II as a baseline. COMO-CMA-ES performs well although it is *a priori* designed to approximate the optimal p-distribution of the Hypervolume Indicator, for p the population size (number of kernels). As observed in [12], this performance is mainly due to the the large stationary variance obtained with non-elitist (comma) evolution strategies and with the use of UHVI which guides the evolution towards the uncrowded space in the non-dominated region. The overall results for MO-CMA-ES and COMO-CMA-ES show that a smaller population size performs better for smaller budget, while a larger population size end up performing better for a budget large enough. Hence as done in [11] with MO-CMA-ES, a population size adaptation for COMO-CMA-ES should guarantee better performance on COCO.

REFERENCES

[1] Anne Auger, Dimo Brockhoff, Nikolaus Hansen, Dejan Tušar, Tea Tušar, and Tobias Wagner. 2016. Benchmarking MATLAB's gamultiobj (NSGA-II) on the Bi-objective BBOB-2016 Test Suite. In *GECCO 2016 - Genetic and Evolutionary Computation Conference*. ACM, Denver, CO, United States, 1233–1239. DOI: http://dx.doi.org/10.1145/2908961.2931706

[2] D. Brockhoff, T. Tušar, D. Tušar, T. Wagner, N. Hansen, and A. Auger. 2016. Biobjective Performance Assessment with the COCO Platform. *ArXiv e-prints* arXiv:1605.01746 (2016).

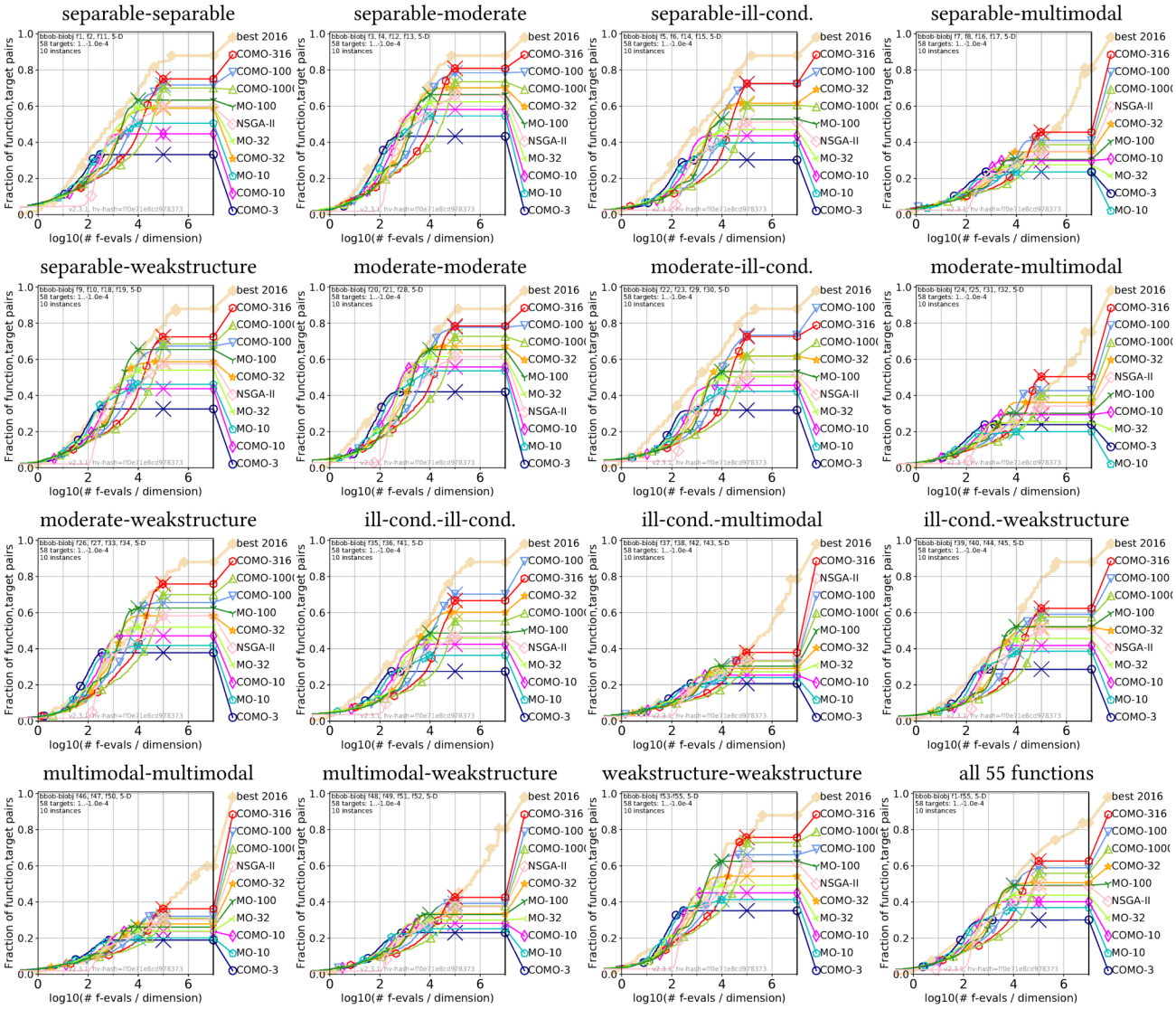

Figure 3: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for 58 targets with target precision in $\{10^{-4}, 10^{-4.2}, 10^{-4.4}, 10^{-4.6}, 10^{-4.8}, 10^{-5}, 0, 10^{-5}, 10^{-4.9}, 10^{-4.8}, \ldots, 10^{-0.1}, 10^{0}\}$ for all functions and subgroups in 5-D. As reference algorithm, the best algorithm from BBOB 2016 is shown as light thick line with diamond markers.

[3] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. 2002. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *Trans. Evol. Comp* 6, 2 (April 2002), 182–197. DOI: http://dx.doi.org/10.1109/4235.996017

[4] Nikolaus Hansen. 2016. The CMA evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772* (2016).

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

[6] N. Hansen, A. Auger, O. Mersmann, T. Tušar, and D. Brockhoff. 2016. COCO: A Platform for Comparing Continuous Optimizers in a Black-Box Setting. *ArXiv e-prints* arXiv:1603.08785 (2016).

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

[8] Christian Igel, Nikolaus Hansen, and Stefan Roth. 2007. Covariance Matrix Adaptation for Multi-objective Optimization. *Evolutionary Computation* 15, 1 (2007), 1–28. DOI: http://dx.doi.org/10.1162/evco.2007.15.1.1 arXiv:https://doi.org/10.1162/evco.2007.15.1.1 PMID: 17388777.

[9] Oswin Krause, Tobias Glasmachers, Nikolaus Hansen, and Christian Igel. 2016. Unbounded population MO-CMA-ES for the bi-objective BBOB test suite. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*. ACM, 1177–1184.

[10] Steffen Limmer and Dietmar Fey. 2016. Investigation of strategies for an increasing population size in multi-objective CMA-ES. In *2016 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, Vancouver, BC, Canada, 476–483. DOI: http://dx.doi.org/10.1109/CEC.2016.7743832

[11] Ilya Loshchilov and Tobias Glasmachers. 2016. Anytime bi-objective optimization with a hybrid multi-objective CMA-ES (HMO-CMA-ES). In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*. ACM, 1169–1176.

[12] Cheikh Toure, Nikolaus Hansen, Anne Auger, and Dimo Brockhoff. 2019. Uncrowded Hypervolume Improvement: COMO-CMA-ES and the Sofomore framework. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, To appear.

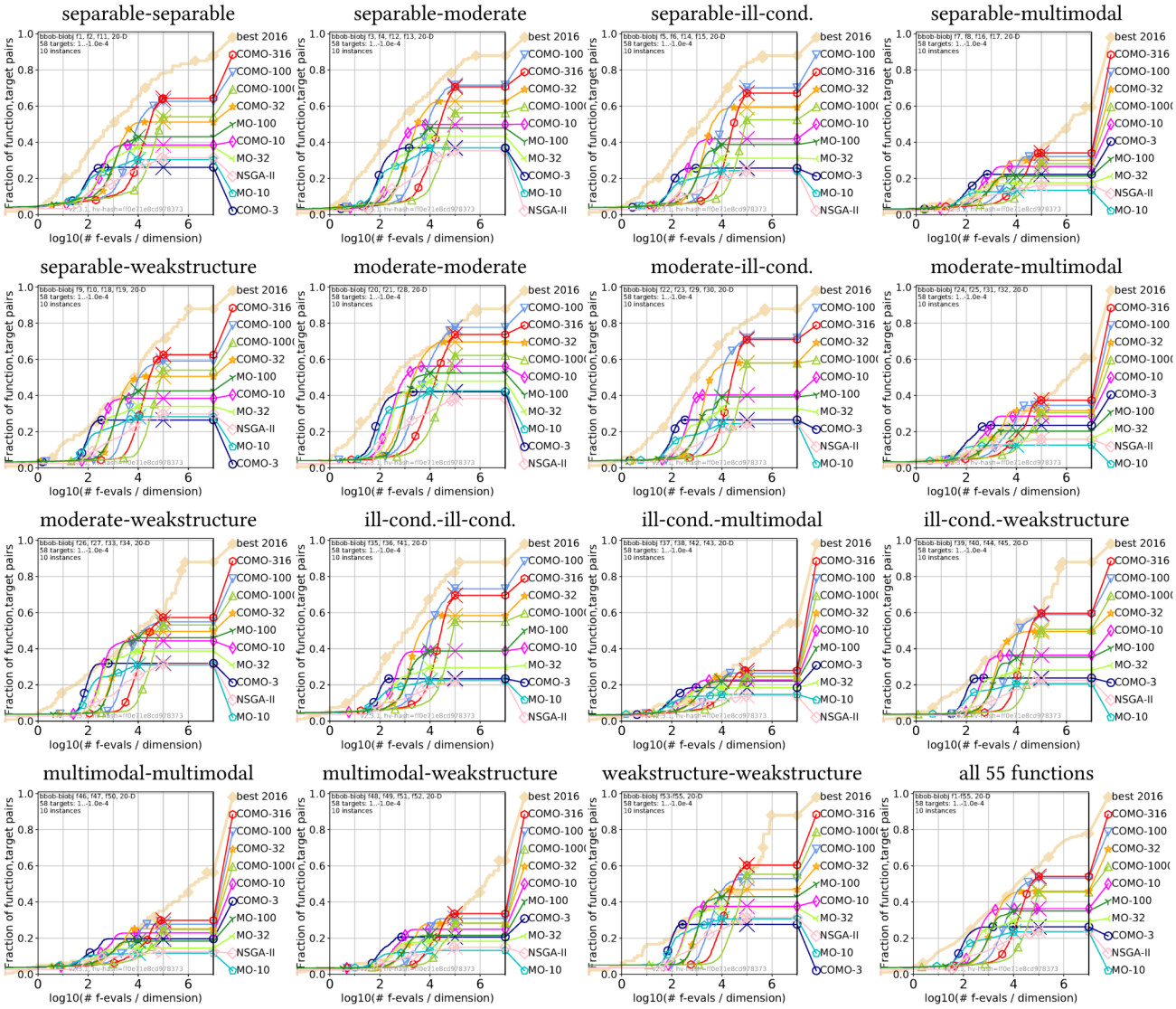

Figure 4: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for 58 targets with target precision in $\{-10^{-4}, -10^{-4.2}, -10^{-4.4}, -10^{-4.6}, -10^{-4.8}, -10^{-5}, 0, 10^{-5}, 10^{-4.9}, 10^{-4.8}, \ldots, 10^{-0.1}, 10^{0}\}$ for all functions and subgroups in 20-D. As reference algorithm, the best algorithm from BBOB 2016 is shown as light thick line with diamond markers.

[13] T. Tušar, D. Brockhoff, N. Hansen, and A. Auger. 2016. COCO: The Bi-objective Black-Box Optimization Benchmarking (bbob-biobj) Test Suite. *ArXiv e-prints* arXiv:1604.00359 (2016).

[14] Thomas Voß, Nikolaus Hansen, and Christian Igel. 2010. Improved Step Size Adaptation for the MO-CMA-ES. In *Genetic And Evolutionary Computation Conference*. ACM, Portland, United States, 487–494. DOI: http://dx.doi.org/10.1145/1830483.1830573

Δf	1e0	1e-1	1e-2	1e-3	#succ
f1	1	75	584	3660	10/10
COMO	**1.2**(0)	20(5)	20(3)	8.7(0.6)	0/10
MO	4.1(8)	12(10)	12(3)	**4.0**(0.1)	0/10
NSII	39(0)	14(2)	11(13)	35(7)	0/10
f2	5.0	105	601	3715	10/10
COMO	5.1(16)	34(38)	22(13)	8.7(3)	0/10
MO	11(23)	18(20)	12(5)	**3.9**(0.6)	0/10
NSII	63(82)	11(1)	**4.1**(2)	36(54)	0/10
f3	3.0	115	665	5170	10/10
COMO	3.1(10)	21(24)	20(6)	6.5(3)	0/10
MO	3.8(0.2)	13(17)	11(3)	**2.9**(0.6)	0/10
NSII	89(153)	22(35)	14(6)	93(116)	0/10
f4	2.0	109	571	2669	10/10
COMO	4.8(0)	15(9)	20(6)	11(2)	0/10
MO	**1.4**(2)	14(13)	12(3)	5.1(0.8)	0/10
NSII	171(211)	10(2)	13(25)	46(12)	0/10
f5	2.0	120	1277	21889	10/10
COMO	**1.1**(0)	27(14)	17(2)	**3.4**(1.0)	0/10
MO	**0.60**(0.2)	19(11)	7.8(1)	3.5(3)	0/10
NSII	86(74)	11(2)	19(19)	∞ *5e5*	0/10
f6	3.0	73	688	3976	10/10
COMO	3.5(8)	34(30)	20(7)	8.5(2)	0/10
MO	**1.2**(3)	22(19)	11(3)	**3.7**(0.5)	0/10
NSII	104(142)	18(7)	11(10)	114(207)	0/10
f7	2.0	2763	1.2e5	3.5e6	0/10
COMO	**1.9**(3)	3.3(4)	6.6(6)	∞ *5e5*	0/10
MO	4.0(18)	**1.7**(1)	**1.7**(0.9)	∞ *5e4*	0/10
NSII	80(81)	3.2(10)	4.2(4)	∞ *5e5*	0/10
f8	3.0	2167	1.6e5	2.1e6	0/10
COMO	19(30)	4.7(3)	**1.9**(0.9)	∞ *5e5*	0/10
MO	**3.9**(7)	**2.5**(0.9)	**1.4**(1)	∞ *5e4*	0/10
NSII	142(122)	**1.0**(0.4)	4.6(4)	∞ *5e5*	0/10
f9	4.0	96	521	1986	10/10
COMO	7.9(0)	18(21)	22(8)	14(2)	0/10
MO	**2.7**(7)	12(14)	12(4)	6.4(1)	0/10
NSII	67(40)	25(73)	11(24)	30(46)	0/10
f10	4.0	323	9839	52107	10/10
COMO	16(48)	13(7)	**2.2**(0.7)	11(7)	0/10
MO	**2.1**(3)	10(4)	**0.90**(0.1)	**0.42**(0.5)	0/10
NSII	95(82)	10(14)	6.1(2)	10(13)	0/10
f11	5.0	56	436	9189	9/10
COMO	4.5(0)	10(37)	12(16)	3.1(3)	0/10
MO	**0.66**(0.5)	4.8(9)	4.6(6)	**1.3**(3)	0/10
NSII	28(54)	11(5)	3.2(4)	5.7(11)	0/10
f12	5.0	44	641	3991	10/10
COMO	**1**(1)	29(21)	11(26)	4.6(8)	6/10
MO	**2.5**(4)	12(5)	4.2(4)	4.5(16)	1/10
NSII	80(51)	23(17)	11(21)	36(94)	5/10
f13	7.0	60	560	5582	10/10
COMO	6.2(11)	15(17)	11(9)	3.6(3)	6/10
MO	8.0(23)	8.5(13)	5.8(6)	3.4(5)	0/10
NSII	65(31)	15(7)	9.4(15)	10(37)	3/10
f14	5.0	247	1713	12801	10/10
COMO	**0.54**(0.4)	27(15)	17(5)	**5.4**(3)	0/10
MO	**1.8**(2)	16(8)	36(38)	∞ *5e4*	0/10
NSII	66(63)	13(2)	45(78)	378(449)	0/10
f15	6.0	74	677	6559	10/10
COMO	10(21)	59(45)	24(10)	5.8(2)	0/10
MO	14(32)	24(25)	11(4)	**3.1**(2)	0/10
NSII	80(93)	32(25)	11(19)	28(27)	0/10
f16	3.0	2810	2.0e5	4.2e6	3/10
COMO	**0.63**(0.7)	**2.5**(2)	**0.27**(0.2)	**0.50**(0.7)	1/10
MO	**1.4**(2)	**1.7**(2)	**2.3**(1)	∞ *5e4*	0/10
NSII	135(123)	4.0(0.4)	6.5(7)	∞ *5e5*	0/10
f17	29	2935	48624	2.4e6	3/10
COMO	51(0.0)	3.3(5)	**2.2**(3)	**0.34**(0.4)	0/10
MO	4.6(11)	**1.4**(2)	**1.8**(3)	∞ *5e4*	0/10
NSII	16(19)	3.4(8)	10(13)	**2.0**(2)	0/10
f18	2.0	56	557	6912	8/10
COMO	4.2(1)	5.6(8)	7.6(9)	4.7(3)	0/10
MO	4.3(12)	3.5(2)	3.5(5)	**1.3**(0.4)	0/10
NSII	158(215)	15(4)	**2.4**(0.8)	41(73)	0/10
f19	9.0	1292	6164	88464	9/10
COMO	4.9(10)	**2.5**(4)	**2.4**(1)	**0.39**(0.2)	1/10
MO	**1.8**(3)	**1.0**(0.8)	**1.1**(0.4)	**0.39**(0.3)	1/10
NSII	67(44)	18(43)	11(16)	**1.8**(2)	0/10

Δf	1e0	1e-1	1e-2	1e-3	#succ
f20	4.0	70	1162	5724	10/10
COMO	33(32)	26(28)	11(6)	6.5(4)	2/10
MO	13(60)	21(20)	5.3(3)	**3.5**(2)	0/10
NSII	761(118)	63(6)	12(23)	153(285)	0/10
f21	5.0	86	3249	9924	10/10
COMO	**3.4**(5)	21(22)	3.5(3)	3.3(2)	3/10
MO	8.8(18)	16(8)	**1.9**(1)	**1.8**(4)	1/10
NSII	56(76)	13(6)	19(41)	20(18)	1/10
f22	3.0	97	1168	12608	9/10
COMO	16(71)	43(23)	22(7)	**5.5**(2)	0/10
MO	**2.3**(5)	22(22)	13(2)	18(18)	0/10
NSII	41(102)	16(5)	25(34)	∞ *5e5*	0/10
f23	1	59	618	4410	10/10
COMO	**1.9**(0)	21(18)	21(8)	**7.5**(2)	0/10
MO	**1.2**(0.5)	12(6)	11(3)	**3.3**(0.8)	0/10
NSII	202(318)	23(6)	11(11)	143(227)	0/10
f24	5.0	2347	1.8e5	4.1e6	0/10
COMO	**1.3**(1)	4.0(2)	**0.36**(0.0)	∞ *5e5*	0/10
MO	**1**(3)	**2.1**(0.4)	∞	∞ *5e4*	0/10
NSII	71(50)	5.2(14)	5.7(4)	∞ *5e5*	0/10
f25	9.0	3143	1.2e5	2.5e6	3/10
COMO	14(35)	**2.8**(3)	**0.99**(4)	**0.36**(0.4)	0/10
MO	4.5(0.3)	**1.5**(1)	**1.1**(0.6)	∞ *5e4*	0/10
NSII	47(113)	10(6)	7.2(9)	∞ *5e5*	0/10
f26	7.0	59	2182	13673	8/10
COMO	13(30)	21(14)	5.6(6)	5.9(19)	0/10
MO	7.6(14)	12(24)	**1.9**(2)	**1.1**(0.4)	0/10
NSII	88(73)	31(28)	**1.5**(2)	7.5(3)	0/10
f27	6.0	2631	21971	44576	10/10
COMO	27(64)	**1.2**(0.9)	**0.83**(0.4)	5.5(11)	0/10
MO	7.7(17)	**0.70**(0.7)	**0.33**(0.1)	**0.56**(0.9)	1/10
NSII	386(53)	5.5(9)	12(9)	15(11)	0/10
f28	2.0	20	145	1230	9/10
COMO	7.4(1)	36(50)	35(19)	16(6)	8/10
MO	12(0)	30(64)	25(7)	8.0(1)	0/10
NSII	95(96)	46(11)	16(15)	27(23)	0/10
f29	3.0	114	1413	13660	10/10
COMO	41(0.2)	54(46)	18(6)	**5.8**(2)	0/10
MO	14(68)	29(17)	20(22)	34(46)	0/10
NSII	140(110)	36(6)	29(32)	∞ *5e5*	0/10
f30	1	33	366	2294	10/10
COMO	**2.5**(8)	41(66)	34(13)	13(3)	2/10
MO	3.9(7)	33(42)	17(2)	**6.3**(5)	0/10
NSII	155(384)	92(9)	42(72)	71(102)	0/10
f31	3.0	2166	50028	3.2e6	0/10
COMO	7.4(6)	4.2(4)	**1.1**(0.4)	∞ *5e5*	0/10
MO	19(46)	**2.3**(1)	9.2(9)	∞ *5e4*	0/10
NSII	71(118)	7.0(11)	10(6)	∞ *5e5*	0/10
f32	3.0	2131	1.1e5	2.4e6	1/10
COMO	5.3(21)	**2.7**(2)	**0.53**(0.2)	∞ *5e5*	0/10
MO	8.8(21)	**1.6**(1)	**1.1**(3)	∞ *5e4*	0/10
NSII	73(87)	5.1(11)	8.2(8)	∞ *5e5*	0/10
f33	6.0	627	1214	3730	7/10
COMO	**10**(4)	90(0.8)	50(108)	20(68)	2/10
MO	26(18)	10(41)	**6.7**(1)	**5.1**(7)	0/10
NSII	64(71)	90(0.4)	47(105)	18(67)	1/10
f34	9.0	1376	15575	54195	10/10
COMO	**1.5**(3)	**2.2**(2)	**1.1**(0.4)	**3.0**(5)	0/10
MO	**1.8**(4)	**1.6**(1)	**0.51**(0.1)	**0.89**(2)	0/10
NSII	21(32)	**1.6**(2)	5.2(5)	15(13)	0/10
f35	1	141	2268	51388	8/10
COMO	3.7(6)	28(24)	10(3)	**1.7**(0.6)	0/10
MO	3.7(13)	22(12)	21(21)	10(7)	0/10
NSII	64(158)	9.5(3)	29(35)	∞ *5e5*	0/10
f36	2.0	204	4640	41237	10/10
COMO	4.2(16)	32(16)	8.3(4)	**3.0**(0.7)	0/10
MO	7.0(3)	18(6)	3.3(1)	5.6(7)	0/10
NSII	91(164)	12(14)	20(7)	∞ *5e5*	0/10
f37	2.0	3956	1.5e5	2.5e6	1/10
COMO	**1**(0)	3.2(1)	**2.0**(2)	∞ *5e5*	0/10
MO	**0.55**(0)	**1.5**(1)	3.2(4)	∞ *5e4*	0/10
NSII	73(121)	**1.0**(0.9)	5.0(7)	∞ *5e5*	0/10

Δf	1e0	1e-1	1e-2	1e-3	#succ
f38	4.0	4366	2.7e5	5.4e6	1/10
COMO	**1.0**(0.8)	4.6(2)	17(35)	∞ *5e5*	0/10
MO	**2.3**(5)	**1.7**(0.6)	**1.8**(2)	∞ *5e4*	0/10
NSII	88(76)	**2.2**(2)	5.2(4)	∞ *5e5*	0/10
f39	2.0	215	1160	47472	8/10
COMO	5.0(11)	19(12)	22(8)	**1.9**(0.8)	0/10
MO	**1.4**(1)	10(6)	14(2)	10(4)	0/10
NSII	95(144)	6.8(2)	19(22)	31(16)	0/10
f40	2.0	998	34442	2.0e5	8/10
COMO	**2.9**(4)	9.3(3)	**2.6**(7)	11(16)	0/10
MO	**1.7**(1)	4.5(1)	**0.53**(0.4)	**2.4**(5)	0/10
NSII	161(173)	**2.7**(1)	4.1(11)	∞ *5e5*	0/10
f41	2.0	48	708	6343	10/10
COMO	**4.0**(7)	53(53)	20(11)	5.5(2)	0/10
MO	21(32)	43(42)	11(3)	**2.3**(0.4)	0/10
NSII	173(185)	29(20)	12(6)	81(81)	0/10
f42	2.0	2525	3.4e5	4.3e6	0/10
COMO	**3.9**(1)	26(102)	13(18)	∞ *5e5*	0/10
MO	21(50)	**2.0**(2)	**1.4**(2)	∞ *5e4*	0/10
NSII	220(200)	4.3(8)	**2.8**(1)	∞ *5e5*	0/10
f43	4.0	2468	1.5e5	3.8e6	2/10
COMO	**2.2**(3)	3.1(3)	**1.8**(3)	∞ *5e5*	0/10
MO	4.1(2)	**1.5**(2)	**0.62**(0.9)	∞ *5e4*	0/10
NSII	65(99)	6.8(6)	4.3(4)	∞ *5e5*	0/10
f44	6.0	86	513	1853	10/10
COMO	23(91)	34(77)	28(19)	17(5)	1/10
MO	15(50)	16(32)	13(3)	**7.4**(2)	0/10
NSII	80(117)	14(9)	7.4(10)	34(31)	0/10
f45	4.0	816	5730	53653	10/10
COMO	3.7(0.9)	5.8(2)	14(0.7)	23(54)	0/10
MO	4.4(10)	4.1(2)	**2.6**(2)	**0.71**(0.5)	0/10
NSII	53(90)	**2.3**(1)	10(7)	90(77)	0/10
f46	4.0	11925	4.9e5	5.6e7	0/10
COMO	**3.0**(8)	**2.3**(1.0)	**2.7**(3)	∞ *5e5*	0/10
MO	**3.9**(6)	**0.78**(0.4)	∞	∞ *5e4*	0/10
NSII	88(99)	3.5(4)	∞	∞ *5e5*	0/10
f47	3.0	4172	2.7e5	4.4e6	0/10
COMO	**1.2**(2)	4.6(4)	**7.7**(13)	∞ *5e5*	0/10
MO	**1.6**(1)	**1.6**(0.8)	∞	∞ *5e4*	0/10
NSII	67(91)	6.8(14)	17(28)	∞ *5e5*	0/10
f48	9.0	2160	1.1e5	2.1e6	2/10
COMO	8.8(6)	6.7(8)	**2.3**(2)	∞ *5e5*	0/10
MO	**3.0**(7)	**2.0**(2)	**1.9**(3)	∞ *5e4*	0/10
NSII	35(57)	11(52)	**2.7**(5)	∞ *5e5*	0/10
f49	9.0	3737	2.0e5	1.3e6	1/10
COMO	10(45)	18(35)	10(21)	∞ *5e5*	0/10
MO	4.6(5)	**1.4**(0.9)	**1.1**(1)	∞ *5e4*	0/10
NSII	42(54)	7.9(3)	5.2(3)	**3.5**(7)	0/10
f50	6.0	3913	2.2e5	2.6e6	1/10
COMO	**0.93**(0.7)	3.7(6)	**1.4**(2)	∞ *5e5*	0/10
MO	**1.4**(2)	**1.6**(1.0)	**2.1**(3)	∞ *5e4*	0/10
NSII	39(61)	**2.4**(4)	5.8(4)	∞ *5e5*	0/10
f51	8.0	1251	32465	2.5e6	1/10
COMO	3.3(3)	5.4(7)	**1.7**(0.6)	**0.86**(2)	1/10
MO	**2.4**(5)	**2.8**(3)	**1.9**(4)	∞ *5e4*	0/10
NSII	53(51)	**2.6**(0.2)	11(11)	∞ *5e5*	0/10
f52	2.0	3027	1.4e5	2.1e6	1/10
COMO	**2.1**(4)	**2.6**(2)	**1.2**(2)	**2.2**(3)	0/10
MO	3.4(3)	**1.2**(0.6)	**0.20**(0.2)	∞ *5e4*	0/10
NSII	133(171)	3.2(6)	3.0(3)	∞ *5e5*	0/10
f53	2.0	13	42	1443	2/10
COMO	9.1(21)	12(11)	29(9)	42(90)	1/10
MO	6.8(16)	17(12)	27(14)	6.1(10)	0/10
NSII	116(229)	62(8)	29(19)	40(87)	1/10
f54	98	1514	12856	45502	10/10
COMO	**2.1**(0.0)	**1.8**(1)	**1.2**(0.4)	**0.72**(0.2)	0/10
MO	**0.94**(0.0)	**0.74**(0.8)	**0.60**(0.1)	**0.74**(0.6)	1/10
NSII	23(102)	**2.0**(7)	4.5(4)	7.3(6)	0/10
f55	3.0	1415	18086	51245	9/10
COMO	**2.2**(8)	5.2(2)	**1.2**(0.4)	7.4(17)	0/10
MO	6.0(3)	**2.4**(1)	**2.3**(4)	**2.6**(3)	0/10
NSII	98(121)	7.7(9)	10(8)	27(17)	0/10

Table 2: Average runtime (aRT in number of function evaluations) divided by the respective best aRT measured during BBOB-2016 in dimension 5. This aRT 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 aRT in the first row. The different target $\Delta I_{\mathrm{HV}}^{\mathrm{COCO}}$-values are shown in the top row. #succ is the number of trials that reached the (final) target $I_{\mathrm{ref}} + 10^{-5}$. 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 (55). A ↓ indicates the same tested against the best algorithm from BBOB 2016. Best results are printed in bold.

Data produced with COCO v2.3.1, hv-hash=ff0e71e8cd978373

Δf	1e0	1e-1	1e-2	1e-3	#succ
f1	1	157	1468	8244	10/10
COMO	**1**(0)	189(19)	48(0.7)	**17**(0.4)	0/10
MO	**1**(0)	101(13)	25(1)	∞ *2e5*	0/10
NSII	**1**(0)	702(1343)	∞	∞ *2e6*	0/10
f2	5.0	163	2170	21153	10/10
COMO	**2.0**(0)	237(61)	42(9)	**9.2**(1)	0/10
MO	8.1(20)	86(9)	16(2)	∞ *2e5*	0/10
NSII	50(83)	654(455)	8459(7374)	∞ *2e6*	0/10
f3	1	216	2384	11104	10/10
COMO	**1**(0)	163(40)	32(3)	**14**(2)	0/10
MO	**1**(0)	71(9)	18(12)	55(40)	0/10
NSII	**1**(0)	663(671)	∞	∞ *2e6*	0/10
f4	1	172	1682	10200	9/10
COMO	**1**(0)	210(30)	45(2)	**14**(0.4)	0/10
MO	**1**(0)	93(9)	29(9)	∞ *2e5*	0/10
NSII	**1**(0)	504(504)	1.1e4(1e4)	∞ *2e6*	0/10
f5	1	190	3207	32860	10/10
COMO	**1**(0)	245(31)	35(3)	**7.9**(1)	0/10
MO	**1**(0)	102(13)	19(8)	∞ *2e5*	0/10
NSII	**1**(0)	1635(2105)	∞	∞ *2e6*	0/10
f6	3.0	236	2134	16568	10/10
COMO	**2.2**(0)	150(13)	39(5)	**10**(0.7)	0/10
MO	3.4(16)	66(10)	17(1)	∞ *2e5*	0/10
NSII	72(171)	459(646)	∞	∞ *2e6*	0/10
f7	1	24981	6.8e5	3.8e7	0/10
COMO	**1**(0)	3.7(2)	27(18)	∞ *2e6*	0/10
MO	**1**(0)	**1.1**(0.5)	∞	∞ *2e5*	0/10
NSII	**1**(0)	326(241)	∞	∞ *2e6*	0/10
f8	4.0	16448	2.4e6	2.8e7	0/10
COMO	162(405)	6.0(2)	∞	∞ *2e6*	0/10
MO	**5.2**(12)	3.4(6)	∞	∞ *2e5*	0/10
NSII	27(37)	1164(942)	∞	∞ *2e6*	0/10
f9	1	144	1468	5944	10/10
COMO	**1**(0)	248(27)	49(2)	**22**(0.7)	8/10
MO	**1**(0)	104(15)	22(2)	48(72)	0/10
NSII	111(550)	375(286)	1654(2268)	∞ *2e6*	0/10
f10	1	6124	2.0e5	2.4e5	10/10
COMO	**1**(0)	9.2(0.5)	**1.7**(0.1)	**20**(25)	0/10
MO	**1**(0)	4.1(1)	**0.56**(0.8)	∞ *2e5*	0/10
NSII	94(464)	223(415)	∞	∞ *2e6*	0/10
f11	3.0	246	3177	1.1e5	1/10
COMO	**1.3**(2)	89(77)	48(19)	**16**(6)	0/10
MO	**2.1**(0)	13(10)	**6.7**(4)	∞ *2e5*	0/10
NSII	84(140)	19(27)	215(458)	∞ *2e6*	0/10
f12	11	131	1483	11435	10/10
COMO	**0.98**(2)	161(106)	76(17)	24(8)	0/10
MO	**1.2**(0.2)	50(11)	13(5)	52(122)	0/10
NSII	38(36)	142(218)	236(329)	779(437)	0/10
f13	7.0	139	1087	17899	7/10
COMO	318(795)	197(94)	70(13)	**13**(4)	0/10
MO	**9.4**(0.8)	61(18)	19(6)	∞ *2e5*	0/10
NSII	77(85)	128(239)	782(1486)	∞ *2e6*	0/10
f14	3.0	349	6102	63789	10/10
COMO	**0.63**(0.8)	180(56)	23(4)	**14**(18)	0/10
MO	**1.4**(0)	68(11)	102(155)	∞ *2e5*	0/10
NSII	112(152)	4054(4803)	∞	∞ *2e6*	0/10
f15	6.0	210	2099	25094	9/10
COMO	6.0(28)	224(49)	53(15)	**8.7**(0.9)	0/10
MO	4.7(2)	53(25)	24(28)	∞ *2e5*	0/10
NSII	74(68)	192(218)	∞	∞ *2e6*	0/10
f16	6.0	32242	1.0e6	1.9e7	0/10
COMO	**1.1**(1)	3.5(2)	**2.4**(3)	∞ *2e6*	0/10
MO	5.1(7)	**1.5**(3)	∞	∞ *2e5*	0/10
NSII	85(78)	84(95)	∞	∞ *2e6*	0/10
f17	1	33890	9.3e5	2.7e7	0/10
COMO	**1**(0)	**2.4**(1)	**1.7**(3)	**0.69**(1)	0/10
MO	**1**(0)	4.4(6)	∞	∞ *2e5*	0/10
NSII	227(298)	139(251)	∞	∞ *2e6*	0/10
f18	6.0	141	1507	17541	10/10
COMO	22(94)	221(73)	54(12)	**17**(4)	0/10
MO	3.8(0)	54(15)	12(1)	52(36)	0/10
NSII	62(88)	27(21)	54(45)	∞ *2e6*	0/10
f19	7.0	18889	2.4e5	5.1e6	6/10
COMO	**1.7**(0.1)	3.3(1)	**1.5**(4)	3.6(4)	0/10
MO	11(50)	**1.1**(0.2)	**2.2**(1)	∞ *2e5*	0/10
NSII	69(72)	42(99)	∞	∞ *2e6*	0/10

Δf	1e0	1e-1	1e-2	1e-3	#succ
f20	3.0	136	1870	8641	10/10
COMO	**0.57**(1)	125(77)	30(30)	26(8)	5/10
MO	6.9(16)	65(15)	10(6)	**8.2**(12)	0/10
NSII	101(75)	56(102)	570(1070)	330(1039)	0/10
f21	5.0	331	1753	9785	8/10
COMO	39(128)	76(26)	39(12)	**22**(6)	3/10
MO	37(92)	34(8)	15(4)	27(24)	0/10
NSII	126(69)	84(133)	1993(2912)	∞ *2e6*	0/10
f22	1	313	4469	23797	10/10
COMO	**1**(0)	160(17)	**26**(2)	**11**(0.6)	0/10
MO	**1**(0)	67(13)	61(25)	∞ *2e5*	0/10
NSII	**1**(0)	6470(5583)	∞	∞ *2e6*	0/10
f23	3.0	332	2660	16975	10/10
COMO	**1.6**(0)	130(23)	37(2)	**13**(0.8)	0/10
MO	**0.47**(0.7)	40(8)	**32**(20)	∞ *2e5*	0/10
NSII	110(184)	710(1248)	7301(6391)	∞ *2e6*	0/10
f24	1	61504	8.1e5	6.5e7	0/10
COMO	963(2404)	**1.4**(1)	23(16)	∞ *2e6*	0/10
MO	26(64)	**0.73**(0.1)	∞	∞ *2e5*	0/10
NSII	589(405)	293(333)	∞	∞ *2e6*	0/10
f25	1	34160	9.2e5	1.2e7	0/10
COMO	**1**(0)	**1.7**(2)	**1.2**(2)	∞ *2e6*	0/10
MO	**1**(0)	**2.9**(5)	∞	∞ *2e5*	0/10
NSII	193(0)	42(59)	∞	∞ *2e6*	0/10
f26	3.0	136	1117	4425	9/10
COMO	615(3072)	195(70)	56(24)	88(140)	1/10
MO	186(0)	76(22)	19(7)	**11**(10)	0/10
NSII	53(24)	62(116)	315(910)	807(1938)	0/10
f27	1	60235	1.3e6	6.4e6	7/10
COMO	**2.3**(6)	4.8(9)	**2.4**(6)	**1.3**(0.8)	0/10
MO	44(217)	**1.2**(2)	**0.26**(0.2)	**0.29**(0.3)	0/10
NSII	400(568)	58(28)	∞	∞ *2e6*	0/10
f28	1	62	539	6898	8/10
COMO	**1**(0)	387(90)	102(8)	**17**(1)	5/10
MO	**1**(0)	180(29)	44(3)	279(370)	0/10
NSII	195(506)	441(309)	2960(1876)	∞ *2e6*	0/10
f29	1	359	3841	30739	8/10
COMO	**1**(0)	154(19)	31(2)	**7.2**(0.8)	0/10
MO	**1**(0)	64(6)	60(40)	∞ *2e5*	0/10
NSII	126(0)	1327(1652)	∞	∞ *2e6*	0/10
f30	1	220	1777	10156	9/10
COMO	7.4(0)	158(62)	47(7)	**15**(1)	0/10
MO	**4.3**(0)	58(22)	25(18)	∞ *2e5*	0/10
NSII	183(229)	293(683)	5082(3377)	∞ *2e6*	0/10
f31	3.0	27722	8.6e5	3.0e7	0/10
COMO	92(229)	3.9(2)	**21**(14)	∞ *2e6*	0/10
MO	**33**(80)	5.8(11)	∞	∞ *2e5*	0/10
NSII	45(110)	657(505)	∞	∞ *2e6*	0/10
f32	6.0	32985	1.1e6	2.6e7	0/10
COMO	289(718)	**2.5**(1)	**1.1**(1)	∞ *2e6*	0/10
MO	111(276)	4.8(8)	∞	∞ *2e5*	0/10
NSII	99(98)	149(80)	∞	∞ *2e6*	0/10
f33	1	33	278	3348	6/10
COMO	**1**(0)	662(192)	184(42)	47(12)	0/10
MO	**1**(0)	289(70)	71(8)	28(60)	0/10
NSII	106(262)	578(498)	2412(2127)	∞ *2e6*	0/10
f34	3.0	45588	1.9e5	2.3e5	8/10
COMO	477(1192)	**1.2**(0.3)	5.2(13)	36(26)	0/10
MO	174(0)	**0.49**(0.1)	**2.0**(3)	∞ *2e5*	0/10
NSII	49(121)	17(11)	∞	∞ *2e6*	0/10
f35	1	338	4782	41429	10/10
COMO	**1**(0)	127(21)	**22**(1)	**5.2**(0.5)	0/10
MO	**1**(0)	61(10)	203(262)	∞ *2e5*	0/10
NSII	**1**(0)	1968(2556)	∞	∞ *2e6*	0/10
f36	1	454	4824	48159	10/10
COMO	**1**(0)	142(28)	30(4)	**6.8**(0.4)	0/10
MO	**1**(0)	48(5)	18(13)	∞ *2e5*	0/10
NSII	**1**(0)	4570(8867)	∞	∞ *2e6*	0/10
f37	1	10528	5.2e5	3.5e7	0/10
COMO	**1**(0)	10(4)	∞	∞ *2e6*	0/10
MO	**1**(0)	**3.4**(1)	∞	∞ *2e5*	0/10
NSII	**1**(0)	∞	∞	∞ *2e6*	0/10

Δf	1e0	1e-1	1e-2	1e-3	#succ
f38	3.0	36705	3.9e6	2.2e8	0/10
COMO	**1.0**(0)	**5.3**(3)	∞	∞ *2e6*	0/10
MO	**1.0**(0)	**2.1**(2)	∞	∞ *2e5*	0/10
NSII	64(84)	∞	∞	∞ *2e6*	0/10
f39	1	457	3876	25958	10/10
COMO	**1**(0)	113(7)	30(3)	**8.5**(0.5)	0/10
MO	**1**(0)	49(10)	53(44)	∞ *2e5*	0/10
NSII	**1**(0)	2584(4838)	∞	∞ *2e6*	0/10
f40	1	26134	2.1e5	2.6e6	7/10
COMO	**1**(0)	**2.3**(0.3)	**7.0**(24)	**1.9**(2)	0/10
MO	**1**(0)	**0.96**(0.2)	9.5(8)	∞ *2e5*	0/10
NSII	**1**(0)	144(99)	∞	∞ *2e6*	0/10
f41	2.0	245	2389	15818	10/10
COMO	42(0)	144(36)	37(5)	**10**(0.9)	1/10
MO	10(47)	51(5)	14(2)	∞ *2e5*	0/10
NSII	107(252)	456(955)	∞	∞ *2e6*	0/10
f42	1	34578	2.9e6	2.3e7	0/10
COMO	**1**(0)	3.3(2)	∞	∞ *2e6*	0/10
MO	**1**(0)	**0.73**(0.3)	∞	∞ *2e5*	0/10
NSII	191(474)	149(295)	∞	∞ *2e6*	0/10
f43	2.0	55957	2.3e6	3.9e7	0/10
COMO	**1.3**(2)	**1.4**(0.5)	**2.1**(2)	∞ *2e6*	0/10
MO	**1.8**(3)	**1.1**(3)	∞	∞ *2e5*	0/10
NSII	122(230)	162(158)	∞	∞ *2e6*	0/10
f44	3.0	168	1485	6584	10/10
COMO	**31**(97)	200(61)	52(10)	**21**(4)	4/10
MO	37(65)	74(36)	31(8)	280(197)	0/10
NSII	93(172)	261(67)	1477(2069)	∞ *2e6*	0/10
f45	1	49797	2.4e6	5.2e6	7/10
COMO	**1.1**(0)	**1.3**(0.5)	**0.88**(1)	**3.5**(4)	0/10
MO	**2.8**(4)	**0.92**(1)	**0.35**(0.2)	∞ *2e5*	0/10
NSII	170(376)	32(27)	∞	∞ *2e6*	0/10
f46	1	55141	1.0e6	1.1e8	0/10
COMO	**1**(0)	**2.8**(1)	∞	∞ *2e6*	0/10
MO	**1**(0)	4.2(6)	∞	∞ *2e5*	0/10
NSII	81(400)	∞	∞	∞ *2e6*	0/10
f47	2.0	54618	2.2e6	5.2e7	0/10
COMO	**0.60**(0)	3.3(0.6)	∞	∞ *2e6*	0/10
MO	**0.55**(0.1)	4.4(4)	∞	∞ *2e5*	0/10
NSII	45(0)	331(494)	∞	∞ *2e6*	0/10
f48	1	22301	3.5e5	3.3e7	0/10
COMO	**1**(0)	3.6(3)	**23**(20)	∞ *2e6*	0/10
MO	**1**(0)	**1.1**(0.4)	∞	∞ *2e5*	0/10
NSII	91(450)	220(474)	∞	∞ *2e6*	0/10
f49	2.0	1.6e5	1.1e7	7.1e7	0/10
COMO	**1.1**(0)	**0.84**(0.3)	∞	∞ *2e6*	0/10
MO	3.4(7)	**0.53**(0.3)	∞	∞ *2e5*	0/10
NSII	52(0)	113(67)	∞	∞ *2e6*	0/10
f50	3.0	22169	1.3e6	2.0e7	0/10
COMO	**0.67**(0.8)	4.8(2)	**6.7**(4)	∞ *2e6*	0/10
MO	**1.5**(4)	8.3(8)	∞	∞ *2e5*	0/10
NSII	72(0)	∞	∞	∞ *2e6*	0/10
f51	1	8574	9.9e5	2.2e7	0/10
COMO	**1**(0)	7.4(3)	**0.28**(0.1)	∞ *2e6*	0/10
MO	**1**(0)	6.7(4)	∞	∞ *2e5*	0/10
NSII	269(486)	566(482)	∞	∞ *2e6*	0/10
f52	1	82166	4.9e6	2.4e7	1/10
COMO	**1**(0)	**1.5**(0.8)	∞	∞ *2e6*	0/10
MO	**1**(0)	**1.0**(1)	∞	∞ *2e5*	0/10
NSII	39(0)	240(438)	∞	∞ *2e6*	0/10
f53	1	28	159	1952	8/10
COMO	**1**(0)	649(139)	250(25)	46(8)	6/10
MO	**1**(0)	246(88)	95(7)	13(2)	0/10
NSII	**1**(0)	123(23)	147(52)	124(50)	0/10
f54	7.0	6049	1.6e5	2.5e6	8/10
COMO	293(1143)	8.5(1)	5.9(12)	**2.0**(5)	0/10
MO	28(69)	3.3(0.5)	**2.1**(2)	∞ *2e5*	0/10
NSII	66(118)	155(177)	∞	∞ *2e6*	0/10
f55	1	8.4e5	8.8e6	8.8e6	7/10
COMO	**1**(0)	**0.37**(0.6)	**0.93**(1)	**2.1**(1)	0/10
MO	**1**(0)	**0.27**(0.4)	**0.21**(0.3)	∞ *2e5*	0/10
NSII	**1**(0)	∞	∞	∞ *2e6*	0/10

Table 3: Average runtime (aRT in number of function evaluations) divided by the respective best aRT measured during BBOB-2016 in dimension 20. This aRT 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 aRT in the first row. The different target $\Delta I_{\mathrm{HV}}^{\mathrm{COCO}}$-values are shown in the top row. #succ is the number of trials that reached the (final) target $I_{\mathrm{ref}} + 10^{-5}$. 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 (55). A ↓ indicates the same tested against the best algorithm from BBOB 2016. Best results are printed in bold.

Data produced with COCO v2.3.1, hv-hash=ff0e71e8cd978373