The Impact of Variation Operators on the Performance of SMS-EMOA on the Bi-objective BBOB-2016 Test Suite

Anne Auger, Dimo Brockhoff, Nikolaus Hansen, Dejan Tušar, Tea Tušar, Tobias Wagner

The Impact of Variation Operators on the Performance of SMS-EMOA on the Bi-objective BBOB-2016 Test Suite

Anne Auger[⋆] Dimo Brockhoff[•] Nikolaus Hansen[⋆]
Dejan Tušar[•] Tea Tušar[•] Tobias Wagner[⋄]

[⋆]Inria Saclay—Ile-de-France
TAO team, France
LRI, Univ. Paris-Sud
firstname.lastname@inria.fr

[•]Inria Lille Nord-Europe
DOLPHIN team, France
Univ. Lille, CNRS, UMR 9189 – CRIStAL
firstname.lastname@inria.fr

[⋄]TU Dortmund University
Institute of Machining
Technology (ISF), Germany
wagner@isf.de

ABSTRACT

The S-metric-Selection Evolutionary Multi-objective Optimization Algorithm (SMS-EMOA) is one of the best-known indicator-based multi-objective optimization algorithms. It employs the S-metric or hypervolume indicator in its (steady-state) selection by deleting in each iteration the solution that has the smallest contribution to the hypervolume indicator. In the SMS-EMOA, the conceptual idea is this hypervolume-based selection. Hence the algorithm can, for example, be combined with several variation operators. Here, we benchmark two versions of SMS-EMOA which employ differential evolution (DE) and simulated binary crossover (SBX) with polynomial mutation (PM) respectively on the newly introduced bi-objective `bbob-biobj` test suite of the Comparing Continuous Optimizers (COCO) platform. The results unsurprisingly reveal that the choice of the variation operator is crucial for performance with a clear advantage of the DE variant on almost all functions.

Keywords

Benchmarking, Black-box optimization, Bi-objective optimization

1. INTRODUCTION

Indicator-based algorithms constitute an important class of (stochastic) multi-objective optimization algorithms. The S-metric-Selection Evolutionary Multi-objective Optimization Algorithm (SMS-EMOA) [4] is among the best known multi-objective algorithms and as such rather an algorithm framework with a fixed environmental selection. It uses the so-called S-metric or hypervolume contribution or hypervolume indicator loss to assign a quality to each solution in its selection procedure but does not specify the variation operators used to create new candidate solutions. Note that the hypervolume loss selection criterion is, in more general forms, also used in other well-known algorithms such as the MO-CMA-ES [12] or HypE [3]. It is therefore natural to benchmark an algorithm with hypervolume-based selection on the recently proposed bi-objective `bbob-biobj` test suite [15] of the Comparing Continuous Optimizers platform (COCO, [10]) and to investigate the impact of different variation operators on the algorithm's performance.

2. ALGORITHM PRESENTATION

The main functionality of the SMS-EMOA is summarized in the pseudo code of Algorithm 1. After a random population initialization, the algorithm performs iteration-wise the following steps until the total budget is exhausted: First, a single new candidate solution is created from the current population via a to-be-specified variation operator and evaluated on the vector-valued objective function. Then, a non-dominated ranking [8] is performed on the population augmented by the new point. Within the solutions of worst rank, the hypervolume loss of each solution to this set is computed, i.e. the amount of hypervolume difference between the set of worst rank with and without this solution. Finally, the solution with the smallest hypervolume loss is deleted[1] and the next iteration starts. The hypervolume loss computation depends on the reference point of the hypervolume indicator, which is typically chosen *relative* to the current population in the SMS-EMOA. This makes its hypervolume-based selection unique from others with fixed reference point such as the ones in the MO-CMA-ES or HypE.

2.1 Algorithm Variants

In the following, we compare two SMS-EMOA variants that differ only in the variation operators used. On the one hand, we have SMS-EMOA with polynomial mutation (PM, [7]) and simulated binary crossover (SBX, [6]): we denote this variant by SMS-PM and abbreviate it further in the tables to PM. On the other hand, we have an SMS-EMOA variant that employs differential evolution [14] and that we call SMS-DE or DE for short. In addition, we display the performance of a pure random search within $[-5, 5]^n$ [1] as a baseline.

[1]In practice, the computation of the hypervolume loss is skipped if only one solution with worst rank exists in the population.

The definitive version was published at GECCO'16, July 20–24, 2016, Denver, CO, USA, http://dx.doi.org/10.1145/2908961.2931705

Algorithm 1 Pseudocode of the SMS-EMOA framework with unspecified variation operator variate. The parameters of the algorithm are a benchmark problem `problem` including its dimension n, a population size `popsize`, the lower bounds `lb` and upper bounds `ub` for each variable, and a `budget` in number of function evaluations.

procedure SMS-EMOA(`problem`, n, `popsize`, `lb`, `ub`, `budget`)
  Initialize the first `popsize` solutions uniformly at random within $[\texttt{lb}, \texttt{ub}]^n$ as P
  while `budget` $>$ COCOGetEvaluations(`problem`) **do**
    y = variate(P);
    COCOEvaluateFunction(`problem`, x);
    $P' = P \cup \{y\}$
    Compute set $W \subseteq P'$ of worst non-domination rank via non-dominated sorting
    For all $x \in W$, determine its hypervolume loss:
      $d(x) = I_H(W) - I_H(W \setminus \{x\})$
    Choose $z \in W$ with smallest loss $d(x)$ for removal:
      $P = P' \setminus \{z\}$
  end while
end procedure

2.2 Parameters

The two benchmarked variants of the SMS-EMOA use a fixed population size of 100 and a dynamic reference point $r = (r_1, r_2)$ for the hypervolume computation defined by $r_i = \max_{p \in W} f_i(p) + 1$ ($i \in \{1, 2\}$) where W is the current set of solutions with worst non-domination rank. The variant using simulated binary crossover and polynomial mutation applies the parameters $\eta_c = 15$ and $\eta_m = 20$. The crossover probability is set to 0.9, the swap probability to 0.5, and the mutation probability per component to $1/n$, where n denotes the problem's input dimension. The DE variant applies the simple scheme, where one difference of two random solutions is added to a third one. The weight of the difference is set to $F = 0.2 + 0.6R$ with $R \in [0, 1]$ being sampled from a uniform random distribution. The crossover probability is set to $CR = 0.9$. Starting from a random index, only one block is used for the crossover. The length of the block is determined by sampling $R \in [0, 1]$ from a uniform random distribution until $R > CR$. No restarts are implemented.

3. CPU TIMING

In order to evaluate the CPU timing of the algorithm, we have run the two SMS-EMOA variants SMS-PM and SMS-DE, on the entire `bbob-biobj` test suite for $500 \cdot n$ function evaluations, i.e. for $5 \cdot n$ generations and a population size of 100. The Matlab code was run under Matlab 2008b on a Windows XP machine with Intel(R) Core(TM)2 Duo T9600 CPU 2.80GHz with 1 processor and 2 cores. The time per function evaluation over different dimensions is shown in Table 1. We observe that the dimension has almost no effect on the time per function evaluation for the algorithms tested and the relatively low budget of $500 \cdot n$ function evaluations. We also see slightly smaller times per function evaluation for the SMS-EMOA variant employing polynomial mutation.

4. RESULTS AND DISCUSSIONS

Results from experiments according to [11], [9] and [5] on the benchmark functions given in [15] are presented in Figures 1, 2, 3 and 4 and in Tables 2 and 3. The experiments were performed with COCO [10], version 1.0.1, the plots were produced with version 1.1.1.

Table 1: Results of CPU timing experiment in runtime per function evaluation (in 10^{-4} seconds) for the two SMS-EMOA variants, benchmarked here.

	time per function evaluation (in 10^{-4}s)					
algorithm	2-D	3-D	5-D	10-D	20-D	40-D
SMS-PM	5.8	5.8	5.6	5.6	5.6	5.7
SMS-DE	6.2	6.1	6.2	5.8	5.7	5.9

The **average running time (aRT)**, used in the figures and tables, depends on a given quality indicator value, $I_{\text{target}} = I^{\text{ref}} + \Delta I$, 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} [11, 13]. **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 ΔI-value achieved, measured only up to the smallest number of overall function evaluations for any unsuccessful trial under consideration.

Investigating the performance data in the shown figures and tables, results in the following general observations.

In an initial phase, both tested SMS-EMOA variants show a performance worse than the pure random search without reaching any (or just a few) hypervolume indicator targets in the first $100n$ function evaluations. This can be attributed to the fact that the pure random search is sampling within $[-5, 5]^n$ while the initial population of the SMS-EMOA variants is chosen uniformly at random within the much larger space $[-100, 100]^n$, see also [2], and thus their populations need more time to approach the Pareto set.

After this initial phase, SMS-PM is almost always better than SMS-DE in the beginning with the Attractive Sector/Different Powers function (f23) being the only exception in 10-D. The difference, however, is relatively small with function f12 (in 10-D) and the separable-moderate function group (in 5-D) showing the largest effects of a factor of around 2.

Slightly after this, SMS-DE takes over and is better in almost all functions with respect to the number of solved targets at the end of the run. The only exceptions in 10-D are the Sphere/sep. Ellipsoid function (f2) where SMS-PM stays better all the time, the sep. Ellipsoid/sep. Ellipsoid function (f11) where SMS-PM is taking over again around $4..4.5 \cdot 10^5 n$ function evaluations, and the sep. Ellipsoid/Sharp ridge (f14) on which both algos have equal performance after $10^3 n$ function evaluations. But also here, the difference between the algorithms is relatively small: only on the Sphere/Sphere (f1), the Sphere/Different Powers (f6), and the Attractive sector/Attractive sector (f20) functions are the proportions of solved target precisions in 10-D larger than 20% for some budgets.

With respect to the baseline random search, the SMS-EMOA variants are better for budgets larger than around $10^3 n$ on most functions, except for Sphere/sep. Ellipsoid (f2) and sep. Ellipsoid/sep. Ellipsoid (f11). But also for other

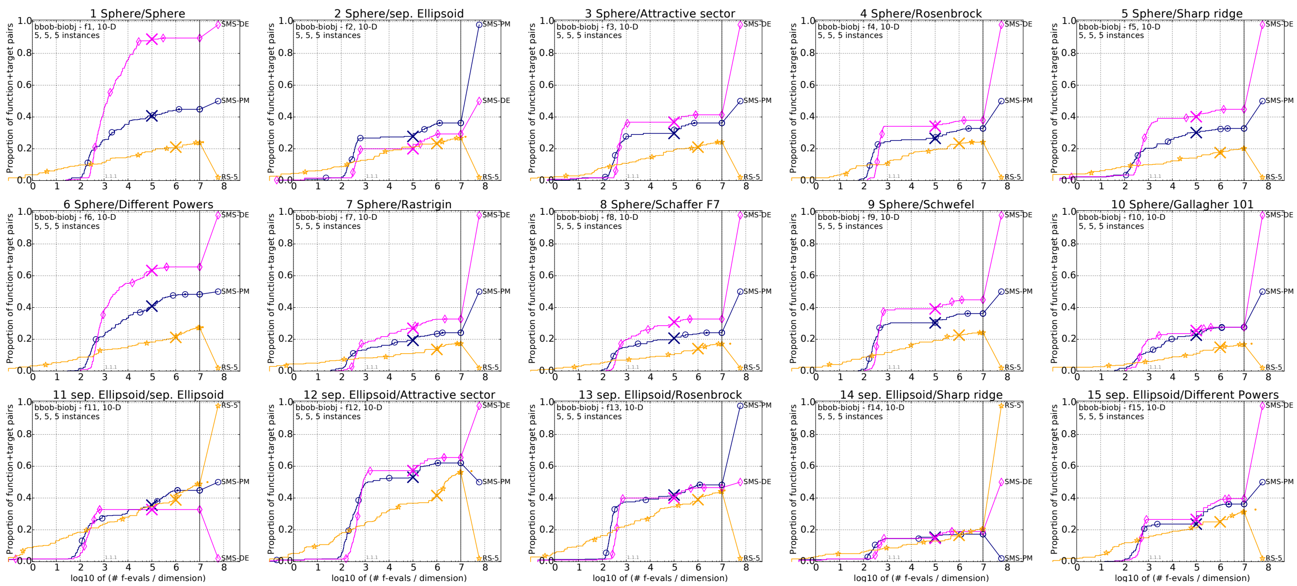

Figure 1: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for 58 targets with target precision in $\{-10^{-4}, -10^{-4.2}, -10^{-4.4}, -10^{-4.6}, -10^{-4.8}, -10^{-5}, 0, 10^{-5}, 10^{-4.9}, 10^{-4.8}, \ldots, 10^{-0.1}, 10^{0}\}$ for each single function f_1 to f_{15} in 10-D.

functions, the pure random search is expected to take over for larger budgets which is related to the fact that on many functions, the performance of the SMS-EMOA algorithms flattens out or even does not improve anymore at some point. This indicates that the algorithm has converged. Whether all solutions have arrived close to the Pareto set cannot be decided from the data, while we believe that this is not the case most of the time. Then, restarts will likely be beneficial in terms of the performance measure underlying the COCO platform, namely the hypervolume indicator of the archive of all non-dominated solutions found so far [5].

5. CONCLUSIONS

We have compared numerically two variants of the SMS-EMOA on the `bbob-biobj` test suite of the COCO platform. It turns out that the choice of the variation operators has a strong effect on the algorithm performance—with an advantage on almost all `bbob-biobj` test functions for the differential evolution variant over the polynomial mutation / simulated binary crossover version. This effect of the variation operator is expected to be present for other algorithms as well while its investigation is kept for future research.

6. ACKNOWLEDGMENTS

This work was supported by the grant ANR-12-MONU-0009 (NumBBO) of the French National Research Agency.

7. REFERENCES

[1] A. Auger, D. Brockhoff, N. Hansen, D. Tušar, T. Tušar, and T. Wagner. Benchmarking the pure random search on the bi-objective BBOB-2016 testbed. In *GECCO (Companion)*, 2016. DOI: 10.1145/2908961.2931704, to appear.

[2] A. Auger, D. Brockhoff, N. Hansen, D. Tušar, T. Tušar, and T. Wagner. The impact of search volume on the performance of RANDOMSEARCH on the bi-objective BBOB-2016 test suite. In *GECCO (Companion)*, 2016. DOI: 10.1145/2908961.2931709, to appear.

[3] J. Bader and E. Zitzler. HypE: An Algorithm for Fast Hypervolume-Based Many-Objective Optimization. *Evolutionary Computation*, 19(1):45Ö■-76, 2011.

[4] N. Beume, B. Naujoks, and M. Emmerich. SMS-EMOA: Multiobjective Selection Based on Dominated Hypervolume. *European Journal of Operational Research*, 181(3):1653–1669, 2007.

[5] D. Brockhoff, T. Tušar, D. Tušar, T. Wagner, N. Hansen, and A. Auger. Biobjective performance assessment with the COCO platform. *ArXiv e-prints*, arXiv:1605.01746, 2016.

[6] K. Deb and R. B. Agrawal. Simulated binary crossover for continuous search space. *Complex systems*, 9(2):115–148, 1995.

[7] K. Deb and M. Goyal. A combined genetic adaptive search (GeneAS) for engineering design. *Computer Science and Informatics*, 26:30–45, 1996.

[8] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, Massachusetts, 1989.

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

[10] N. Hansen, A. Auger, O. Mersmann, T. Tušar, and D. Brockhoff. COCO: A platform for comparing continuous optimizers in a black-box setting. *ArXiv e-prints*, arXiv:1603.08785, 2016.

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

[12] C. Igel, N. Hansen, and S. Roth. Covariance Matrix

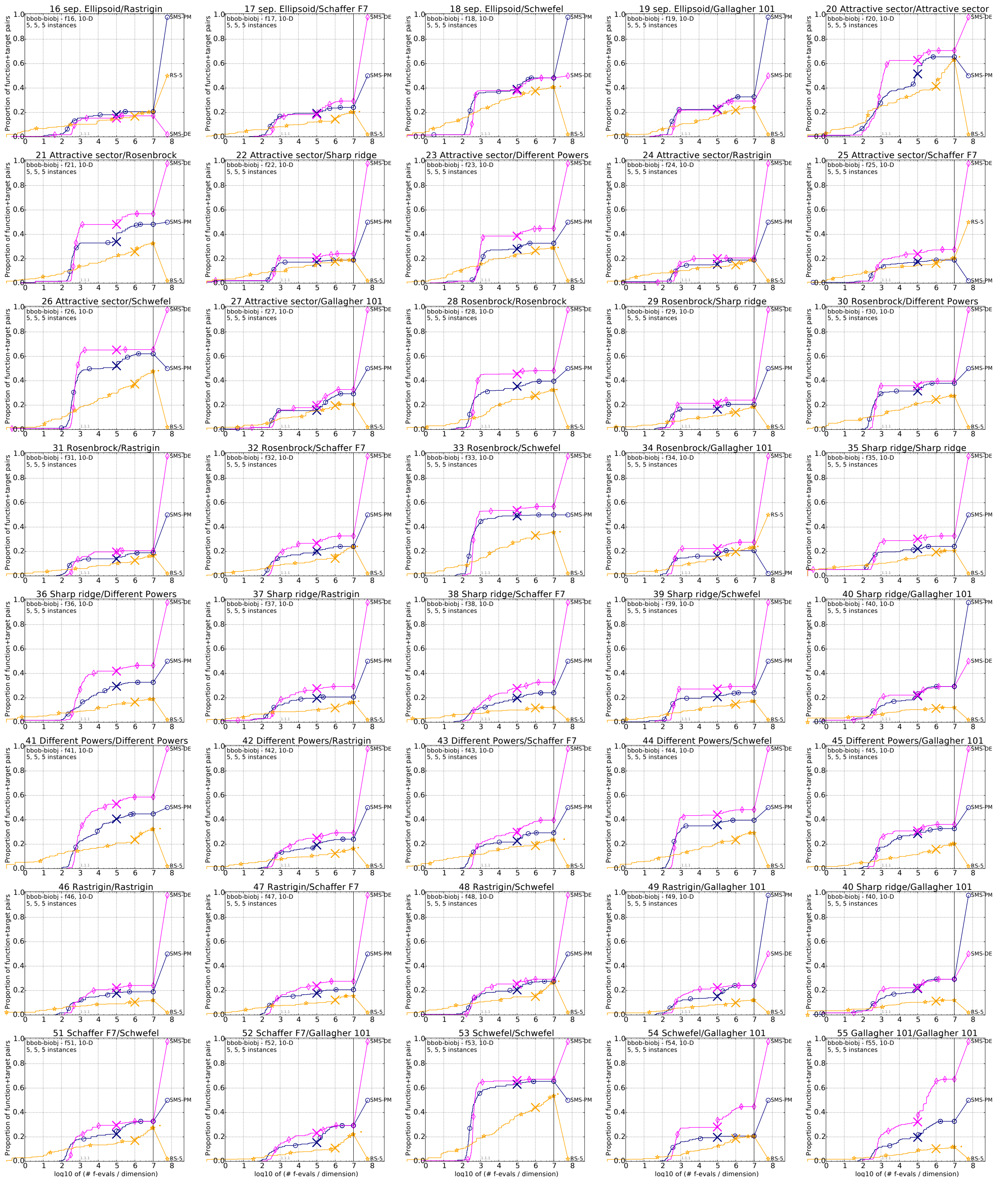

Figure 2: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) as in Fig. 1 but for functions f_{16} to f_{55} in 10-D.

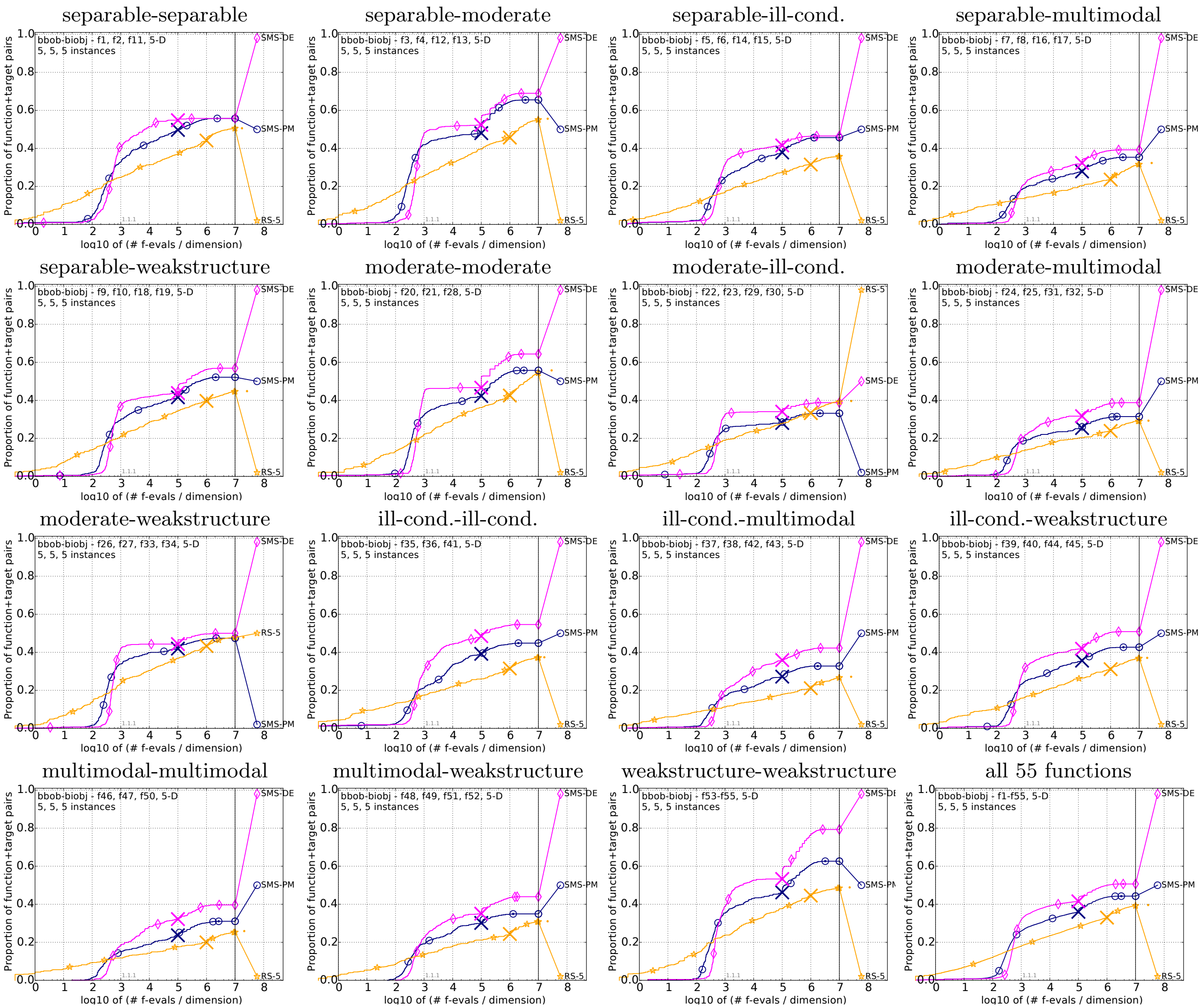

Figure 3: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for 58 **targets with target precision in** $\{-10^{-4}, -10^{-4.2}, -10^{-4.4}, -10^{-4.6}, -10^{-4.8}, -10^{-5}, 0, 10^{-5}, 10^{-4.9}, 10^{-4.8}, \ldots, 10^{-0.1}, 10^{0}\}$ **for all functions and subgroups in 5-D.**

Adaptation for Multi-objective Optimization. *Evolutionary Computation*, 15(1):1–28, 2007.

[13] K. Price. Differential evolution vs. the functions of the second ICEO. In *Proceedings of the IEEE International Congress on Evolutionary Computation*, pages 153–157, 1997.

[14] R. Storn and K. Price. Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[15] T. Tušar, D. Brockhoff, N. Hansen, and A. Auger. COCO: The bi-objective black-box optimization benchmarking (bbob-biobj) test suite. *ArXiv e-prints*, arXiv:1604.00359, 2016.

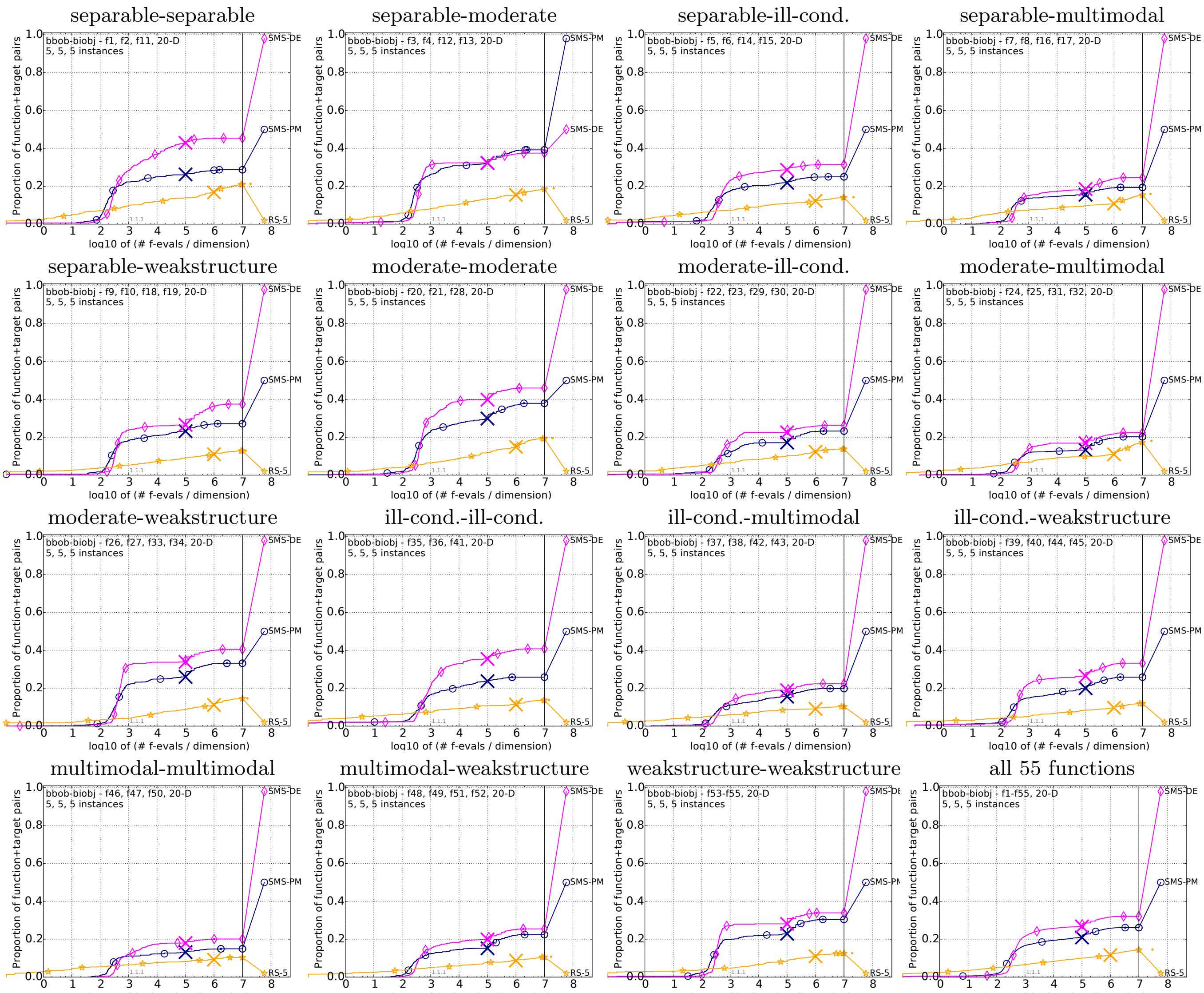

Figure 4: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for 58 **targets with target precision in** $\{-10^{-4}, -10^{-4.2}, -10^{-4.4}, -10^{-4.6}, -10^{-4.8}, -10^{-5}, 0, 10^{-5}, 10^{-4.9}, 10^{-4.8}, \ldots, 10^{-0.1}, 10^{0}\}$ **for all functions and subgroups in 20-D.**

Δf	1e0	1e-2	1e-5	#succ
f1				
PM	197(224)	8480(2457)	∞ *5e5*	0/5
DE	373(367)	**3173**(229)	**9.3e4**(3176)	5/5
RS-5	**1**(0)	3.0e6(1e6)	∞ *5e6*	0/5
f2				
PM	182(78)	**3.1e4**(4e4)	∞ *5e5*	0/5
DE	190(305)	1.3e5(1e5)	∞ *5e5*	0/5
RS-5	**3.0**(2)	3.8e6(4e6)	∞ *5e6*	0/5
f3				
PM	64(133)	7.8e5(2e6)	∞ *5e5*	0/5
DE	234(286)	**1.3e5**(3e5)	∞ *5e5*	0/5
RS-5	**1**(0)	5.1e6(5e6)	∞ *5e6*	0/5
f4				
PM	398(154)	8.0e5(5e5)	∞ *5e5*	0/5
DE	765(268)	**3480**(284)	∞ *5e5*	0/5
RS-5	**1**(0)	1.4e6(5e5)	∞ *5e6*	0/5
f5				
PM	5.6(2)	**5.0e4**(3e4)	∞ *5e5*	0/5
DE	5.2(4)	1.3e5(5e5)	∞ *5e5*	0/5
RS-5	**1**(0)	∞	∞ *5e6*	0/5
f6				
PM	304(254)	1.5e4(2e4)	∞ *5e5*	0/5
DE	581(421)	**3846**(439)	∞ *5e5*	0/5
RS-5	**1**(0)	4.2e6(4e6)	∞ *5e6*	0/5
f7				
PM	200(244)	2.4e6(4e6)	∞ *5e5*	0/5
DE	288(137)	**2.0e5**(4e5)	∞ *5e5*	0/5
RS-5	**1**(0)	∞	∞ *5e6*	0/5
f8				
PM	562(186)	2.3e6(2e6)	∞ *5e5*	0/5
DE	1159(119)	**1.7e5**(2e5)	∞ *5e5*	0/5
RS-5	**3.8**(0)	∞	∞ *5e6*	0/5
f9				
PM	357(177)	6.3e4(1e5)	∞ *5e5*	0/5
DE	705(550)	**3125**(314)	∞ *5e5*	0/5
RS-5	**2.4**(4)	2.4e6(4e6)	∞ *5e6*	0/5
f10				
PM	339(107)	1.8e5(3e5)	∞ *5e5*	0/5
DE	660(127)	**1.5e5**(2e5)	∞ *5e5*	0/5
RS-5	**1**(0)	2.3e7(2e7)	∞ *5e6*	0/5
f11				
PM	1.2(0.5)	2.4e4(6e4)	∞ *5e5*	0/5
DE	1.2(0)	**2225**(450)	∞ *5e5*	0/5
RS-5	**1**(0)	4.5e5(1e6)	∞ *5e6*	0/5
f12				
PM	65(82)	**1.3e5**(1e5)	∞ *5e5*	0/5
DE	145(260)	1.3e5(5e5)	**2.0e6**(2e6)	1/5
RS-5	**1**(0)	2.1e6(2e6)	∞ *5e6*	0/5
f13				
PM	220(272)	**1583**(390)	∞ *5e5*	0/5
DE	573(582)	2954(386)	∞ *5e5*	0/5
RS-5	**2.8**(4)	1.3e5(2e5)	∞ *5e6*	0/5
f14				
PM	2.0(0)	∞	∞ *5e5*	0/5
DE	2.0(0)	∞	∞ *5e5*	0/5
RS-5	**1**(0)	∞	∞ *5e6*	0/5
f15				
PM	280(220)	7.6e5(4e5)	∞ *5e5*	0/5
DE	634(918)	**7.6e5**(2e6)	∞ *5e5*	0/5
RS-5	**6.4**(6)	8.4e6(9e6)	∞ *5e6*	0/5
f16				
PM	65(127)	∞	∞ *5e5*	0/5
DE	42(72)	∞	∞ *5e5*	0/5
RS-5	**2.0**(2)	**2.2e7**(2e7)	∞ *5e6*	0/5
f17				
PM	**195**(371)	2.2e6(3e6)	∞ *5e5*	0/5
DE	584(718)	**2.0e6**(2e6)	∞ *5e5*	0/5
RS-5	1702(2)	∞	∞ *5e6*	0/5
f18				
PM	4.8(4)	1.1e4(2e4)	∞ *5e5*	0/5
DE	3.2(4)	**2688**(196)	∞ *5e5*	0/5
RS-5	**1**(0)	4.2e5(5e5)	∞ *5e6*	0/5
f19				
PM	147(115)	2.0e6(2e6)	∞ *5e5*	0/5
DE	383(359)	2.0e6(3e6)	∞ *5e5*	0/5
RS-5	**1.8**(1)	**5.5e5**(2e5)	∞ *5e6*	0/5
f20				
PM	166(0.2)	3.6e5(3e5)	∞ *5e5*	0/5
DE	348(866)	**1.3e5**(5e5)	∞ *5e5*	0/5
RS-5	**1**(0)	1.6e6(5e6)	∞ *5e6*	0/5
f21				
PM	114(128)	**1.6e5**(3e5)	∞ *5e5*	0/5
DE	295(344)	3.4e5(6e5)	∞ *5e5*	0/5
RS-5	**1**(0)	5.7e6(3e6)	∞ *5e6*	0/5
f22				
PM	1.6(0.5)	∞	∞ *5e5*	0/5
DE	1.6(0.5)	**2.0e6**(4e6)	∞ *5e5*	0/5
RS-5	**1**(0)	∞	∞ *5e6*	0/5
f23				
PM	15(18)	7.6e5(6e5)	∞ *5e5*	0/5
DE	16(10)	**1.3e5**(3e5)	∞ *5e5*	0/5
RS-5	**1**(0)	3.8e6(4e6)	∞ *5e6*	0/5
f24				
PM	100(18)	∞	∞ *5e5*	0/5
DE	239(556)	**2.1e6**(2e6)	∞ *5e5*	0/5
RS-5	**1.4**(1)	∞	∞ *5e6*	0/5
f25				
PM	151(193)	2.2e6(3e6)	∞ *5e5*	0/5
DE	311(382)	**4.1e5**(6e5)	∞ *5e5*	0/5
RS-5	**3.0**(5)	∞	∞ *5e6*	0/5
f26				
PM	153(232)	1.4e5(5e5)	∞ *5e5*	0/5
DE	560(697)	**3365**(770)	∞ *5e5*	0/5
RS-5	**3.6**(6)	1.7e6(3e6)	∞ *5e6*	0/5
f27				
PM	30(40)	2.0e6(2e6)	∞ *5e5*	0/5
DE	30(38)	**3.4e5**(6e5)	∞ *5e5*	0/5
RS-5	**2.0**(2)	1.7e6(2e6)	∞ *5e6*	0/5
f28				
PM	275(59)	2.9e4(3e4)	∞ *5e5*	0/5
DE	531(198)	**3118**(464)	∞ *5e5*	0/5
RS-5	**1**(0)	7.4e5(2e6)	∞ *5e6*	0/5
f29				
PM	233(170)	∞	∞ *5e5*	0/5
DE	402(433)	∞	∞ *5e5*	0/5
RS-5	**1**(0)	∞	∞ *5e6*	0/5
f30				
PM	569(176)	7.9e5(9e5)	∞ *5e5*	0/5
DE	1014(264)	**1.3e5**(6e5)	∞ *5e5*	0/5
RS-5	**2.0**(2)	3.8e6(4e6)	∞ *5e6*	0/5
f31				
PM	450(150)	∞	∞ *5e5*	0/5
DE	767(588)	**8.0e5**(1e6)	∞ *5e5*	0/5
RS-5	**1**(0)	∞	∞ *5e6*	0/5
f32				
PM	587(101)	∞	∞ *5e5*	0/5
DE	1130(270)	**8.2e5**(8e5)	∞ *5e5*	0/5
RS-5	**3.8**(2)	∞	∞ *5e6*	0/5
f33				
PM	388(138)	**1949**(856)	∞ *5e5*	0/5
DE	1010(741)	2461(78)	∞ *5e5*	0/5
RS-5	**4.6**(0)	2.4e4(2e4)	∞ *5e6*	0/5
f34				
PM	393(84)	∞	∞ *5e5*	0/5
DE	880(138)	**7.5e5**(8e5)	∞ *5e5*	0/5
RS-5	**1.6**(0.5)	1.6e6(3e6)	∞ *5e6*	0/5
f35				
PM	**1**(0)	∞	∞ *5e5*	0/5
DE	**1**(0)	∞	∞ *5e5*	0/5
RS-5	**1**(0)	**2.3e7**(2e7)	∞ *5e6*	0/5
f36				
PM	83(166)	4.7e5(5e5)	∞ *5e5*	0/5
DE	153(185)	**2.6e5**(5e5)	∞ *5e5*	0/5
RS-5	**1**(0)	∞	∞ *5e6*	0/5
f37				
PM	7.4(10)	∞	∞ *5e5*	0/5
DE	7.4(2)	**9.3e5**(1e6)	∞ *5e5*	0/5
RS-5	**1**(0)	∞	∞ *5e6*	0/5
f38				
PM	239(185)	∞	∞ *5e5*	0/5
DE	602(346)	**5.5e5**(5e5)	∞ *5e5*	0/5
RS-5	**1**(0)	∞	∞ *5e6*	0/5
f39				
PM	8.4(2)	∞	∞ *5e5*	0/5
DE	8.4(2)	**2.0e6**(2e6)	∞ *5e5*	0/5
RS-5	**1**(0)	∞	∞ *5e6*	0/5
f40				
PM	51(95)	1.1e6(1e6)	∞ *5e5*	0/5
DE	106(18)	**4.7e4**(5289)	∞ *5e5*	0/5
RS-5	**1**(0)	∞	∞ *5e6*	0/5
f41				
PM	478(168)	2.2e4(7055)	∞ *5e5*	0/5
DE	1041(578)	**5201**(1499)	∞ *5e5*	0/5
RS-5	**3.8**(4)	3.6e6(5e6)	∞ *5e6*	0/5
f42				
PM	533(168)	∞	∞ *5e5*	0/5
DE	1119(800)	**9.9e5**(6e5)	∞ *5e5*	0/5
RS-5	**3.0**(2)	∞	∞ *5e6*	0/5
f43				
PM	628(186)	8.7e5(8e5)	∞ *5e5*	0/5
DE	1310(227)	**1.3e5**(1e5)	∞ *5e5*	0/5
RS-5	**2.2**(2)	∞	∞ *5e6*	0/5
f44				
PM	521(347)	8.9e4(2e5)	∞ *5e5*	0/5
DE	935(497)	**3365**(876)	∞ *5e5*	0/5
RS-5	**1**(0)	2.9e6(3e6)	∞ *5e6*	0/5
f45				
PM	507(329)	**6.8e4**(6e4)	∞ *5e5*	0/5
DE	871(597)	1.4e5(1e5)	∞ *5e5*	0/5
RS-5	**3.8**(6)	2.4e7(2e7)	∞ *5e6*	0/5
f46				
PM	362(182)	∞	∞ *5e5*	0/5
DE	716(588)	∞	∞ *5e5*	0/5
RS-5	**1**(0)	∞	∞ *5e6*	0/5
f47				
PM	460(98)	∞	∞ *5e5*	0/5
DE	908(246)	**2.1e6**(2e6)	∞ *5e5*	0/5
RS-5	**1**(0)	∞	∞ *5e6*	0/5
f48				
PM	703(292)	**1.0e6**(1e6)	∞ *5e5*	0/5
DE	1232(290)	2.0e6(3e6)	∞ *5e5*	0/5
RS-5	**5.6**(4)	∞	∞ *5e6*	0/5
f49				
PM	442(181)	∞	∞ *5e5*	0/5
DE	787(229)	**8.0e5**(1e6)	∞ *5e5*	0/5
RS-5	**2.0**(2)	∞	∞ *5e6*	0/5
f50				
PM	541(122)	2.5e6(4e6)	∞ *5e5*	0/5
DE	1084(332)	**1.5e5**(1e5)	∞ *5e5*	0/5
RS-5	**1**(0)	∞	∞ *5e6*	0/5
f51				
PM	558(130)	8.2e5(1e6)	∞ *5e5*	0/5
DE	1032(428)	**3.5e4**(3e4)	∞ *5e5*	0/5
RS-5	**1.2**(0.5)	∞	∞ *5e6*	0/5
f52				
PM	507(270)	∞	∞ *5e5*	0/5
DE	999(138)	**2.0e6**(2e6)	∞ *5e5*	0/5
RS-5	**1.4**(0.5)	2.1e7(3e7)	∞ *5e6*	0/5
f53				
PM	321(294)	**1613**(460)	∞ *5e5*	0/5
DE	650(552)	2302(194)	**2.0e6**(2e6)	1/5
RS-5	**20**(24)	1.8e4(2e4)	∞ *5e6*	0/5
f54				
PM	348(180)	8.5e5(8e5)	∞ *5e5*	0/5
DE	1077(124)	**3.4e5**(6e5)	∞ *5e5*	0/5
RS-5	**1**(0)	9.9e5(5e5)	∞ *5e6*	0/5
f55				
PM	447(183)	2.0e6(2e6)	∞ *5e5*	0/5
DE	638(272)	**7.8e5**(1e6)	**2.1e6**(3e6)	1/5
RS-5	**1**(0)	2.2e7(2e7)	∞ *5e6*	0/5

Table 2: Average runtime (aRT) to reach given targets, measured in number of function evaluations, in dimension 5. 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 ΔI-values as shown in the top row. #succ is the number of trials that reached the last target $I^{\text{ref}} + 10^{-5}$. 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 of 110. Best results are printed in bold.

Δf	1e0	1e-2	1e-5	#succ
f1				
PM	525(86)	∞	∞ *2e6*	0/5
DE	1201(244)	**1.4e4**(1787)	**9.6e6**(9e6)	1/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
f2				
PM	609(308)	∞	∞ *2e6*	0/5
DE	1127(319)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
f3				
PM	155(75)	∞	∞ *2e6*	0/5
DE	284(488)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
f4				
PM	1567(406)	∞	∞ *2e6*	0/5
DE	3144(406)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
f5				
PM	1.4(0.5)	∞	∞ *2e6*	0/5
DE	1.4(0.5)	**1.9e5**(1e5)	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
f6				
PM	1216(374)	∞	∞ *2e6*	0/5
DE	2694(930)	**2.4e4**(6796)	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
f7				
PM	548(129)	∞	∞ *2e6*	0/5
DE	1448(858)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
f8				
PM	1852(264)	∞	∞ *2e6*	0/5
DE	3949(1128)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
f9				
PM	1280(161)	∞	∞ *2e6*	0/5
DE	2719(146)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
f10				
PM	782(94)	∞	∞ *2e6*	0/5
DE	1899(286)	**1.4e6**(3e6)	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
f11				
PM	**1**(0)	∞	∞ *2e6*	0/5
DE	**1**(0)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	**8.6e7**(6e7)	∞ *2e7*	0/5
f12				
PM	4.0(6)	8.8e4(2e5)	∞ *2e6*	0/5
DE	4.0(3)	**1.3e4**(5150)	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
f13				
PM	46(65)	**1.8e6**(2e6)	∞ *2e6*	0/5
DE	47(66)	8.0e6(1e7)	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
f14				
PM	6.2(5)	∞	∞ *2e6*	0/5
DE	6.2(6)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
f15				
PM	921(426)	∞	∞ *2e6*	0/5
DE	1627(814)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
f16				
PM	346(133)	∞	∞ *2e6*	0/5
DE	516(152)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
f17				
PM	1083(354)	∞	∞ *2e6*	0/5
DE	1816(978)	∞	∞ *2e6*	0/5
RS-5	**1.8**(1)	∞	∞ *2e7*	0/5
f18				
PM	**1**(0)	∞	∞ *2e6*	0/5
DE	**1**(0)	**8.0e6**(1e7)	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
f19				
PM	768(116)	∞	∞ *2e6*	0/5
DE	1427(652)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
f20				
PM	77(167)	8.0e6(4e6)	∞ *2e6*	0/5
DE	110(249)	**5.6e5**(2e4)	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
f21				
PM	538(351)	∞	∞ *2e6*	0/5
DE	1227(704)	**5.1e5**(2e6)	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
f22				
PM	1.6(1)	∞	∞ *2e6*	0/5
DE	1.6(1)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
f23				
PM	359(152)	∞	∞ *2e6*	0/5
DE	708(498)	∞	∞ *2e6*	0/5
RS-5	**1.2**(0.5)	∞	∞ *2e7*	0/5
f24				
PM	107(9)	∞	∞ *2e6*	0/5
DE	180(216)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
f25				
PM	801(216)	∞	∞ *2e6*	0/5
DE	1579(934)	∞	∞ *2e6*	0/5
RS-5	**1.2**(0.5)	∞	∞ *2e7*	0/5
f26				
PM	1030(951)	1.4e6(2e6)	∞ *2e6*	0/5
DE	2228(2683)	**1.2e4**(2015)	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
f27				
PM	372(230)	∞	∞ *2e6*	0/5
DE	699(258)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
f28				
PM	937(292)	∞	∞ *2e6*	0/5
DE	1844(675)	**5.1e5**(1e6)	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
f29				
PM	972(95)	∞	∞ *2e6*	0/5
DE	1923(286)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
f30				
PM	2791(586)	**8.1e6**(1e7)	∞ *2e6*	0/5
DE	4765(702)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
f31				
PM	1724(252)	∞	∞ *2e6*	0/5
DE	3566(524)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
f32				
PM	2759(775)	∞	∞ *2e6*	0/5
DE	5052(295)	∞	∞ *2e6*	0/5
RS-5	**613**(2)	∞	∞ *2e7*	0/5
f33				
PM	885(468)	3.8e4(6e4)	∞ *2e6*	0/5
DE	1972(585)	**8622**(510)	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
f34				
PM	1855(334)	∞	∞ *2e6*	0/5
DE	3213(190)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
f35				
PM	**1**(0)	∞	∞ *2e6*	0/5
DE	**1**(0)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
f36				
PM	113(157)	∞	∞ *2e6*	0/5
DE	177(172)	**8.0e6**(7e6)	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
f37				
PM	133(88)	∞	∞ *2e6*	0/5
DE	186(160)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
f38				
PM	1202(414)	∞	∞ *2e6*	0/5
DE	2402(59)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
f39				
PM	62(70)	∞	∞ *2e6*	0/5
DE	84(187)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
f40				
PM	9.0(18)	∞	∞ *2e6*	0/5
DE	9.0(18)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
f41				
PM	4012(1168)	∞	∞ *2e6*	0/5
DE	6485(1111)	**6.0e4**(5939)	∞ *2e6*	0/5
RS-5	**1.2**(0.5)	∞	∞ *2e7*	0/5
f42				
PM	2862(1077)	∞	∞ *2e6*	0/5
DE	4875(988)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
f43				
PM	3094(88)	∞	∞ *2e6*	0/5
DE	5232(290)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
f44				
PM	2313(541)	∞	∞ *2e6*	0/5
DE	4603(1290)	**8.0e6**(6e6)	∞ *2e6*	0/5
RS-5	**2.0**(2)	∞	∞ *2e7*	0/5
f45				
PM	1338(301)	∞	∞ *2e6*	0/5
DE	3072(370)	**8.1e6**(9e6)	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
f46				
PM	1048(225)	∞	∞ *2e6*	0/5
DE	2365(622)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
f47				
PM	2327(496)	∞	∞ *2e6*	0/5
DE	4671(459)	∞	∞ *2e6*	0/5
RS-5	**1.2**(0.5)	∞	∞ *2e7*	0/5
f48				
PM	866(312)	∞	∞ *2e6*	0/5
DE	1994(868)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
f49				
PM	1085(272)	∞	∞ *2e6*	0/5
DE	2230(307)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
f50				
PM	2732(606)	∞	∞ *2e6*	0/5
DE	7753(6868)	∞	∞ *2e6*	0/5
RS-5	**1.4**(1)	∞	∞ *2e7*	0/5
f51				
PM	2149(378)	∞	∞ *2e6*	0/5
DE	3970(295)	∞	∞ *2e6*	0/5
RS-5	**6.0**(2)	∞	∞ *2e7*	0/5
f52				
PM	2094(69)	∞	∞ *2e6*	0/5
DE	3596(364)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
f53				
PM	394(445)	4.7e5(6e5)	∞ *2e6*	0/5
DE	762(1201)	**6821**(89)	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
f54				
PM	1615(375)	∞	∞ *2e6*	0/5
DE	3204(339)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
f55				
PM	771(116)	∞	∞ *2e6*	0/5
DE	1905(232)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5

Table 3: Average runtime (aRT) to reach given targets, measured in number of function evaluations, in dimension 20. 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 ΔI-values as shown in the top row. #succ is the number of trials that reached the last 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 of 110. Best results are printed in bold.