HAL
open science

The Impact of Search Volume on the Performance of RANDOMSEARCH on the Bi-objective BBOB-2016 Test Suite

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

To cite this version:

Anne Auger, Dimo Brockhoff, Nikolaus Hansen, Dejan Tušar, Tea Tušar, et al.. The Impact of Search Volume on the Performance of RANDOMSEARCH on the Bi-objective BBOB-2016 Test Suite. GECCO 2016 - Genetic and Evolutionary Computation Conference, Jul 2016, Denver, CO, United States. pp.1257 - 1264, 10.1145/2908961.2931709 . hal-01435453

HAL Id: hal-01435453

https://inria.hal.science/hal-01435453

Submitted on 14 Jan 2017

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

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

The Impact of Search Volume on the Performance of RANDOMSEARCH on the Bi-objective BBOB-2016 Test Suite

Anne Auger*
Dejan Tušar•
*Inria Saclay—Ile-de-France
TAO team, France
LRI, Univ. Paris-Sud
firstname.lastname@inria.fr

Dimo Brockhoff•
Tea Tušar•
•Inria Lille Nord-Europe
DOLPHIN team, France
Univ. Lille, CNRS, UMR 9189 – CRIStAL
firstname.lastname@inria.fr

Nikolaus Hansen*
Tobias Wagner◇
◇TU Dortmund University
Institute of Machining
Technology (ISF), Germany
wagner@isf.de

ABSTRACT

Pure random search is undeniably the simplest stochastic search algorithm for numerical optimization. Essentially the only thing to be determined to implement the algorithm is its sampling space, the influence of which on the performance on the bi-objective `bbob-biobj` test suite of the COCO platform is investigated here. It turns out that the suggested region of interest of $[-100, 100]^n$ (with n being the problem dimension) has a too vast volume for the algorithm to approximate the Pareto set effectively. Better performance can be achieved if solutions are sampled uniformly within $[-5, 5]^n$ or $[-4, 4]^n$. The latter sampling box corresponds to the smallest hypercube encapsulating all single-objective optima of the 55 constructed bi-objective problems of the `bbob-biobj` test suite. However, not all best known Pareto set approximations are entirely contained within $[-5, 5]^n$.

Keywords

Benchmarking, Black-box optimization, Bi-objective optimization

1. INTRODUCTION

Pure random search [2] often serves as a baseline algorithm in benchmarking numerical optimizers. However, the choice of the algorithm's sampling space is crucial and the impact of this choice will be illustrated here on the bi-objective `bbob-biobj` test suite [7] of the Comparing Continuous Optimizers platform COCO [4]. Although `bbob-biobj` is a test suite of unconstrained problems, the COCO platform allows algorithms to access some largest and smallest value of interest on the variables for each problem. They can, in turn, serve as a first estimate about where to sample initial candidate solutions. For the single-objective `bbob` test suite, which is used as basis for the construction of the problems in the `bbob-biobj` suite, the upper and lower variable bounds define a so-called region of interest of $[-5, 5]^n$ with n being the problem dimension. This region contains by construction, the optima of all problems that are located in the hypercube $[-4, 4]^n$. Combining single-objective problems from the `bbob` suite to the bi-objective problems of `bbob-biobj`, however, does not guarantee that the entire Pareto sets are contained within the single-objective region of interest. Preliminary investigations therefore suggested to enlarge the region of interest for `bbob-biobj` to $[-100, 100]^n$.

As we will see below, the significantly larger search volume of $[-100, 100]^n$ makes the pure random search, when sampling in it, clearly less effective than if it samples within $[-4, 4]^n$ or $[-5, 5]^n$—even if parts of the Pareto set for some problems cannot be reached.[1]

2. ALGORITHM PRESENTATION

Until the given budget (in terms of a given number of function evaluations) is exhausted, the pure random search samples a new candidate solution independently identically distributed uniformly at random within a sampling box $[-b, b]^n$. Here, we investigate three variants with $b = 4$, $b = 5$, and $b = 100$ and denote them as RS-4, RS-5, and RS-100 respectively. For the final experiments, a budget of $10^6 \cdot n$ function evaluations has been chosen for RS-5, which will serve as the baseline. The other two algorithms have been run for $10^5 \cdot n$ function evaluations.

3. CPU TIMING

In order to evaluate the CPU timing of the pure random search, we have run RS-5, sampling within $[-5, 5]^n$, on the entire `bbob-biobj` test suite for $10^3 \cdot n$ function evaluations. The Matlab/Octave code was run with Octave 4.0.0 on a Windows 7 machine with Intel(R) Core(TM) i7-5600U CPU 2.60GHz with 1 processor and 2 cores. The time per function evaluation for dimensions 2, 3, 5, 10, 20, and 40 equaled $6.0 \cdot 10^{-5}$, $5.8 \cdot 10^{-5}$, $5.5 \cdot 10^{-5}$, $5.5 \cdot 10^{-5}$, $5.8 \cdot 10^{-5}$, and $6.8 \cdot 10^{-5}$ seconds respectively. The other algorithms being essentially

©The authors, 2016. This is the authors' version of the work. It is posted here by permission of ACM for your personal use. Not for redistribution. The definitive version was published at GECCO'16, July 20–24, 2016, Denver, CO, USA, http://dx.doi.org/10.1145/2908961.2931709

[1]Because no analytical description of the Pareto sets are known for the `bbob-biobj` suite, we should rather talk about parts of the *best known Pareto set approximations* here.

the same internally as RS-5, the timing per function evaluation for RS-4 and RS-100 are very similar and not shown.

4. RESULTS AND DISCUSSIONS

Results from experiments according to [5], [3] and [1] on the benchmark functions given in [7] are presented in Figures 1, 2, 4 and 5 and in Tables 1 and 2. The experiments were performed with COCO [4], version 1.0.1, the plots were produced with version 1.1.1.

The **average running time (aRT)**, used in the 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} [5, 6]. **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.

As to algorithm performance with respect to the currently best known Pareto set approximations (COCO release 1.1), a few general statements can be made:

First of all, we can observe that RS-100 performs clearly worse than the other two random search variants over all functions, targets, and dimensions displayed, except if targets have not been hit or in the rare cases of exact same performances when the easiest displayed target was hit already in the first evaluation (on f11, f22, and f35 in 5-D, and on f5, f11, f18, and f35 in 20-D). The orders of magnitude worse performance of RS-100 can easily be explained by the vast search volume of $[-100, 100]^n$ in comparison to the volume of $[-4, 4]^n$ where the optima of the single objectives and thus the Pareto fronts' extremes are guaranteed to be located.

Second, when comparing RS-4 and RS-5, slight advantages in favor of the former over the latter can be observed on some problems, except for f12 and f20 where the opposite is the case (all results not significant). The largest difference is observed on the weakly-structured Schwefel/Schwefel problem (f53) where RS-4 is about 10 times faster in 10-D than RS-5.

Overall, it seems as if the search performance of the pure random search is not much affected by the fact that on some problems, the best known Pareto set approximations are located (in part) outside of their sampling boxes. A closer analysis shows that even on problems where the currently best Pareto set approximation lies partly outside of $[-6, 6]^n$ on three of the displayed five instances, namely on the functions f11, f14, and f22 in 5-D, RS-100 is by far worse than the other two algorithms when looking at the ECDFs, see Fig. 3.

5. CONCLUSIONS

We have investigated the influence of the sampling box in which a pure random search is sampling its candidate solutions on the performance on the bi-objective `bbob-biobj` test suite. It turned out that the region of interest $[-100, 100]^n$, suggested by the COCO platform, and in which the currently best known Pareto set approximation is certainly contained, has a too large search volume for the algorithm to be effective. Reducing the sampling box to the suggested region of interest $[-5, 5]^n$ for the single-objective functions of which the `bbob-biobj` test suite is composed of, but which does not guarantee that the Pareto set is entirely inside, increases the performance by orders of magnitude. As a consequence, the default pure random search for future comparisons should be the one that samples within the box $[-5, 5]^n$. Also unbounded algorithms which need a bounded sampling box for its initialization, should consider a smaller sampling box than the suggested region of interest.

6. ACKNOWLEDGMENTS

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

7. REFERENCES

[1] 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.

[2] S. H. Brooks. A discussion of random methods for seeking maxima. *Operations Research*, 6(2):244–251, 1958.

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

[4] 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.

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

[6] 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.

[7] 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.

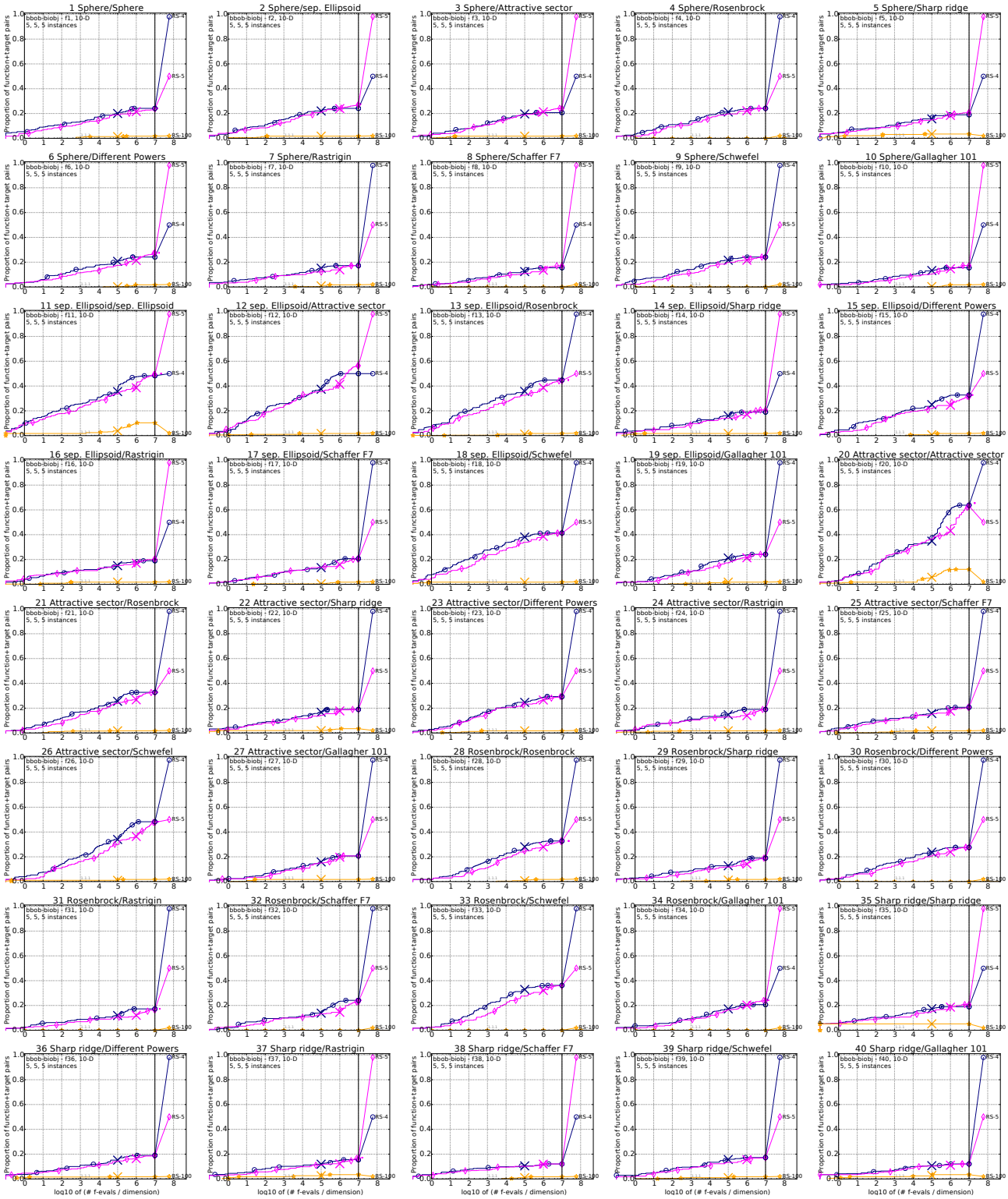

Figure 1: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for 58 targets with target precision in $\{-10^{-4}, -10^{-4.2}, -10^{-4.4}, -10^{-4.6}, -10^{-4.8}, -10^{-5}, 0, 10^{-5}, 10^{-4.9}, 10^{-4.8}, \ldots, 10^{-0.1}, 10^{0}\}$ for each single function f_1 to f_{40} in 10-D.

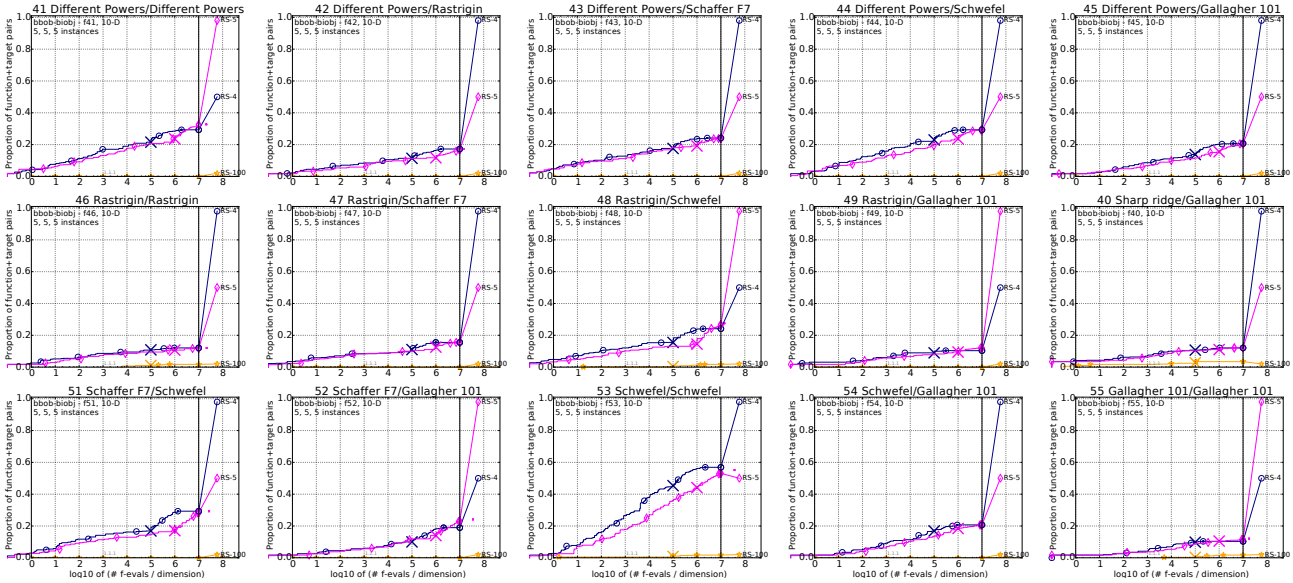

Figure 2: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) as in Fig. 1 but for functions f_{41} to f_{55} in 10-D.

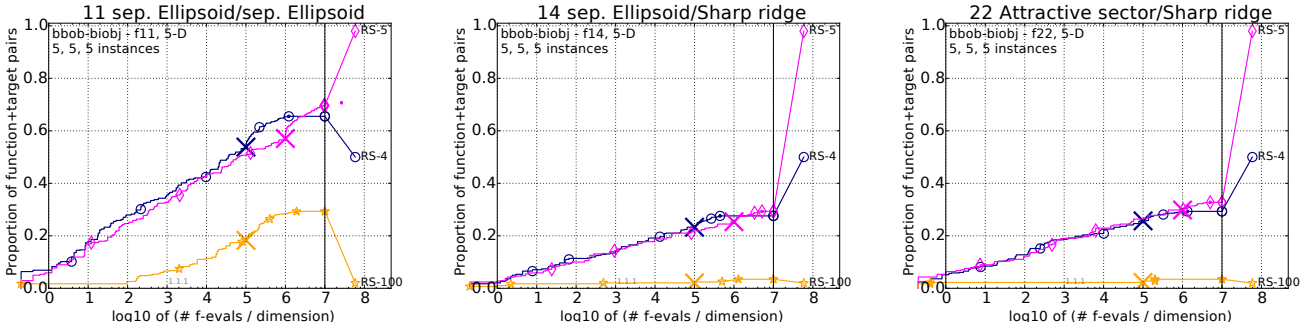

Figure 3: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for the same 58 targets as in Fig. 1, here for three functions in 5-D for which the hypervolume reference sets have solutions outside $[-6, 6]^n$ for three of the five displayed instances.

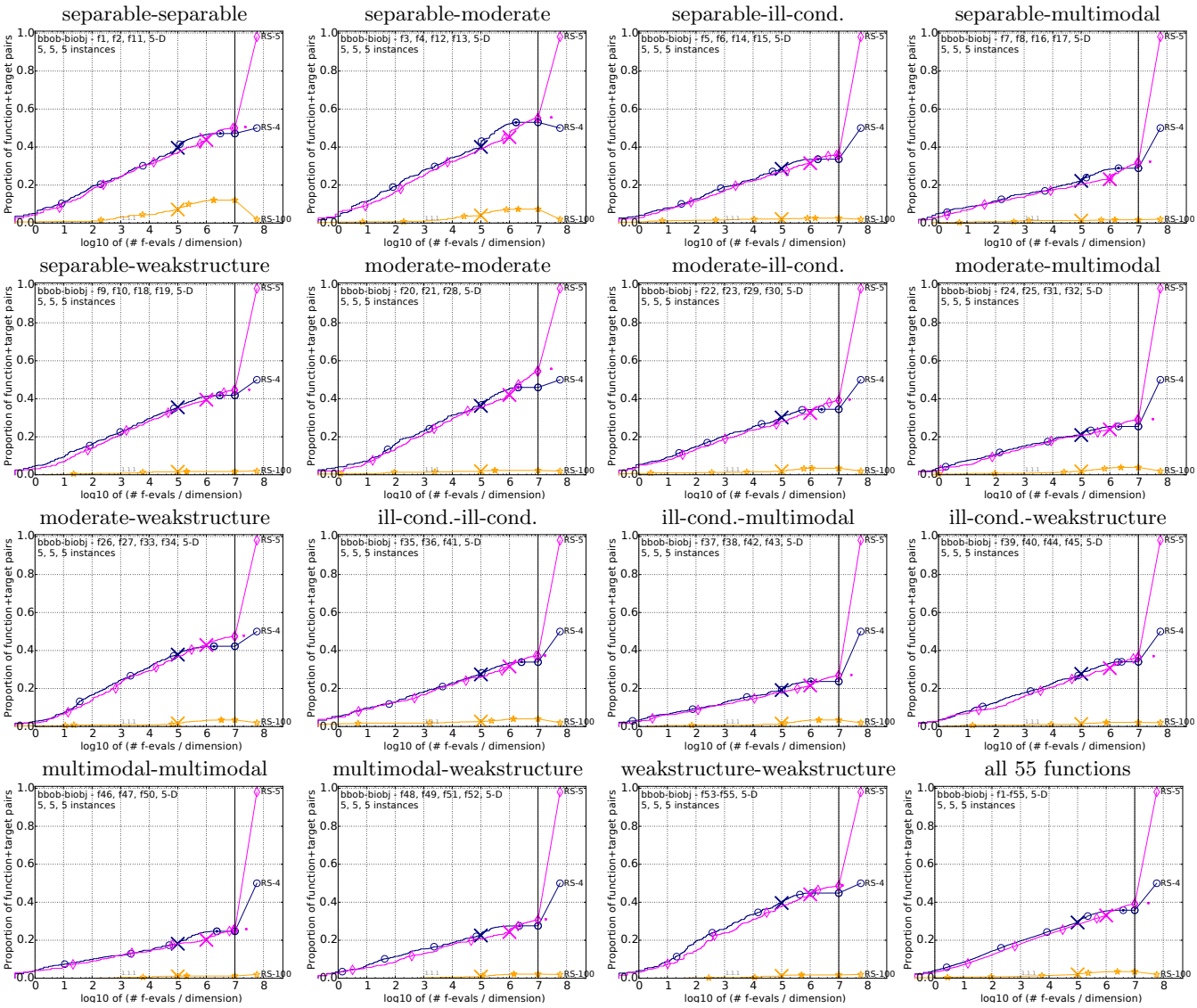

Figure 4: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for 58 targets with target precision in $\{-10^{-4}, -10^{-4.2}, -10^{-4.4}, -10^{-4.6}, -10^{-4.8}, -10^{-5}, 0, 10^{-5}, 10^{-4.9}, 10^{-4.8}, \ldots, 10^{-0.1},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

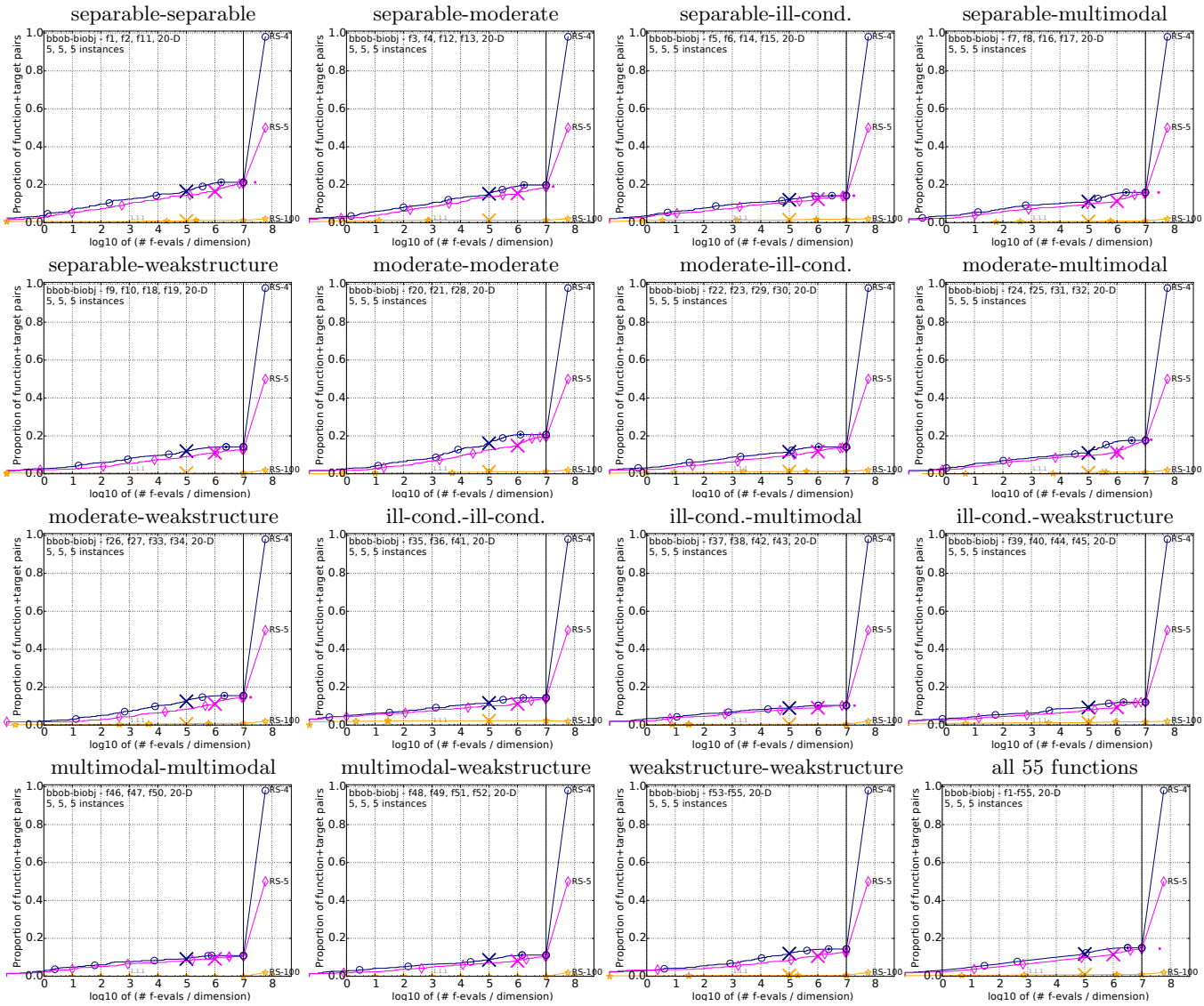

Figure 5: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for 58 targets with target precision in $\{-10^{-4}, -10^{-4.2}, -10^{-4.4}, -10^{-4.6}, -10^{-4.8}, -10^{-5}, 0, 10^{-5}, 10^{-4.9}, 10^{-4.8}, \ldots, 10^{-0.1},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

Δf	1e0	1e-2	1e-5	#succ
f1				
RS-4	**1**(0)	**1.1e6**(1e6)	∞ *5e5*	0/5
RS-5	**1**(0)	3.0e6(4e6)	∞ *5e6*	0/5
RS-100	3.4e4(8e4)	∞	∞ *5e5*	0/5
f2				
RS-4	**1.6**(0.8)	**5.7e5**(7e5)	∞ *5e5*	0/5
RS-5	3.0(2)	3.8e6(4e6)	∞ *5e6*	0/5
RS-100	3440(4551)	∞	∞ *5e5*	0/5
f3				
RS-4	1.2(0)	**2.5e6**(2e6)	∞ *5e5*	0/5
RS-5	**1**(0)	5.1e6(7e6)	∞ *5e6*	0/5
RS-100	1.4e4(3e4)	∞	∞ *5e5*	0/5
f4				
RS-4	**1**(0)	**6.2e5**(3e5)	∞ *5e5*	0/5
RS-5	**1**(0)	1.4e6(2e6)	∞ *5e6*	0/5
RS-100	5.9e4(6e4)	∞	∞ *5e5*	0/5
f5				
RS-4	**1**(0)	∞	∞ *5e5*	0/5
RS-5	**1**(0)	∞	∞ *5e6*	0/5
RS-100	5.6(8)	∞	∞ *5e5*	0/5
f6				
RS-4	**1**(0)	**6.1e5**(3e5)	∞ *5e5*	0/5
RS-5	**1**(0)	4.2e6(4e6)	∞ *5e6*	0/5
RS-100	2.1e4(3e4)	∞	∞ *5e5*	0/5
f7				
RS-4	**1**(0)	∞	∞ *5e5*	0/5
RS-5	**1**(0)	∞	∞ *5e6*	0/5
RS-100	1.1e4(2e4)	∞	∞ *5e5*	0/5
f8				
RS-4	**1.6**(2)	∞	∞ *5e5*	0/5
RS-5	3.8(4)	∞	∞ *5e6*	0/5
RS-100	2.0e6(2e6)	∞	∞ *5e5*	0/5
f9				
RS-4	3.0(5)	**9.8e5**(6e5)	∞ *5e5*	0/5
RS-5	**2.4**(2)	2.4e6(1e6)	∞ *5e6*	0/5
RS-100	1.5e5(3e5)	∞	∞ *5e5*	0/5
f10				
RS-4	**1**(0)	∞	∞ *5e5*	0/5
RS-5	**1**(0)	**2.3e7**(2e7)	∞ *5e6*	0/5
RS-100	9.9e4(2e5)	∞	∞ *5e5*	0/5
f11				
RS-4	**1**(0)	**1.3e5**(3e5)	∞ *5e5*	0/5
RS-5	**1**(0)	4.5e5(1e6)	∞ *5e6*	0/5
RS-100	**1**(0)	∞	∞ *5e5*	0/5
f12				
RS-4	**1**(0)	**3.7e5**(3e5)	∞ *5e5*	0/5
RS-5	**1**(0)	2.1e6(3e6)	∞ *5e6*	0/5
RS-100	281(494)	∞	∞ *5e5*	0/5
f13				
RS-4	**2.8**(4)	**1.1e4**(2e4)	∞ *5e5*	0/5
RS-5	**2.8**(0)	1.3e5(1e5)	∞ *5e6*	0/5
RS-100	1.3e5(3e5)	∞	∞ *5e5*	0/5
f14				
RS-4	**1**(0)	∞	∞ *5e5*	0/5
RS-5	**1**(0)	∞	∞ *5e6*	0/5
RS-100	5.2(4)	∞	∞ *5e5*	0/5
f15				
RS-4	28(65)	**1.0e6**(6e5)	∞ *5e5*	0/5
RS-5	**6.4**(7)	8.4e6(8e6)	∞ *5e6*	0/5
RS-100	1.3e5(3e5)	∞	∞ *5e5*	0/5
f16				
RS-4	**1.4**(0.5)	∞	∞ *5e5*	0/5
RS-5	2.0(2)	**2.2e7**(3e7)	∞ *5e6*	0/5
RS-100	99(112)	∞	∞ *5e5*	0/5
f17				
RS-4	**101**(248)	∞	∞ *5e5*	0/5
RS-5	1702(4250)	∞	∞ *5e6*	0/5
RS-100	1.3e5(3e5)	∞	∞ *5e5*	0/5
f18				
RS-4	**1**(0)	**9.5e4**(1e5)	∞ *5e5*	0/5
RS-5	**1**(0)	4.2e5(1e6)	∞ *5e6*	0/5
RS-100	35(48)	∞	∞ *5e5*	0/5
f19				
RS-4	2.4(2)	2.4e6(2e6)	∞ *5e5*	0/5
RS-5	**1.8**(1)	**5.5e5**(5e5)	∞ *5e6*	0/5
RS-100	2938(3361)	∞	∞ *5e5*	0/5

Δf	1e0	1e-2	1e-5	#succ
f20				
RS-4	**1**(0)	**9.0e5**(1e6)	∞ *5e5*	0/5
RS-5	**1**(0)	1.6e6(3e6)	∞ *5e6*	0/5
RS-100	1.3e5(4e5)	∞	∞ *5e5*	0/5
f21				
RS-4	**1**(0)	**2.0e6**(2e6)	∞ *5e5*	0/5
RS-5	**1**(0)	5.7e6(8e6)	∞ *5e6*	0/5
RS-100	1.2e4(2e4)	∞	∞ *5e5*	0/5
f22				
RS-4	**1**(0)	∞	∞ *5e5*	0/5
RS-5	**1**(0)	∞	∞ *5e6*	0/5
RS-100	**1**(0)	∞	∞ *5e5*	0/5
f23				
RS-4	**1**(0)	**8.8e5**(9e5)	∞ *5e5*	0/5
RS-5	**1**(0)	3.8e6(5e6)	∞ *5e6*	0/5
RS-100	8.4(6)	∞	∞ *5e5*	0/5
f24				
RS-4	1.8(2)	∞	∞ *5e5*	0/5
RS-5	**1.4**(1)	∞	∞ *5e6*	0/5
RS-100	5170(6451)	∞	∞ *5e5*	0/5
f25				
RS-4	**2.4**(4)	∞	∞ *5e5*	0/5
RS-5	3.0(5)	∞	∞ *5e6*	0/5
RS-100	7624(9493)	∞	∞ *5e5*	0/5
f26				
RS-4	6.2(13)	**4.3e5**(3e5)	∞ *5e5*	0/5
RS-5	**3.6**(6)	1.7e6(4e6)	∞ *5e6*	0/5
RS-100	3.3e5(5e5)	∞	∞ *5e5*	0/5
f27				
RS-4	**1.2**(0.5)	∞	∞ *5e5*	0/5
RS-5	2.0(2)	**1.7e6**(2e6)	∞ *5e6*	0/5
RS-100	34(32)	∞	∞ *5e5*	0/5
f28				
RS-4	**1**(0)	**1.5e5**(3e5)	∞ *5e5*	0/5
RS-5	**1**(0)	7.4e5(9e5)	∞ *5e6*	0/5
RS-100	4025(6037)	∞	∞ *5e5*	0/5
f29				
RS-4	**1**(0)	∞	∞ *5e5*	0/5
RS-5	**1**(0)	∞	∞ *5e6*	0/5
RS-100	6504(1e4)	∞	∞ *5e5*	0/5
f30				
RS-4	1(0)	**5.1e5**(7e5)	∞ *5e5*	0/5
RS-5	2.0(2)	3.8e6(6e6)	∞ *5e6*	0/5
RS-100	4.2e5(8e5)	∞	∞ *5e5*	0/5
f31				
RS-4	**1**(0)	∞	∞ *5e5*	0/5
RS-5	**1**(0)	∞	∞ *5e6*	0/5
RS-100	1.6e5(2e5)	∞	∞ *5e5*	0/5
f32				
RS-4	**2.2**(2)	∞	∞ *5e5*	0/5
RS-5	3.8(2)	∞	∞ *5e6*	0/5
RS-100	4.9e5(9e5)	∞	∞ *5e5*	0/5
f33				
RS-4	**3.4**(3)	**1.1e4**(2e4)	∞ *5e5*	0/5
RS-5	4.6(9)	2.4e4(4e4)	∞ *5e6*	0/5
RS-100	8.6e5(4e5)	∞	∞ *5e5*	0/5
f34				
RS-4	**1.6**(2)	**5.7e5**(2e5)	∞ *5e5*	0/5
RS-5	**1.6**(0.2)	1.6e6(3e6)	∞ *5e6*	0/5
RS-100	2.1e5(3e5)	∞	∞ *5e5*	0/5
f35				
RS-4	**1**(0)	∞	∞ *5e5*	0/5
RS-5	**1**(0)	**2.3e7**(2e7)	∞ *5e6*	0/5
RS-100	**1**(0)	∞	∞ *5e5*	0/5
f36				
RS-4	**1**(0)	∞	∞ *5e5*	0/5
RS-5	**1**(0)	∞	∞ *5e6*	0/5
RS-100	1.7e4(4e4)	∞	∞ *5e5*	0/5
f37				
RS-4	**1**(0)	∞	∞ *5e5*	0/5
RS-5	**1**(0)	∞	∞ *5e6*	0/5
RS-100	56(90)	∞	∞ *5e5*	0/5

Δf	1e0	1e-2	1e-5	#succ
f38				
RS-4	**1**(0)	∞	∞ *5e5*	0/5
RS-5	**1**(0)	∞	∞ *5e6*	0/5
RS-100	2.0e4(3e4)	∞	∞ *5e5*	0/5
f39				
RS-4	**1**(0)	∞	∞ *5e5*	0/5
RS-5	**1**(0)	∞	∞ *5e6*	0/5
RS-100	1599(220)	∞	∞ *5e5*	0/5
f40				
RS-4	**1**(0)	∞	∞ *5e5*	0/5
RS-5	**1**(0)	∞	∞ *5e6*	0/5
RS-100	102(174)	∞	∞ *5e5*	0/5
f41				
RS-4	**1**(0)	**4.4e5**(5e5)	∞ *5e5*	0/5
RS-5	3.8(4)	3.6e6(9e6)	∞ *5e6*	0/5
RS-100	2.1e5(4e5)	∞	∞ *5e5*	0/5
f42				
RS-4	**1.6**(2)	∞	∞ *5e5*	0/5
RS-5	3.0(2)	∞	∞ *5e6*	0/5
RS-100	4.2e5(8e5)	∞	∞ *5e5*	0/5
f43				
RS-4	**2.0**(2)	∞	∞ *5e5*	0/5
RS-5	2.2(2)	∞	∞ *5e6*	0/5
RS-100	2.1e6(1e6)	∞	∞ *5e5*	0/5
f44				
RS-4	1.4(0.5)	**6.1e5**(5e5)	∞ *5e5*	0/5
RS-5	**1**(0)	2.9e6(2e6)	∞ *5e6*	0/5
RS-100	7.6e5(9e5)	∞	∞ *5e5*	0/5
f45				
RS-4	**3.8**(4)	∞	∞ *5e5*	0/5
RS-5	**3.8**(6)	**2.4e7**(2e7)	∞ *5e6*	0/5
RS-100	3.1e5(2e5)	∞	∞ *5e5*	0/5
f46				
RS-4	**1**(0)	∞	∞ *5e5*	0/5
RS-5	**1**(0)	∞	∞ *5e6*	0/5
RS-100	1.6e5(5e5)	∞	∞ *5e5*	0/5
f47				
RS-4	1.6(0.8)	∞	∞ *5e5*	0/5
RS-5	**1**(0)	∞	∞ *5e6*	0/5
RS-100	1.9e5(1e5)	∞	∞ *5e5*	0/5
f48				
RS-4	**4.0**(6)	∞	∞ *5e5*	0/5
RS-5	5.6(5)	∞	∞ *5e6*	0/5
RS-100	2.1e6(2e6)	∞	∞ *5e5*	0/5
f49				
RS-4	**1.4**(0.5)	∞	∞ *5e5*	0/5
RS-5	2.0(1)	∞	∞ *5e6*	0/5
RS-100	3.7e4(3e4)	∞	∞ *5e5*	0/5
f50				
RS-4	1.2(0.5)	∞	∞ *5e5*	0/5
RS-5	**1**(0)	∞	∞ *5e6*	0/5
RS-100	∞	∞	∞ *5e5*	0/5
f51				
RS-4	2.0(1)	∞	∞ *5e5*	0/5
RS-5	**1.2**(0.5)	∞	∞ *5e6*	0/5
RS-100	9.4e5(2e6)	∞	∞ *5e5*	0/5
f52				
RS-4	**1**(0)	∞	∞ *5e5*	0/5
RS-5	1.4(0.5)	**2.1e7**(3e7)	∞ *5e6*	0/5
RS-100	1.0e6(2e6)	∞	∞ *5e5*	0/5
f53				
RS-4	30(74)	**6447**(1e4)	∞ *5e5*	0/5
RS-5	**20**(2)	1.8e4(4e4)	∞ *5e6*	0/5
RS-100	3.3e5(1e6)	∞	∞ *5e5*	0/5
f54				
RS-4	1.2(0.5)	**5.8e5**(7e5)	∞ *5e5*	0/5
RS-5	**1**(0)	9.9e5(6e5)	∞ *5e6*	0/5
RS-100	2.0e5(2e5)	∞	∞ *5e5*	0/5
f55				
RS-4	2.8(4)	∞	∞ *5e5*	0/5
RS-5	**1**(0)	**2.2e7**(8e6)	∞ *5e6*	0/5
RS-100	4.0e5(5e5)	∞	∞ *5e5*	0/5

Table 1: 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 $I^{\mathrm{ref}} + 10^{-5}$. 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.

Δf	1e0	1e-2	1e-5	#succ
f1				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
RS-100	∞	∞	∞ *2e6*	0/5
f2				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
RS-100	1.5e6(2e6)	∞	∞ *2e6*	0/5
f3				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
RS-100	3089(7463)	∞	∞ *2e6*	0/5
f4				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
RS-100	∞	∞	∞ *2e6*	0/5
f5				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
RS-100	**1**(0)	∞	∞ *2e6*	0/5
f6				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
RS-100	∞	∞	∞ *2e6*	0/5
f7				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
RS-100	8.0e6(1e7)	∞	∞ *2e6*	0/5
f8				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
RS-100	∞	∞	∞ *2e6*	0/5
f9				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
RS-100	∞	∞	∞ *2e6*	0/5
f10				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
RS-100	∞	∞	∞ *2e6*	0/5
f11				
RS-4	**1**(0)	**9.2e6**(1e7)	∞ *2e6*	0/5
RS-5	**1**(0)	8.6e7(1e8)	∞ *2e7*	0/5
RS-100	**1**(0)	∞	∞ *2e6*	0/5
f12				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
RS-100	6.4(4)	∞	∞ *2e6*	0/5
f13				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
RS-100	208(258)	∞	∞ *2e6*	0/5
f14				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
RS-100	2.6(2)	∞	∞ *2e6*	0/5
f15				
RS-4	1.2(0.2)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
RS-100	3.8e6(5e6)	∞	∞ *2e6*	0/5
f16				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
RS-100	3119(3858)	∞	∞ *2e6*	0/5
f17				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	1.8(2)	∞	∞ *2e7*	0/5
RS-100	∞	∞	∞ *2e6*	0/5
f18				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
RS-100	**1**(0)	∞	∞ *2e6*	0/5
f19				
RS-4	1.4(1)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
RS-100	∞	∞	∞ *2e6*	0/5
f20				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
RS-100	4458(1e4)	∞	∞ *2e6*	0/5
f21				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
RS-100	1.1e6(2e6)	∞	∞ *2e6*	0/5
f22				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
RS-100	4.6(9)	∞	∞ *2e6*	0/5
f23				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	1.2(0.5)	∞	∞ *2e7*	0/5
RS-100	7.4e4(8e4)	∞	∞ *2e6*	0/5
f24				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
RS-100	4.1e4(5e4)	∞	∞ *2e6*	0/5
f25				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	1.2(0.5)	∞	∞ *2e7*	0/5
RS-100	3.1e6(3e6)	∞	∞ *2e6*	0/5
f26				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
RS-100	3.0e6(4e6)	∞	∞ *2e6*	0/5
f27				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
RS-100	2.4e4(5e4)	∞	∞ *2e6*	0/5
f28				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
RS-100	∞	∞	∞ *2e6*	0/5
f29				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
RS-100	∞	∞	∞ *2e6*	0/5
f30				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
RS-100	∞	∞	∞ *2e6*	0/5
f31				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
RS-100	∞	∞	∞ *2e6*	0/5
f32				
RS-4	**37**(0)	∞	∞ *2e6*	0/5
RS-5	613(1526)	∞	∞ *2e7*	0/5
RS-100	∞	∞	∞ *2e6*	0/5
f33				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
RS-100	∞	∞	∞ *2e6*	0/5
f34				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
RS-100	∞	∞	∞ *2e6*	0/5
f35				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
RS-100	**1**(0)	∞	∞ *2e6*	0/5
f36				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
RS-100	2296(5430)	∞	∞ *2e6*	0/5
f37				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
RS-100	177(292)	∞	∞ *2e6*	0/5
f38				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
RS-100	∞	∞	∞ *2e6*	0/5
f39				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
RS-100	426(738)	∞	∞ *2e6*	0/5
f40				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
RS-100	7.8(0.2)	∞	∞ *2e6*	0/5
f41				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	1.2(0.5)	∞	∞ *2e7*	0/5
RS-100	∞	∞	∞ *2e6*	0/5
f42				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
RS-100	∞	∞	∞ *2e6*	0/5
f43				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
RS-100	∞	∞	∞ *2e6*	0/5
f44				
RS-4	3.0(4)	∞	∞ *2e6*	0/5
RS-5	**2.0**(2)	∞	∞ *2e7*	0/5
RS-100	∞	∞	∞ *2e6*	0/5
f45				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
RS-100	∞	∞	∞ *2e6*	0/5
f46				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
RS-100	∞	∞	∞ *2e6*	0/5
f47				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	1.2(0)	∞	∞ *2e7*	0/5
RS-100	∞	∞	∞ *2e6*	0/5
f48				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
RS-100	∞	∞	∞ *2e6*	0/5
f49				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
RS-100	∞	∞	∞ *2e6*	0/5
f50				
RS-4	**1.4**(0.5)	∞	∞ *2e6*	0/5
RS-5	**1.4**(1)	∞	∞ *2e7*	0/5
RS-100	∞	∞	∞ *2e6*	0/5
f51				
RS-4	**1.4**(1)	∞	∞ *2e6*	0/5
RS-5	6.0(4)	∞	∞ *2e7*	0/5
RS-100	∞	∞	∞ *2e6*	0/5
f52				
RS-4	1.2(0.5)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
RS-100	∞	∞	∞ *2e6*	0/5
f53				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
RS-100	1.3e6(2e6)	∞	∞ *2e6*	0/5
f54				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
RS-100	∞	∞	∞ *2e6*	0/5
f55				
RS-4	**1**(0)	∞	∞ *2e6*	0/5
RS-5	**1**(0)	∞	∞ *2e7*	0/5
RS-100	∞	∞	∞ *2e6*	0/5

Table 2: 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.