Anytime Bi-Objective Optimization with a Hybrid Multi-Objective CMA-ES (HMO-CMA-ES)

Ilya Loshchilov
University of Freiburg
Freiburg, Germany
ilya.loshchilov@gmail.com

Tobias Glasmachers
Institut für Neuroinformatik,
Ruhr-Universität Bochum
Bochum, Germany
tobias.glasmachers@ini.rub.de

ABSTRACT

We propose a multi-objective optimization algorithm aimed at achieving good anytime performance over a wide range of problems. Performance is assessed in terms of the hypervolume metric. The algorithm called HMO-CMA-ES represents a hybrid of several old and new variants of CMA-ES, complemented by BOBYQA as a warm start. We benchmark HMO-CMA-ES on the recently introduced bi-objective problem suite of the COCO framework (COmparing Continuous Optimizers), consisting of 55 scalable continuous optimization problems, which is used by the Black-Box Optimization Benchmarking (BBOB) Workshop 2016.

Categories and Subject Descriptors

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

Keywords

Benchmarking, Black-box optimization, Bi-objective optimization

1. INTRODUCTION

The design of anytime optimizers is targeted at achieving good performance for different budgets of computational resources, e.g., ranging from n to $10^6 n$ function evaluations, where n denotes the dimension of the search space. At the same time the black-box optimization paradigm mandates robustness towards problems with vastly differing characteristics. In this work, we followed the approach of [7] where a set of well-performing algorithms was combined to target different classes of problems to achieve good overall anytime performance for single-objective optimization. Here the approach is transferred to multi-objective optimization. This effort requires a careful selection of algorithm components, tuning parameters, and combination strategies.
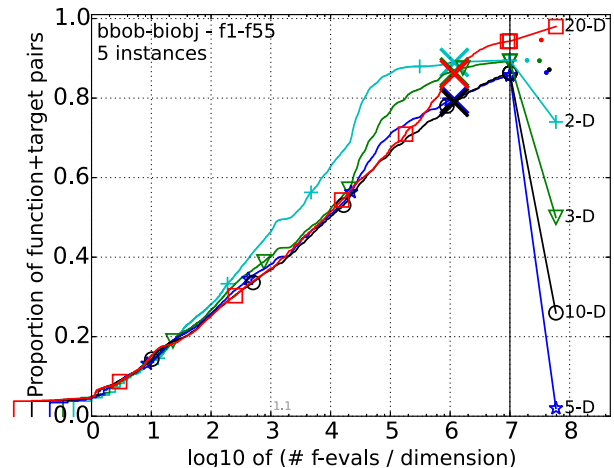
Figure 1: **The aggregated results over all 55 functions for different problem dimensions. See Figure 2 for a more detailed description.**

The proposed Hybrid Multi-Objective Covariance Matrix Adaptation Evolution Strategy (HMO-CMA-ES) consists of the following components:

- BOBYQA [8] on a scalarized objective function as a warm start,

- steady-state multi-objective CMA-ES [5] in our version with increasing population size (ss-MO-CMA-ES),

- our version of CMA-ES with restarts on different scalarized objectives (restart-CMA-ES), and

- generational multi-objective CMA-ES [4] in our version with restarts denoted as IPOP-MO-CMA-ES.

The bi-objective problem suite [9] of the COCO framework consists of 55 classes of bi-objective functions $f_k : \mathbb{R}^n \rightarrow \mathbb{R}^2$, $k \in \{1, \ldots, 55\}$, scalable to any input space dimension $n \geq 2$. Common dimensions for evaluation are $n = 5$ and $n = 20$. The bi-objective functions are formed by combining all 55 combinations of 10 single-objective functions, representing different challenges such as high conditioning number and multi-modality. Five differently parameterized instances of each problem are available for benchmarking, resulting in a total of 275 optimization problems. HMO-CMA-ES is evaluated on this benchmark suite.

2. THE HMO-CMA-ES ALGORITHM

In this section we describe the individual components, their final integration in the HMO-CMA-ES algorithm, and a rationale for the specific design choices. The source code is available at https://sites.google.com/site/hmocmaes/.

2.1 BOBYQA as a Warm Start

BOBYQA is a well-known trust-region method by Michael J. D. Powell [8]. It is well suited for uni-modal problems. It (more exactly, its unconstrained and less advanced variant NEWUOA) is a part of the HCMA algorithm for single-objective optimization [7] that served as inspiration for this work. Its role in HCMA is to solve simple convex-quadratic problems at low cost in the initial phase. In the context of multi-objective optimization we use BOBYQA for a fast approach of the Pareto front. More specifically, we optimize a linear aggregation function $g_\alpha(\boldsymbol{x}) = \alpha f_1(\boldsymbol{x}) + (1-\alpha)f_2(\boldsymbol{x})$ where f_1 and f_2 are the components of the bi-criteria objective function (the two objectives) and $\alpha \in [0,1]$ is an aggregation coefficient. We start BOBYQA with the initial solution $\boldsymbol{x}_{\text{init}} = 0$ in the center of the suggested range for all biobj-BBOB problems. In order to correct for a possible mis-scaling of the objectives we normalize the components of subsequent objective function evaluations:

$$g_\alpha(\boldsymbol{x}) = \alpha \frac{f_1(\boldsymbol{x})}{f_1(\boldsymbol{x}_{\text{init}})} + (1-\alpha)\frac{f_2(\boldsymbol{x})}{f_2(\boldsymbol{x}_{\text{init}})}$$

The first run of BOBYQA stops after at most $5n$ function evaluations or if BOBYQA's relative objective function improvement ratio f_{tol} drops below 10^{-3}. The remaining runs/restarts are launched with different values of α in the following order: $0.5, 0.0, 1.0, 0.95, 0.9, 0.85, \ldots, 0.05, 0.0$. This chain represents a sweep along (convex parts of) the Pareto front, hence the procedure yields a first rough approximation of the front. Further restarts are conducted with a smaller stopping tolerance of $f_{tol} = 10^{-4}$ to improve the approximation. We also decrease the radius of the initial trust-region from 6 (a rather global search in $[-5,5]^n$) to 2 (a rather local search) as each restart with a new value of α is initialized in the best solution of the previous run. These settings for BOBYQA are designed for budgets of up to $100n$ function evaluations. Most of these settings are irrelevant for HMO-CMA-ES, where BOBYQA is run only for $10n$ function evaluations and hence performs only few restarts within this very low budget.

2.2 Steady-state MO-CMA-ES with Increasing Population Size

The initial runs of BOBYQA are expected to find a better-than-random approximates of the Pareto front. We collect all solutions generated by BOBYQA and apply non-dominated sorting with the hypervolume metric as secondary sorting criterion [4]. The five best solutions form the initial population of the steady-state MO-CMA-ES [5], which is started with an initial step size of $\sigma = \frac{1}{2}$. The population size is increased by one every $50n$ iterations. This mechanism achieves a fast approach and a good coverage of the Pareto front. A very similar idea was introduced recently in [1]. In addition we employ a crossover procedure with probability 10%. It randomly selects two solutions $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ and generates an offspring $\boldsymbol{x}_3 \leftarrow \boldsymbol{x}_1 + a(\boldsymbol{x}_2 - \boldsymbol{x}_1)$ with blending coefficient $a \sim \mathcal{N}(\frac{1}{2}, \frac{1}{4})$. The offspring inherits the averaged step-size and covariance matrix from its parents.

2.3 Generational MO-CMA-ES with Restarts

We use a version of generational MO-CMA-ES [4] where we double the population size (initially set to 10) after each restart happening every $50n$ iterations of the algorithm. We denote it as IPOP-MO-CMA-ES (the idea first appeared in [6]). The initial step size is set to $\sigma = 2$.

2.4 CMA-ES with Restarts

We apply CMA-ES with a new restart variant to a linear aggregation of the objective function. In each restart, a new aggregation coefficient α is sampled uniformly from $[0,1]$. The first population consists of $\lambda = \lambda_{\min} = 50$ individuals. At the i-th restart, the population size is sampled as $\lambda \leftarrow \lambda_{\min}(\lambda_{\max}/\lambda_{\min})^b$, where b is drawn from a uniform distribution on $[0,2]$ and $\lambda_{\max} = \lambda_{\min}1.02^i$. This proceeding is inspired by BIPOP-CMA-ES but with a far smaller increase factor of 1.02 compared to 2 in standard BIPOP-CMA-ES. We set the maximum number of iterations to 100×1.02^i.

We had initially planned to use multiple BIPOP-CMA-ES instances [2], each optimizing a different aggregated objective function. This approach would guarantee very good performance for large budgets, but the initialization phase of multiple BIPOP-CMA-ES takes a while and this would negatively impact the anytime performance of the algorithm. The above proposal of a restart CMA-ES with random aggregation coefficient α acts as a replacement.

2.5 HMO-CMA-ES

The proposed Hybrid Multi-objective CMA-ES algorithm is designed to achieve best anytime performance. It has four phases, with a different set of algorithms running. If multiple algorithms are active at the same time then they are running in parallel, in a round-robin fashion.

We start with BOBYQA for the first $10n$ function evaluations (phase 1). The best solutions of BOBYQA are used to initialize the ss-MO-CMA-ES. This algorithm runs until $1,000n$ function evaluations (phase 2). Then we launch restart-CMA-ES to run in parallel to ss-MO-CMA-ES (phase 3) such that the best solution found by each run of restart-CMA-ES is injected as a candidate solution for the next iteration of ss-MO-CMA-ES. After $20,000n$ function evaluations we also launch IPOP-MO-CMA-ES to run in parallel to ss-MO-CMA-ES and restart-CMA-ES (phase 4). With a probability of 10% a random solution from the current population of IPOP-MO-CMA-ES is injected into ss-MO-CMA-ES. The role of ss-MO-CMA-ES is to fine-tune the hypervolume metric. The three algorithms are running for $400,000n$ function evaluations each, comprising the total budget of $1.2 \times 10^6 n$ function evaluations.

3. CPU TIMING

In order to evaluate the CPU timing of the algorithm, we have run HMO-CMA-ES with restarts on the entire bbob-biobj test suite for $1000n$ function evaluations. The C++ code (called from Matlab) was run on one core of Intel(R) Core(TM) i5-4690 CPU @ 3.50GHz. The time per function evaluation for dimensions 2, 3, 5, 10 and 20 equals 40, 37, 36, 39, and 51 microseconds, respectively.

4. RESULTS

Results of HMO-CMA-ES from experiments according to [3] on the benchmark functions given in [9] are presented in Figures 1, 2, 3, 4, and 5, and in Table 1.

For each problem instance, the performance of HMO-CMA-ES is assessed in terms of the hypervolume metric [10] (to be maximized), the Lebesgue measure of the points that are a) dominated by at least one objective vector found by the algorithm, and b) dominate a given reference point. This hypervolume is assessed relative to a reference value, which is the dominated hypervolume of a reference Pareto front consisting of the best known set of objective vectors for this problem. The task of maximizing the hypervolume is equivalent to minimizing the difference between reference hypervolume and achieved hypervolume (to be minimized). The reference hypervolume defines 58 target values for this difference, which are multiples of the reference hypervolume with the factors $\{-10^{-4}, -10^{-4.2}, -10^{-4.4}, -10^{-4.6}, -10^{-4.8}, -10^{-5}, 0, 10^{-5}, 10^{-4.9}, \ldots, 10^0\}$. All results are reported in terms of the fraction of reached target values. This normalization makes the results roughly comparable across different problem types.

Hence, if an algorithm finds a non-dominated front of exactly the same quality as the best known, then it reaches 52 out of 58 targets (including 0). This corresponds to roughly $0.9 \approx \frac{52}{58}$ on the vertical axis of the plots used in this paper, i.e., a curve stopping at around 0.9 suggest that the best known approximation was reached. In most cases, the best known approximation provided by the biobj-BBOB 2016 is very close to the true best value of the Pareto front and thus 0.9 is about the maximum possible value one can reach. This is often the case on uni-modal functions. However, on some multi-modal functions (i.e., when at least one of the objectives is multi-modal) the current best approximation can be further improved and thus an algorithm can reach targets with negative factors corresponding to better hypervolume values than the reference of biobj-BBOB 2016. Indeed, the introduction of negative targets was motivated by the fact that the current known approximations are not the best possible ones in some cases.

Figure 1 shows the aggregated results over all 55 functions for search space dimension $n = 2, 3, 5, 10, 20$. The saturation at a value of 0.9 can be well observed on 2-dimensional problems, where the best known hypervolume values are indeed very close to the optimal ones. In this case, HMO-CMA-ES solves most of the problems after about $10^5 n$ function evaluations, from where on the curves stagnate. The problems become harder with n, hence more function evaluations are typically needed to reach a similar average performance.

Table 1 shows the average runtime to reach given targets on 5- and 20-dimensional problems. Figures 2, 3, 4 show the empirical cumulative distribution of simulated (bootstrapped) runtimes [3] for all 55 functions and all considered problem dimensions. Some functions can be associated with a much higher variance in the results and with a faster than linear growth of the complexity w.r.t. n. This is often the case for multi-modal functions, but sometimes also appears for ill-conditioned problems. On some (often 20-dimensional) problems the best known hypervolume value can be improved, this happens when the curve crosses the value of about 0.9.

5. CONCLUSION

We have presented a hybrid algorithm for multi-objective optimization, combining of a number of well-performing components for single- and multi-objective optimization. We showed that the proposed algorithm can solve almost all biobj-BBOB problems. When large computational budgets are considered it finds high quality solution sets for nearly all combinations of problem type and search space dimension. We attribute this robustness to the combination of different optimizer components into a hybrid algorithm. The performance relative to other multi-objective algorithms will be known as soon as the results of the biobj-BBOB 2016 edition are available.

The algorithm has a set of hyperparameters, mostly start and stop times (iteration numbers) encoded as multiples of n. Better tuning of these would most probably improve its performance. It may be possible to replace some of the hard numbers with adaptive stopping criteria.

The algorithm can be outperformed on some functions even by its own individual components (the price of hybridization) or when a particular computational budget is considered (the price for its good anytime performance). It should be possible to considerably reduce these effects with online prioritization of individual components depending on their relative performance. However, this step is left for future work.

6. REFERENCES

[1] T. Glasmachers, B. Naujoks, and G. Rudolph. Start Small, Grow Big? Saving Multi-objective Function Evaluations. In *Parallel Problem Solving from Nature–PPSN XIII*, pages 579–588. Springer, 2014.

[2] N. Hansen. Benchmarking a BI-population CMA-ES on the BBOB-2009 function testbed. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, pages 2389–2396. ACM, 2009.

[3] N. Hansen, T. Tusar, O. Mersmann, A. Auger, and D. Brockhoff. COCO: The Experimental Procedure. Technical Report arXiv:1603.08776, arxig.org, 2016.

[4] C. Igel, N. Hansen, and S. Roth. Covariance matrix adaptation for multi-objective optimization. *Evolutionary computation*, 15(1):1–28, 2007.

[5] C. Igel, T. Suttorp, and N. Hansen. Steady-state selection and efficient covariance matrix update in the multi-objective CMA-ES. In *Evolutionary Multi-Criterion Optimization*, pages 171–185. Springer, 2007.

[6] I. Loshchilov and M. Pilát. Personal communication, 2013.

[7] I. Loshchilov, M. Schoenauer, and M. Sebag. Bi-population CMA-ES algorithms with surrogate models and line searches. In *Proceedings of the 15th annual conference companion on Genetic and evolutionary computation*, pages 1177–1184. ACM, 2013.

[8] M. J. Powell. The BOBYQA algorithm for bound constrained optimization without derivatives. 2009.

[9] T. Tusar, D. Brockhoff, N. Hansen, and A. Auger. COCO: The Bi-objective Black Box Optimization Benchmarking (bbob-biobj) Test Suite. Technical Report arXiv:1604.00359, arxig.org, 2016.

5-D

Δf	1e+0	1e-1	1e-2	1e-3	1e-4	1e-5	#succ
f_1	1	76(34)	622(166)	3680(266)	38122(1604)	6.1e5(9e5)	5/5
f_2	3.0(2)	137(23)	675(224)	3282(1538)	37044(13668)	3.0e5(1e5)	5/5
f_3	1	92(92)	706(298)	5848(4912)	44638(31038)	3.0e5(2e5)	5/5
f_4	1	105(26)	573(237)	2722(1446)	30360(7352)	3.0e5(81053)	5/5
f_5	1	107(26)	1246(393)	26575(38218)	1.2e5(1e5)	6.8e5(3e5)	5/5
f_6	1	55(20)	698(424)	3951(1096)	47723(10868)	2.9e6(52016)	5/5
f_7	1	698(588)	1.1e5(1e5)	4.9e6(1e7)	∞	∞5.8e6	0/5
f_8	2.8(2)	2622(4735)	1.7e5(2e5)	2.0e6(2e6)	2.7e7(2e7)	∞5.8e6	0/5
f_9	1	104(67)	525(135)	2004(377)	20926(604)	4.4e5(3e5)	5/5
f_{10}	1	390(492)	46136(82956)	1.2e5(1e5)	1.6e5(1e5)	4.1e5(35852)	5/5
f_{11}	1	56(37)	519(598)	9095(14876)	1.3e5(1e5)	9.7e5(2e5)	5/5
f_{12}	1	53(46)	584(943)	6412(6691)	28731(51373)	1.6e5(2e5)	5/5
f_{13}	2.0(2)	26(22)	300(608)	3749(4303)	26314(44084)	86884(1e5)	5/5
f_{14}	1	283(244)	2094(1206)	15271(4918)	1.3e5(35230)	3.9e5(3e5)	5/5
f_{15}	4.8(5)	101(55)	874(538)	6392(3528)	51584(21449)	3.1e5(1e5)	5/5
f_{16}	1	3283(7912)	98937(1e5)	5.1e6(7e6)	2.6e7(2e7)	2.6e7(3e7)	1/5
f_{17}	48(110)	8237(7601)	1.1e5(42161)	3.0e6(4e6)	2.5e7(2e7)	2.7e7(4e7)	1/5
f_{18}	1	62(74)	881(1333)	7694(9780)	1.2e5(89543)	2.0e6(5e6)	4/5
f_{19}	4.8(4)	1449(2646)	51687(51038)	1.5e5(1e5)	2.7e5(5e5)	1.9e6(2e6)	4/5
f_{20}	2.0(2)	43(46)	587(512)	5553(5173)	33785(32218)	2.4e5(1e5)	5/5
f_{21}	1	149(130)	13829(33339)	34787(77474)	60168(78548)	2.2e5(81198)	5/5
f_{22}	1	95(26)	1275(457)	9521(1078)	75476(13788)	3.9e5(2e5)	5/5
f_{23}	1	61(35)	662(132)	3948(1621)	38675(19852)	2.8e5(76170)	5/5
f_{24}	2.2(3)	536(113)	1.3e5(25555)	3.4e6(1e6)	∞	∞5.8e6	0/5
f_{25}	2.4(4)	10730(11800)	1.3e5(1e5)	2.2e6(2e6)	1.2e7(3e7)	2.7e7(3e7)	1/5
f_{26}	2.4(4)	102(141)	537(452)	1672(682)	3.8e5(8e5)	4.1e6(4e6)	3/5
f_{27}	1	2443(2837)	83976(59564)	1.9e5(1e5)	3.2e5(2e5)	7.4e5(7e5)	5/5
f_{28}	1	24(5)	152(98)	1177(514)	90425(1e5)	1.5e6(1e6)	5/5
f_{29}	1	84(63)	1774(1312)	16753(12980)	1.3e5(49966)	4.3e5(1e5)	5/5
f_{30}	1	34(9)	476(330)	2851(1191)	86136(84655)	4.0e5(6e5)	5/5
f_{31}	1	1371(2668)	67643(66756)	3.0e6(4e6)	2.8e7(2e7)	∞5.9e6	0/5
f_{32}	2.4(4)	5032(7620)	93375(55990)	2.0e6(2e6)	1.3e7(9e6)	2.9e7(4e7)	1/5
f_{33}	2.0(1)	11(4)	70(54)	491(297)	3.6e5(8e5)	5.2e6(9e6)	3/5
f_{34}	1	222(192)	43144(54731)	2.5e5(1e5)	2.9e5(63528)	8.2e5(1e6)	5/5
f_{35}	1	102(34)	2666(2738)	55515(71183)	3.5e5(3e5)	1.5e6(1e5)	5/5
f_{36}	1	279(208)	7123(14276)	40761(37216)	1.3e5(1e5)	4.8e5(89917)	5/5
f_{37}	1	932(310)	1.7e5(1e5)	3.3e6(2e6)	2.7e7(3e7)	∞5.9e6	0/5
f_{38}	1	27778(51968)	3.1e5(2e5)	1.2e7(1e7)	∞	∞5.9e6	0/5
f_{39}	1	213(156)	1043(318)	89093(1e5)	2.6e5(3e5)	7.0e5(5e5)	5/5
f_{40}	1	1505(406)	18105(35332)	1.8e5(2e5)	3.7e5(2e5)	1.6e6(1e6)	5/5
f_{41}	1	38(35)	633(170)	5997(4212)	42241(10062)	2.6e5(1e5)	5/5
f_{42}	1	648(629)	2.4e5(4e5)	4.8e6(3e6)	∞	∞5.9e6	0/5
f_{43}	1.8	25272(30125)	2.0e5(2e5)	4.9e6(6e6)	2.6e7(2e7)	2.7e7(5e7)	1/5
f_{44}	1	51(19)	540(256)	1733(445)	36645(48512)	9.6e5(5e5)	5/5
f_{45}	1	256(124)	56060(55916)	1.3e5(97388)	1.8e5(1e5)	4.3e5(1e5)	5/5
f_{46}	1	14236(2722)	4.7e5(4e5)	∞	∞	∞6.0e6	0/5
f_{47}	1	14416(11142)	2.7e5(2e5)	6.4e6(5e6)	2.7e7(1e7)	2.8e7(2e7)	1/5
f_{48}	1	11373(24814)	95569(1e5)	1.0e6(9e5)	9.7e6(9e6)	1.1e7(2e7)	2/5
f_{49}	1	2650(3412)	1.8e5(74726)	5.9e5(3e5)	1.2e7(2e7)	2.7e7(4e7)	1/5
f_{50}	1	11162(8061)	1.6e5(30361)	2.9e6(5e6)	∞	∞5.9e6	0/5
f_{51}	1	7873(13724)	1.6e5(1e5)	4.8e6(4e6)	2.7e7(5e7)	2.7e7(3e7)	1/5
f_{52}	1	10582(4914)	1.5e5(44882)	3.6e6(4e6)	1.3e7(2e7)	∞5.8e6	0/5
f_{53}	2.4(2)	16(18)	50(22)	38574(96032)	88948(1e5)	2.3e6(3e6)	4/5
f_{54}	1	89(138)	44791(18158)	1.3e5(78937)	1.5e5(1e5)	2.6e5(80522)	5/5
f_{55}	1.2(0.5)	16706(14289)	1.3e5(81033)	1.7e5(81270)	2.4e5(2e5)	3.1e5(2e5)	5/5

20-D

Δf	1e+0	1e-1	1e-2	1e-3	1e-4	1e-5	#succ
f_1	1	173(38)	1414(178)	8267(967)	76089(4616)	3.9e5(21373)	5/5
f_2	1	176(75)	2284(1026)	23672(7342)	2.7e5(60898)	3.1e6(2e6)	5/5
f_3	1	271(222)	2628(847)	11178(2880)	88645(22800)	4.6e5(84986)	5/5
f_4	1	175(78)	1537(453)	9688(1253)	92783(17732)	9.8e5(5e5)	5/5
f_5	1	229(28)	2991(297)	30127(9511)	1.9e5(21931)	8.0e5(49360)	5/5
f_6	1	288(125)	2139(457)	13912(1583)	1.2e5(15284)	5.6e5(58583)	5/5
f_7	1	15922(35563)	4.7e5(5e5)	1.5e7(9e6)	3.3e7(3e7)	1.1e8(2e8)	1/5
f_8	1	40384(34773)	1.4e6(2e6)	1.4e7(2e7)	1.9e7(7e6)	1.9e7(2e7)	4/5
f_9	1	163(46)	1720(559)	6034(1274)	47190(4944)	5.7e5(2e5)	5/5
f_{10}	1	634(472)	2.3e5(3e5)	7.3e5(8e5)	9.3e5(7e5)	1.3e6(7e5)	5/5
f_{11}	1	258(244)	3511(3284)	1.1e5(49142)	3.2e6(2e6)	1.8e7(2e7)	4/5
f_{12}	1	139(46)	1524(1349)	11420(11915)	1.4e5(1e5)	1.9e6(2e6)	5/5
f_{13}	1	175(72)	1419(678)	23144(12096)	9.8e6(2e7)	4.6e7(3e7)	2/5
f_{14}	1	229(171)	6010(2231)	53814(7210)	4.7e5(3e5)	2.2e6(1e6)	5/5
f_{15}	1	173(56)	1775(431)	13552(1835)	2.3e5(92055)	1.0e7(1e7)	4/5
f_{16}	1	31269(10436)	5.4e5(6e5)	5.1e6(4e6)	1.4e7(3e7)	1.4e7(1e7)	4/5
f_{17}	1	48714(39244)	6.0e5(6e5)	4.9e6(4e6)	1.5e7(2e7)	1.6e7(1e7)	4/5
f_{18}	1	164(26)	1567(610)	17826(14184)	5.3e5(5e5)	4.4e6(2e6)	5/5
f_{19}	1	6392(7100)	2.5e6(6e6)	7.4e6(1e7)	8.2e6(1e7)	1.1e7(1e7)	4/5
f_{20}	1	145(98)	2113(2334)	9529(6937)	40822(7234)	4.8e5(67012)	5/5
f_{21}	1	339(192)	1886(2756)	9899(12090)	6.0e6(2e7)	6.9e6(1e6)	4/5
f_{22}	1	236(58)	3727(548)	16569(1849)	1.5e5(9730)	5.9e5(3e5)	5/5
f_{23}	1	236(98)	2311(392)	13117(9686)	94467(34764)	6.2e5(2e5)	5/5
f_{24}	1	15684(21238)	6.6e5(1e6)	7.7e6(5e6)	2.7e7(2e7)	4.3e7(4e7)	2/5
f_{25}	1	49579(76506)	6.9e6(6e5)	7.1e6(4e6)	1.0e7(3e6)	1.1e7(5e6)	5/5
f_{26}	1	107(112)	766(844)	2602(2672)	1.9e5(4e5)	8.0e6(2e7)	4/5
f_{27}	1	2.1e5(5e5)	1.5e6(2e6)	5.4e6(1e7)	7.3e6(1e7)	8.0e6(2e7)	4/5
f_{28}	1	56(34)	353(140)	7009(4906)	2.9e5(4e5)	1.0e7(2e7)	4/5
f_{29}	1	407(139)	4107(679)	28043(7530)	2.1e5(67076)	1.4e6(9e5)	5/5
f_{30}	1	117(32)	1880(765)	11138(4587)	1.7e5(21871)	8.2e6(8e6)	4/5
f_{31}	1	28458(8210)	4.6e5(3e5)	1.8e7(2e7)	2.5e7(2e7)	4.0e7(4e7)	2/5
f_{32}	1	32778(26356)	4.7e5(3e5)	6.7e6(2e6)	1.6e7(5e6)	2.1e7(1e7)	4/5
f_{33}	1	29(7)	212(16)	3520(1434)	5.7e5(5e5)	1.7e7(1e7)	4/5
f_{34}	1	540(132)	7.5e6(1e7)	9.3e6(1e7)	9.6e6(8e6)	2.1e7(3e7)	3/5
f_{35}	1	410(162)	4748(1141)	41160(7250)	2.4e5(79726)	7.7e5(5e5)	5/5
f_{36}	1	569(432)	4542(1043)	34402(10079)	2.8e5(2e5)	1.5e6(9e5)	5/5
f_{37}	1	7210(6640)	4.7e5(2e5)	1.1e7(8e6)	1.0e8(1e8)	1.0e8(2e8)	1/5
f_{38}	1	84291(79832)	1.5e6(8e5)	1.1e7(5e6)	2.7e7(3e7)	2.7e7(2e7)	3/5
f_{39}	1	488(352)	4413(1268)	27408(5738)	1.6e5(23793)	9.1e5(4e5)	5/5
f_{40}	1	42097(62640)	6.4e6(6e6)	1.7e7(3e7)	1.7e7(2e7)	1.7e7(2e7)	3/5
f_{41}	1	285(276)	2579(846)	16752(8787)	1.2e5(37074)	6.3e5(4e5)	5/5
f_{42}	1	48328(55502)	2.9e6(3e6)	4.3e7(2e7)	4.6e7(4e7)	4.6e7(4e7)	2/5
f_{43}	1	96614(73918)	2.0e6(1e6)	1.2e7(4e6)	2.2e7(6e6)	2.3e7(2e7)	4/5
f_{44}	1	171(74)	1490(414)	46162(1838)	40596(16886)	7.4e5(5e5)	5/5
f_{45}	1	54016(88648)	1.4e6(2e6)	1.0e7(1e7)	2.1e7(2e7)	2.1e7(3e7)	3/5
f_{46}	1	48335(13908)	7.4e5(3e5)	1.2e7(1e7)	1.7e7(1e7)	1.9e7(2e7)	4/5
f_{47}	1	60949(16417)	1.1e6(7e5)	7.2e6(6e6)	8.7e6(7e6)	9.2e6(7e6)	5/5
f_{48}	1	37048(18562)	3.8e5(2e5)	6.9e6(3e6)	2.4e7(3e7)	2.5e7(2e7)	3/5
f_{49}	1	2.5e5(2e5)	1.9e7(3e7)	∞	∞	∞2.3e7	0/5
f_{50}	1	71298(61799)	6.3e5(2e5)	5.0e6(2e6)	7.3e6(4e6)	7.4e6(4e6)	5/5
f_{51}	1	46809(1020)	7.1e5(5e5)	6.4e6(4e6)	1.5e7(1e7)	1.6e7(3e7)	4/5
f_{52}	1	87993(60134)	4.4e6(2e6)	2.6e7(3e7)	2.7e7(3e7)	2.7e7(5e7)	3/5
f_{53}	1	29(6)	160(28)	1905(795)	17127(20882)	1.9e7(5e7)	3/5
f_{54}	1	5005(5867)	7.4e5(5e5)	1.5e7(3e7)	2.2e7(2e7)	4.2e7(4e7)	2/5
f_{55}	1	2.3e5(3e5)	6.4e6(1e7)	7.0e6(1e7)	7.1e6(1e7)	7.1e6(6e6)	4/5

Table 1: **Average runtime (aRT) to reach given targets, measured in number of function evaluations. For each function, the aRT and, in braces as dispersion measure, the half difference between 10 and 90%-tile of (bootstrapped) runtimes is shown for the different target Δf-values as shown in the top row. #succ is the number of trials that reached the last target $HV_{\mathrm{ref}} + 10^{-5}$. The median number of conducted function evaluations is additionally given in *italics*, if the target in the last column was never reached.**
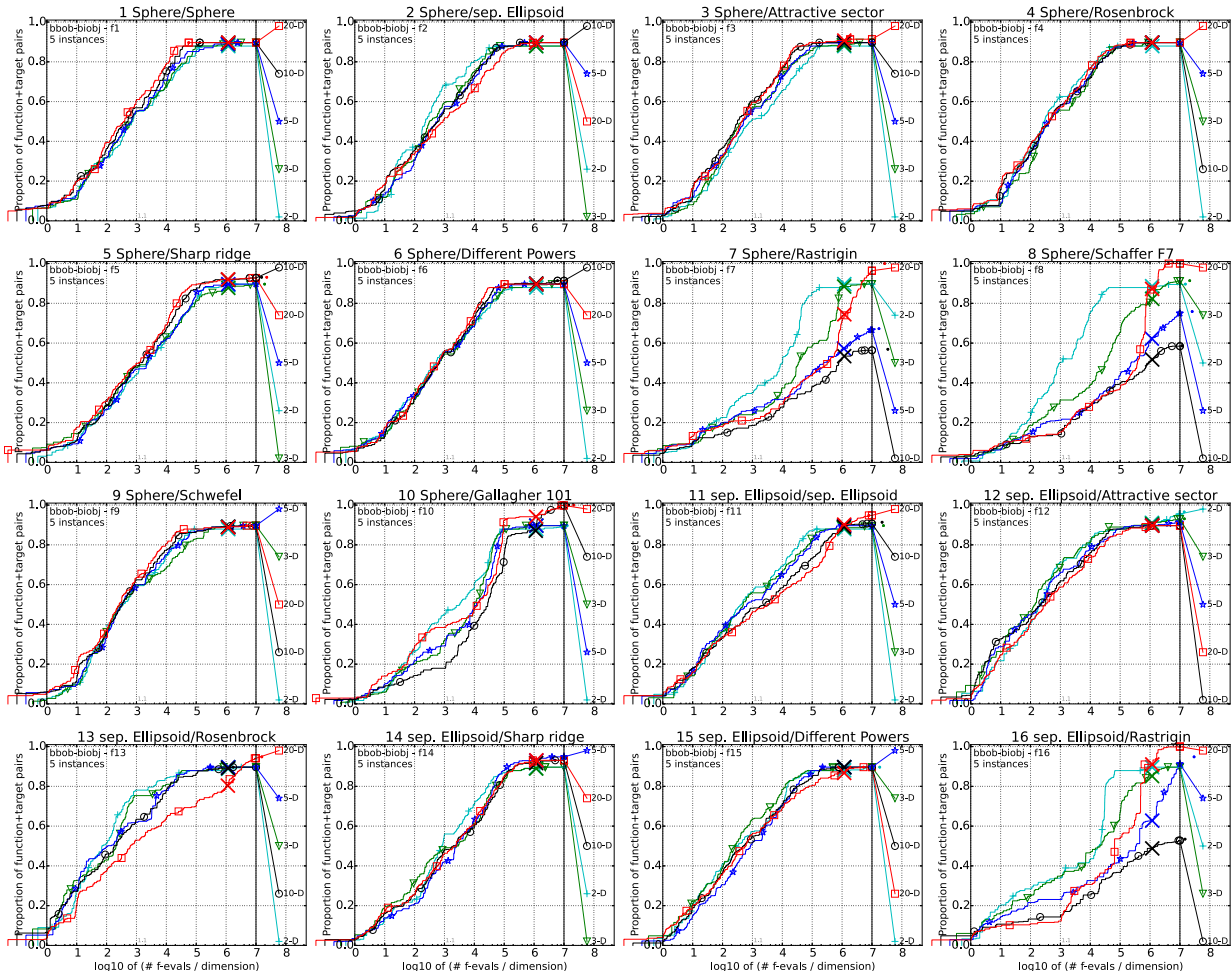
Figure 2: Empirical cumulative distribution of simulated (bootstrapped) runtimes in number of objective function evaluations divided by dimension (FEvals/DIM) for the 58 targets $\{-10^{-4}, -10^{-4.2}, -10^{-4.4}, -10^{-4.6}, -10^{-4.8}, -10^{-5}, 0, 10^{-5}, 10^{-4.9}, 10^{-4.8}, \ldots, 10^{-0.1}, 10^0\}$ for functions f_1 to f_{16} and all dimensions.

[10] T. Wagner, N. Beume, and B. Naujoks. Pareto-, aggregation-, and indicator-based methods in many-objective optimization. In *Evolutionary multi-criterion optimization*, pages 742–756. Springer, 2007.
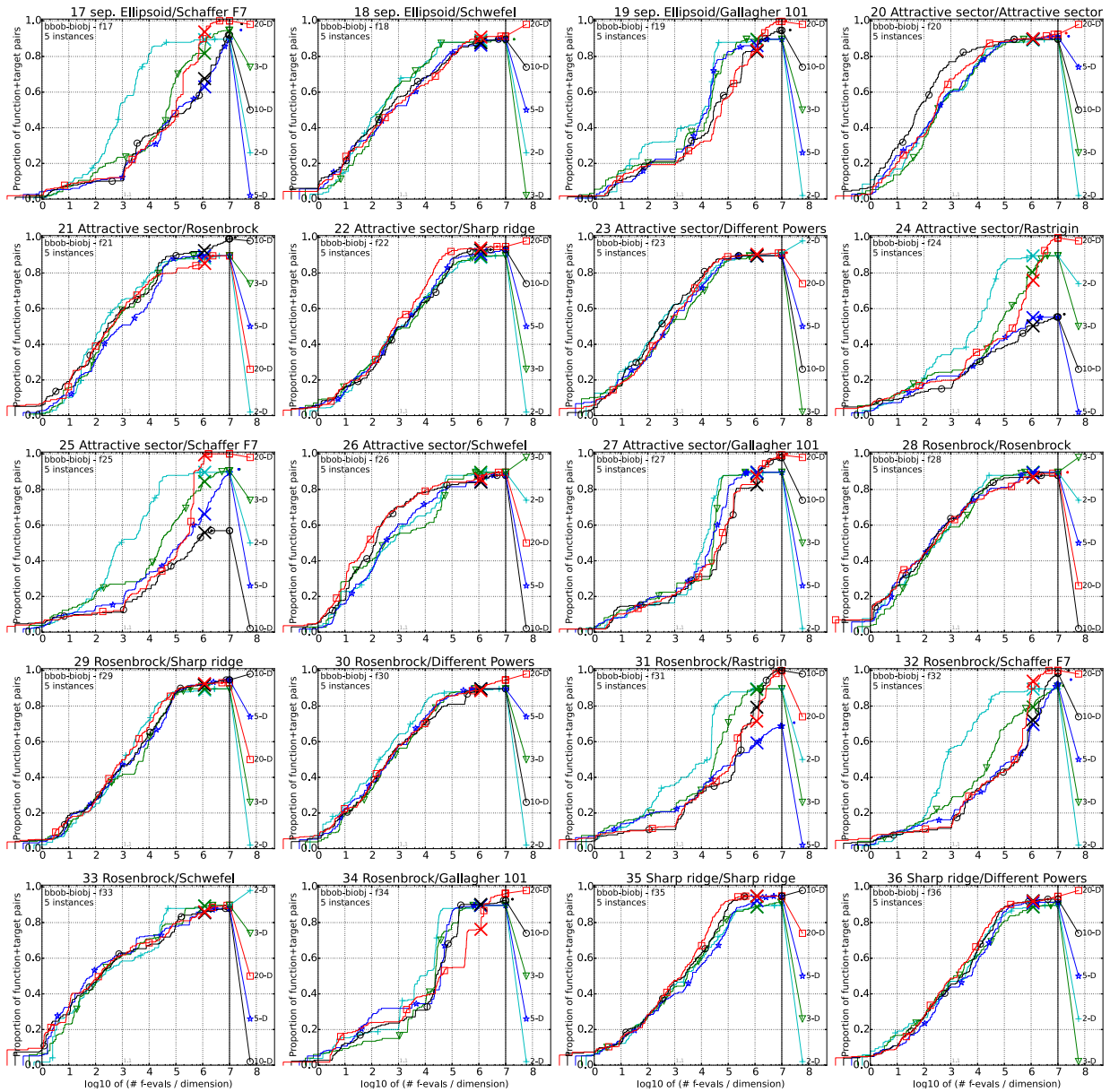
Figure 3: Empirical cumulative distribution of simulated (bootstrapped) runtimes, measured in number of objective function evaluations, divided by dimension (FEvals/DIM) for the targets as given in Fig. 2 for functions f_{17} to f_{36} and all dimensions.
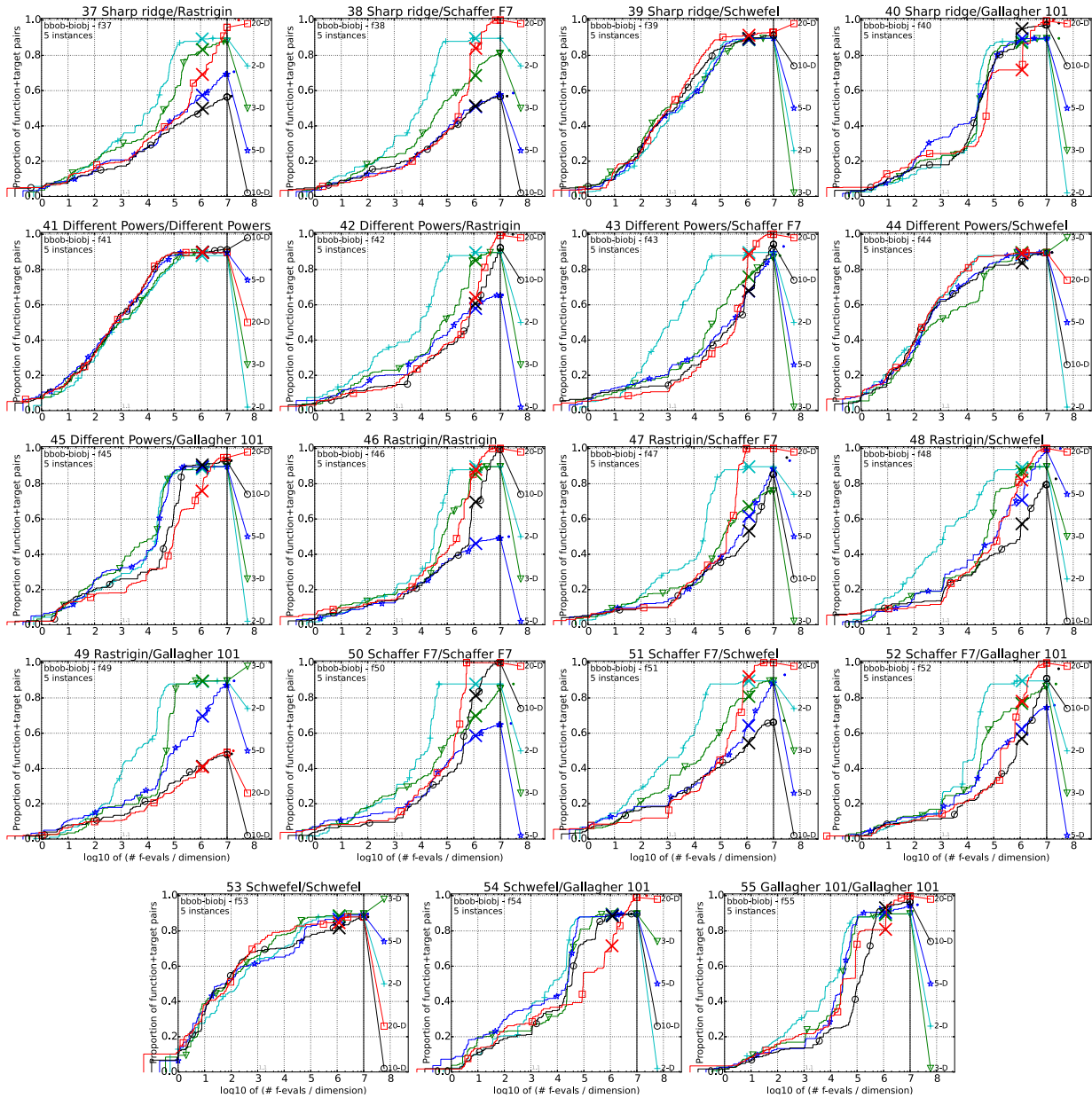
Figure 4: Empirical cumulative distribution of simulated (bootstrapped) runtimes, measured in number of objective function evaluations, divided by dimension (FEvals/DIM) for the targets as given in Fig. 2 for functions f_{37} to f_{55} and all dimensions.
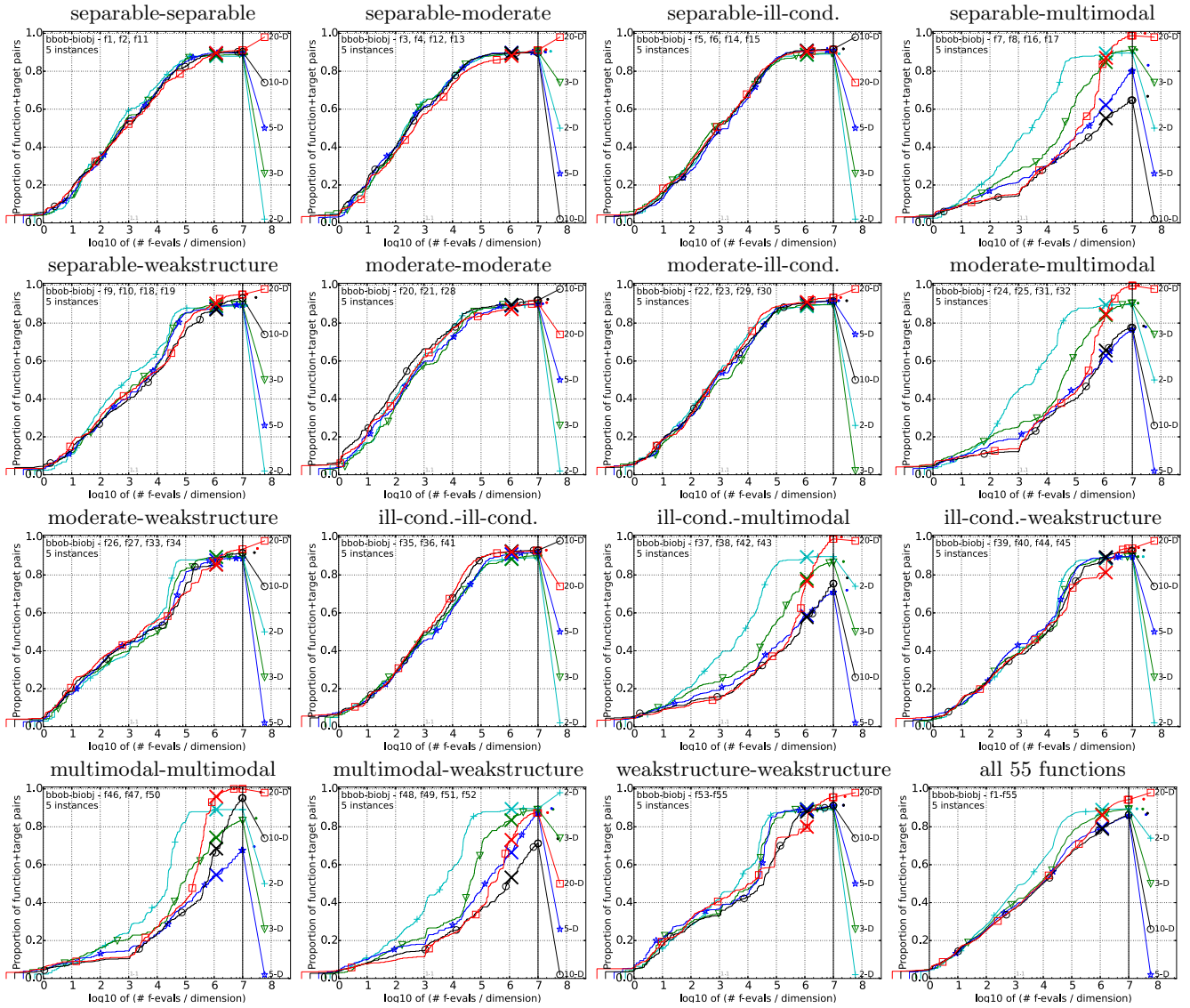
Figure 5: Empirical cumulative distribution of simulated (bootstrapped) runtimes, measured in number of objective function evaluations, divided by dimension (FEvals/DIM) for the 58 targets $\{-10^{-4}, -10^{-4.2}, -10^{-4.4}, -10^{-4.6}, -10^{-4.8}, -10^{-5}, 0, 10^{-5}, 10^{-4.9}, 10^{-4.8}, \ldots, 10^{-0.1}, 10^{0}\}$ for all function groups and all dimensions. The aggregation over all 55 functions is shown in the last plot.