The Impact of Sample Volume in Random Search on the bbob Test Suite

Dimo Brockhoff, Nikolaus Hansen

The Impact of Sample Volume in Random Search on the bbob Test Suite

Dimo Brockhoff and Nikolaus Hansen
Inria and CMAP, Ecole Polytechnique
Institut Polytechnique de Paris
Palaiseau, France
firstname.lastname@inria.fr

ABSTRACT

Uniform Random Search is considered the simplest of all randomized search strategies and thus a natural baseline in benchmarking. Yet, in continuous domain it has its search domain width as a parameter that potentially has a strong effect on its performance. In this paper, we investigate this effect on the well-known 24 functions from the bbob test suite by varying the sample domain of the algorithm ($[-\alpha, \alpha]^n$ for $\alpha \in \{0.5, 1, 2, 3, 4, 5, 6, 10, 20\}$ and n the search space dimension). Though the optima of the bbob testbed are randomly chosen in $[-4, 4]^n$ (with the exception of the linear function f_5), the best strategy depends on the search space dimension and the chosen budget. Small budgets and larger dimensions favor smaller domain widths.

CCS CONCEPTS

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

KEYWORDS

Benchmarking, Black-box optimization

ACM Reference format:
Dimo Brockhoff and Nikolaus Hansen. 2019. The Impact of Sample Volume in Random Search on the bbob Test Suite. In *Proceedings of Genetic and Evolutionary Computation Conference Companion, Prague, Czech Republic, July 13–17, 2019 (GECCO '19 Companion),* 8 pages.
DOI: 10.1145/3319619.3326894

1 INTRODUCTION

In continuous optimization, the simplest *Random Search* algorithm samples uniformly at random from a given subdomain S from the search space $\mathbb{R}^n$. Its performance is often used as a baseline when benchmarking more advanced optimization algorithms, and thus also has been benchmarked as one of the first algorithms in the context of the Comparing Continuous Optimizers platform (COCO, [6]).

GECCO '19 Companion, Prague, Czech Republic

978-1-4503-6748-6/19/07…$15.00
DOI: 10.1145/3319619.3326894

The RANDOMSEARCH as submitted to the BBOB-2009 workshop [2] sampled uniformly in the hypercube $[-5, 5]^n$ where n is the search space dimension. The choice of $[-5, 5]^n$ as sampling domain was motivated by the bounded definitions of the test functions in the bbob test suite [7] which is based on their construction: all optima are known to be located within this interval, while for all but the linear function (f_5), the optimum lies even within the hypercube $[-4, 4]^n$. Already in the context of the biobjective extension of the bbob test suite [9], it has been noted that the search domain of random search has a strong impact on the search performance [1].

In this paper, we will investigate this effect a bit further on the single-objective bbob test suite. We will in particular investigate the question which search domain (more concretely which search volume around the search space origin) results in the best overall performance for random search. A more detailed analysis will allow to see where these performance differences occur and what can be learned by these observations about the bbob test functions. In the following, we distinguish the algorithms by their sample domain and denote whether the search space origin 0^n has been evaluated as the first search point. The algorithm is then denoted by RS-α-initIn0 and RS-α respectively if the search domain is $[-\alpha, \alpha]^n$.

Note that another way to look at our investigations is that random search measures the volume of the sublevel sets for any given target. With this in mind, we actually investigate rather properties of the bbob functions than the performance of the random search, because we precisely understand the latter.

2 CPU TIMING OF RANDOM SEARCH

In order to evaluate the CPU timing of the algorithm, we have run the random search on the bbob test suite [7] with varying sample domains $[-\alpha, \alpha]^n$ for $\alpha \in \{0.5, 1, 2, 3, 4, 5, 6, 10, 20\}$ for a maximum budget equal to $10^3 n$ function evaluations according to [8]. For the final experiments, we run all algorithms up to a budget of $10^6 n$ function evaluations except for RS-4 and RS-5 for which we use previously available data sets from COCO's data archive that have been run for a budget of $10^7 n$.

The Python code of the COCO example experiment was run on a linux machine with 64 Intel(R) Xeon(R) CPU E5-2683 v4 @ 2.10GHz processors on which other processes have been running during the timing experiment. The time per evaluation is shy of 10 microseconds up to dimension 10 and grows sublinear with dimension afterwards. For larger dimensions we naturally expect

linear growth.[1] The function evaluation itself however may take already a quite significant fraction of this time.

3 RESULTS

Results from experiments according to [8] and [5] on the benchmark functions given in [4, 7] are presented in Figures 1, 2, 3, 4, 5, and 6. The experiments were performed with COCO [6], version 2.2.1, the plots were produced with version 2.2.2. The entire data with all plots and tables can be consulted at the urls randopt.gforge.inria.fr/ppdata-archive/2019-RS/ppdata-RSall/ and randopt.gforge.inria.fr/ppdata-archive/2019-RS/ppdata-RSall-initIn0/.

4 OBSERVATIONS

The following observations are made from the data.

Global Performance Differences. In the aggregated empirical runtime distribution plots over all 24 bbob functions, some clear tendencies can be observed. This is best seen for dimension 3 in Figure 1 with the visible differences becoming smaller in higher dimension.[2]

Unsurprisingly, the results generally depend on the budget and the dimension. In 5-D, RS-3 solves in comparison the most problems with a budget of 10 and 100 × dimension, RS-4 with a budget of 1000 and 10,000 × dimension and RS-6 for larger budgets. In 20-D, RS-3 solves in comparison the most problems with a budget of 100 × dimension, RS-4 with 1000 × dimension, and RS-6 with larger budgets. These observations suggest that the easier target values are biased towards the center of the search space.

Note again here that the optima of the bbob functions are placed uniformly at random in the hyperbox $[-4, 4]^n$ with the exception of the linear slope function for which the optimum lies at a corner of the hyperbox $[-5, 5]^n$ or even outside of it.

With increasing budget, only strategies with $\alpha \geq 5$ can be optimal, as only those are able to eventually solve all target values. However, log-linear extrapolation suggests a necessary evaluations budget of roughly $10^{n\times 0.05p}$ to $10^{n\times 0.1p}$ evaluations to solve p percent of all problems which is even in moderate dimension far beyond any feasible number of evaluations to solve, say, 90% of all problems.

A too large sample volume decreases the performance for larger budgets because outside of $[-5, 5]^n$, good targets can be hit only on the linear function: RS-10 and RS-20 are clearly worse than RS-6.

Interesting are the slopes of the empirical runtime distributions: with small sample volume, the slopes of the ECDFs decrease with larger budgets, whereas for sample volumes close to the recommended "region of interest" of $[-5, 5]^n$ the slopes are roughly constant over the number of function evaluations.

Surprising in this context is the good performance of RS-6 that outperforms the other tested variants with smaller sample space when the budget is high(er), i.e. for budgets larger than about $3{\cdot}10^4\times$ dimension evaluations in dimension 5. The observable upsurge at 30 000 × dimension evaluations in the empirical runtime distribution can be attributed to a great extend to the optimization of the linear slope function (f_5) that will be discussed below (see also Figures 2 and 3). A similar upsurge can be observed earlier for larger α. In 20-D however, the budget is too small to observe the upsurge at all.

Evaluating the Search Space Origin. It has been noted that the search space origin $(0, \ldots, 0) \in \mathbb{R}^n$ is, by construction, an especially good search point, in particular for the Griewank Rosenbrock function (f_{19}). In order to not disfavor algorithms that do not evaluate this distinct solution in the beginning of the benchmarking, some example experiments of COCO evaluate the search space origin by default as the first search point. To compare the effect of this evaluation, we also re-run all random search variants with the origin evaluated before the uniform sampling starts. The corresponding algorithms are denoted with the suffix "-initIn0" in the supplementary material which, due to space limitations, we only show selectively in this paper as in Figure 1.

The main difference from evaluating the origin is observed on the Griewank Rosenbrock function f_{19} where evaluating the initial search point $(0, \ldots, 0)$ reaches about 25% of the targets whereas RS-0p5 reaches maximally about 16% of the targets in the first evaluation and the percentage decreases with increasing α falling below 10% for $\alpha \geq 3$ (with slightly decreasing percentage in higher dimensions), compare Figure 6. When not evaluating the origin first, the random search needs some time to reach the same percentage of solved targets. Afterwards, both algorithms show again the same performance for larger budgets. In the larger dimensions, the experiments' budget was not high enough for random search to reach 25% of the targets such that the period of similar performance cannot be observed.

In the following, we provide further observations on single functions that we find remarkable and that let us understand some of the properties of the bbob functions.

The linear slope function. The linear slope function (f_5) is the only bbob function that does not have its optimum in $[-4, 4]^n$ but instead at one of the corners of the hypercube $[-5, 5]^n$ and beyond this corner where each variable is either ≤ -5 or ≥ 5. This explains why random search variants with sample volume larger than $[-5, 5]^n$ perform best on this function.

The linear function is the only one where in dimensions up to 10 all targets can be reached by some variants in the experiment budget. However, in 20-D, even the random search variant sampling in $[-20, 20]^n$ cannot solve more than about 16% of all targets in $10^6\times$ dimension evaluations. For $\alpha \to \infty$, the probability to hit the final target approaches $2^{-n} \approx 10^{-0.3n}$, hence a budget of somewhat above $10^{0.3n}$ should suffice.

Katsuuras function. The Katsuuras function (f_{23}) shows the largest percentage of solved targets in higher dimension for all tested random search variants except for RS-10 and RS-20. Except for RS-10 and RS-20 all variants show comparable performance. This is expected as the function is repetitive within the hypercube $[-\alpha, \alpha]^n$ with $\alpha \leq 5$.

Similar as on the Schaffer function f_{17}, the first evaluation in the domain $[-\alpha, \alpha]^n$ with $\alpha \leq 5$ solves comparably many targets like evaluating the origin.

[1] The actual wall clock times per function evaluation vary little with the different variants and are, for fixed dimension, mostly influenced by the other load on the machine: 8.6–9.6 microseconds (μs) for dimension 2, 8.5–9.6 μs for dimension 3, 6.9–9.7 μs for dimension 5, 9.5–11 μs for dimension 10, 13–14 μs for dimension 20, and 23–24 μs for dimension 40.

[2] Most clear are the differences in dimension 2 (not shown here) with the same overall tendencies.

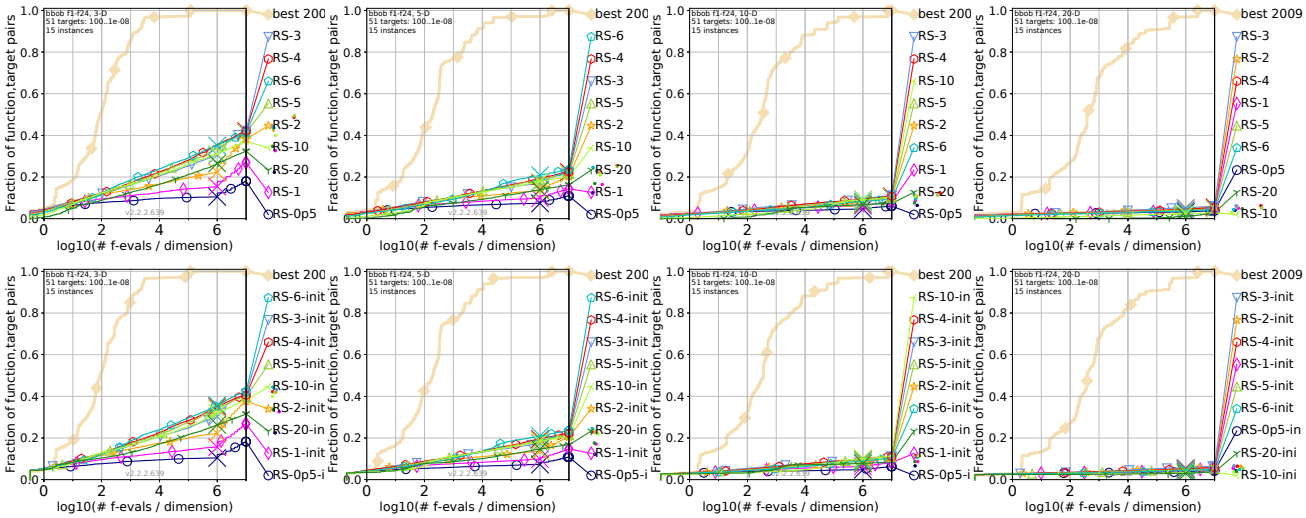

Figure 1: Empirical cumulative distribution functions of runtimes in dimensions 3, 5, 10, and 20 (from left to right) for various random search variants, aggregated over all 24 bbob functions and 51 target precisions $100 \ldots 10^{-8}$. The first row shows the standard random search, sampling uniformly in $[-\alpha, \alpha]^n$ with α indicated in the algorithm name RS-α while the second row shows the variants with the origin evaluated first.

Compared to the algorithm variants that evaluate the search space origin first, we see another effect that is different from the Griewank Rosenbrock function, discussed above: for the function value of the initial search point it does not make a difference whether it is sampled within $[-5, 5]^n$ or chosen as the origin. A difference, however, occurs for the RS-10 and RS-20 variants (not shown here): in the case of RS-20, no other search point better than the origin is found in the entire experiment in 20-D and for RS-10, it takes about $6 \cdot 10^3$ function evaluations to find a better target than in the first evaluation (also in 20-D). This effect is smaller in lower and larger in higher dimension. In this sense, we can conclude that for the Katsuuras function, in contrast to the Griewank Rosenbrock function, the search space origin has no exceptionally good function value compared to a random one within the hypercube $[-5, 5]^n$ but that samples outside of this hypercube have exceptionally low function values.

The Gallagher functions. Besides the Griewank Rosenbrock and Katsuuras functions, the Gallagher functions show the best performance in low dimensions, solving all or almost all targets for variants RS-4, RS-5, and RS-6 in dimension 2 and showing the best performance over all functions (except for the linear slope) in dimension 5 with about 50% of the targets solved for RS-3 and RS-4.

This good performance, however, is not observable in higher dimensions, where for example in 20-D, the performance on the Weierstrass function, the Schaffer function with condition number 10, the Griewank Rosenbrock, and the Katsuuras functions are better. Also the performance on the sphere function is slightly better in 20-D than for the Gallagher function with 21 peaks in the same dimension and for easier targets.

5 CONCLUSIONS

Despite being one of the simplest stochastic search variants, uniform random search has an internal parameter, its sample volume, that plays an important role for its performance. We have compared different variants of random search on the bbob test suite and observed some significant differences on some functions that resulted in some insights into the construction of the function. Not surprisingly, only for the single bbob function that has its optimum at one of the corners or outside the hypercube $[-5, 5]^n$ it is best to sample in a large volume. For some other functions such as the rotated Rosenbrock function and the Griewank Rosenbrock function, we observed that a smaller sample volume around the search space origin is better for budgets up to $10^6 \times$ dimension.

Over all functions, the performance of RS-3, RS-4, RS-5, RS-6, and RS-10 is surprisingly similar, however also depending on the budget and dimension. The best performance is observed for the variants RS-4, RS-5, and in smaller dimension also RS-6 due to the better performance on the linear function f_5.

Evaluating the search space origin as first solution has an additional advantage on several functions and in particular on the Griewank Rosenbrock function, where the origin has an exceptionally good function value by construction—an issue that has been corrected in the recent `bbob-largescale` test suite [3].

The sample volume of RS-5 is in dimension 5, 10, and 20 about 3, 9, and 87 times larger than that of RS-4. That means, for example, with a 9 times larger budget, RS-5 will perform at least on par with RS-4 in dimension 10.

ACKNOWLEDGEMENTS

This work was supported by a public grant as part of the Investissement d'avenir project, reference ANR-11-LABX-0056-LMH, LabEx

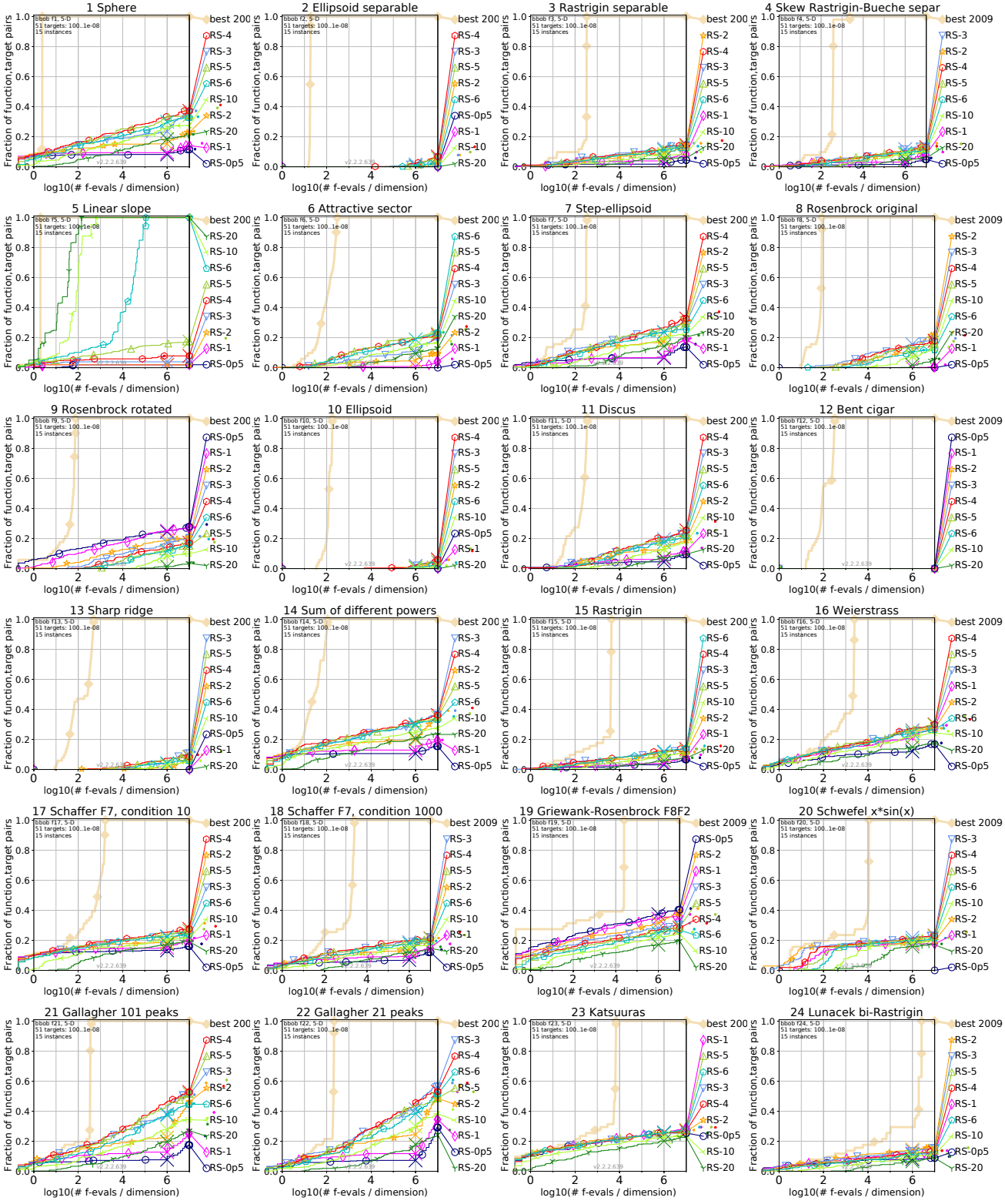

Figure 2: Empirical cumulative distribution of simulated (bootstrapped) runtimes, measured in number of objective function evaluations, divided by dimension (FEvals/DIM) for the 51 targets $10^{[-8..2]}$ in dimension 5.

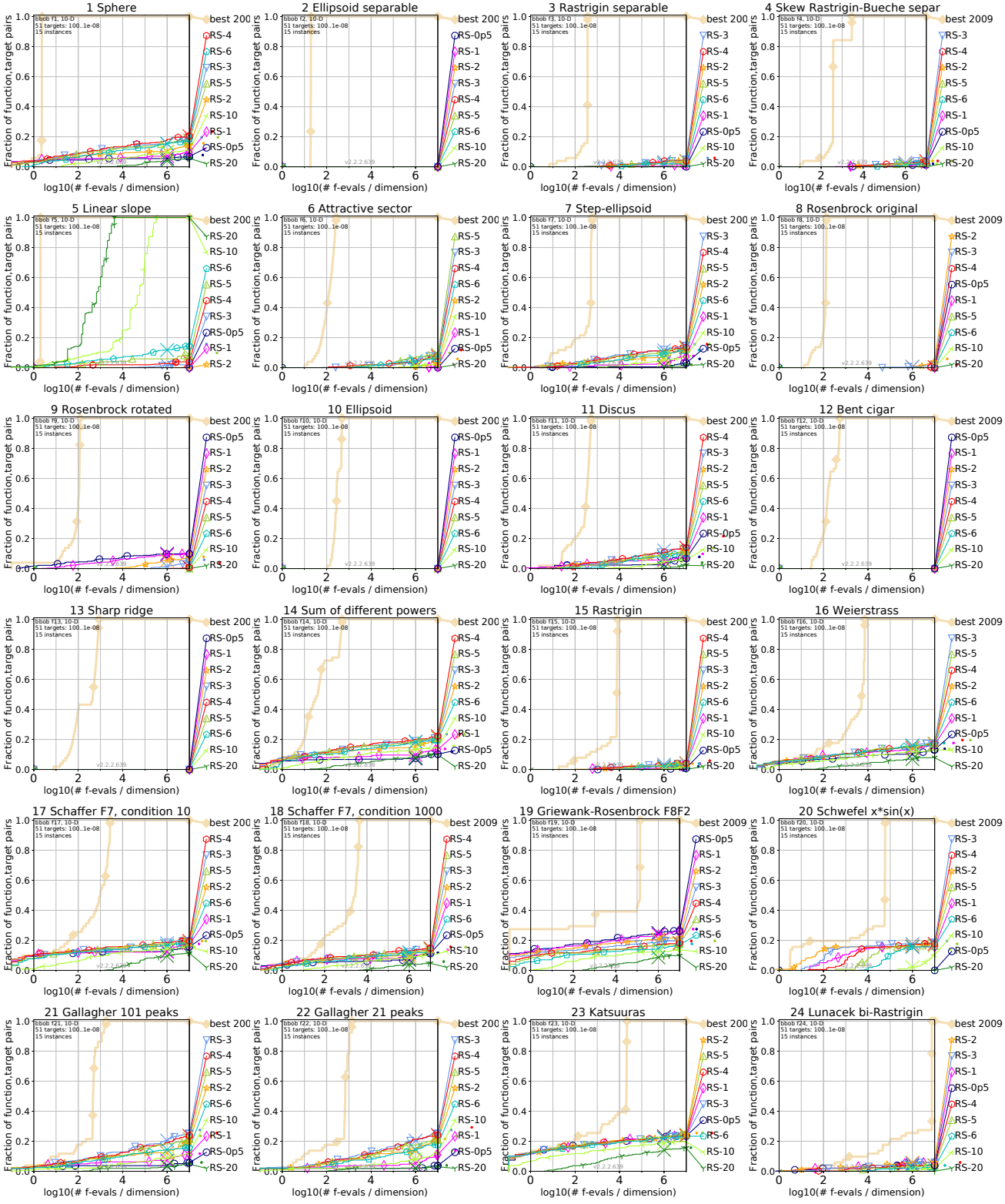

Figure 3: Empirical cumulative distribution of simulated (bootstrapped) runtimes, measured in number of objective function evaluations, divided by dimension (FEvals/DIM) for the 51 targets $10^{[-8..2]}$ in dimension 10.

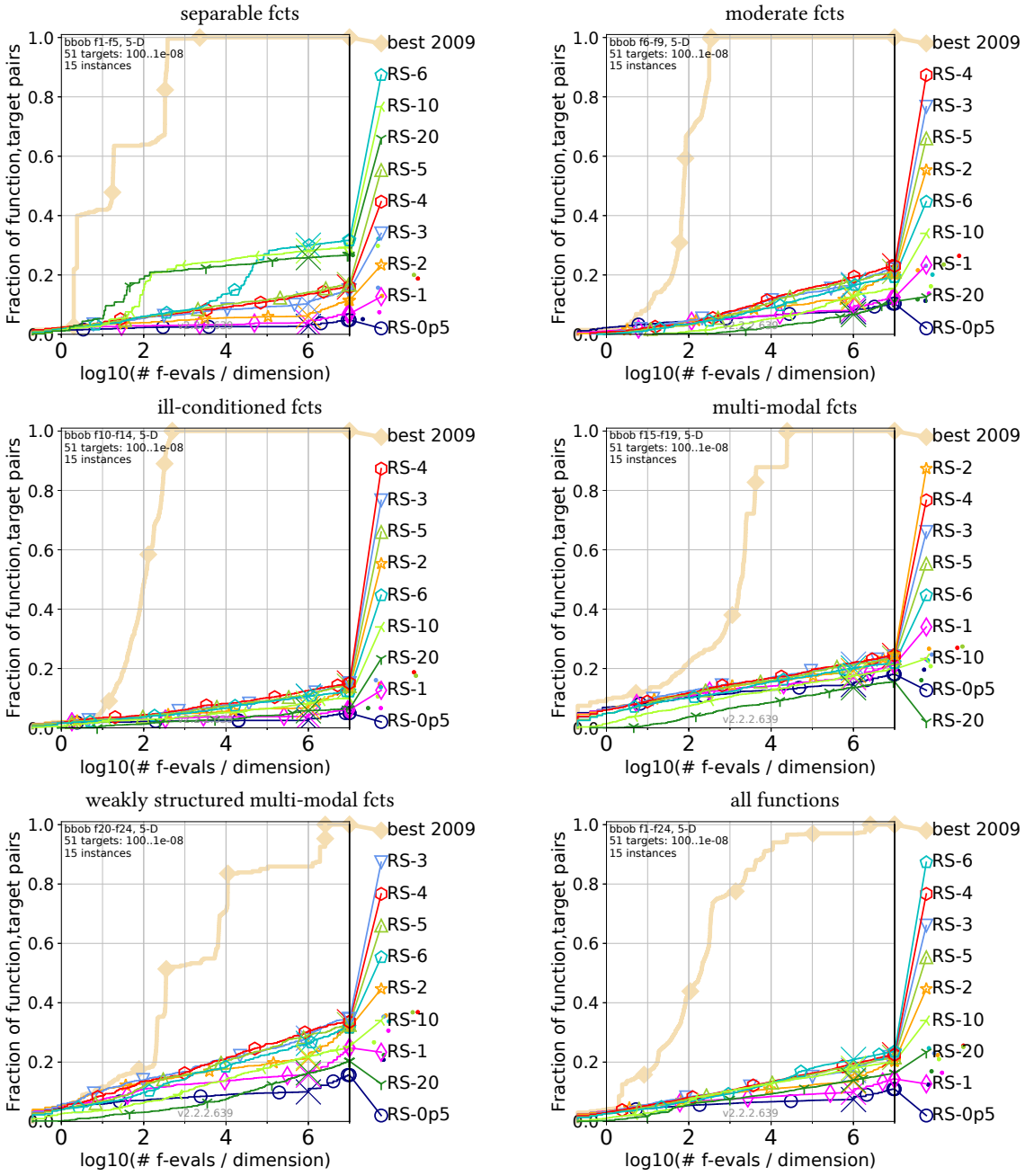

Figure 4: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for 51 targets with target precision in $10^{[-8..2]}$ for all functions and subgroups in 5-D. As reference algorithm, the best algorithm from BBOB 2009 is shown as light thick line with diamond markers.

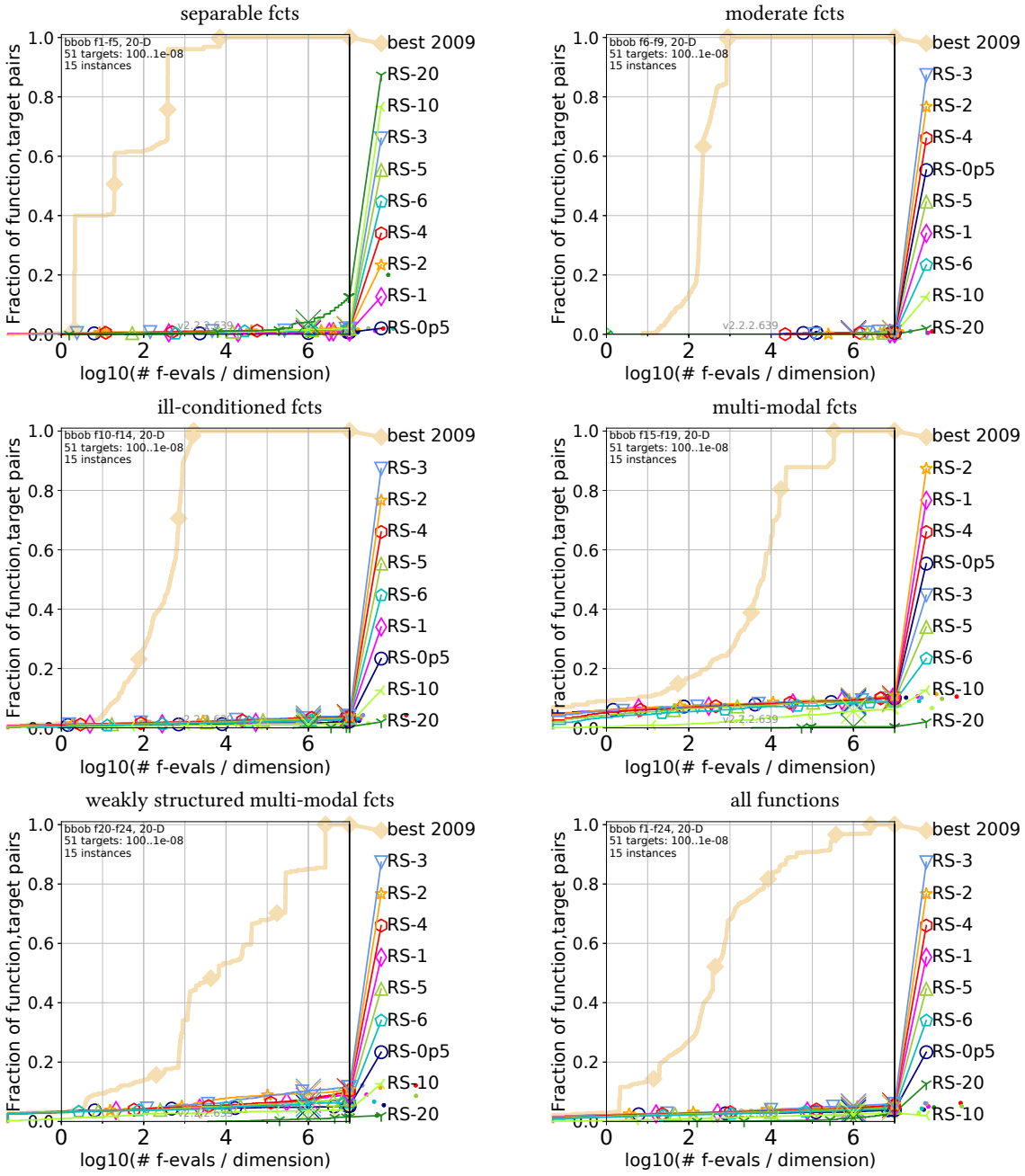

Figure 5: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for 51 targets with target precision in $10^{[-8..2]}$ for all functions and subgroups in 20-D. As reference algorithm, the best algorithm from BBOB 2009 is shown as light thick line with diamond markers.

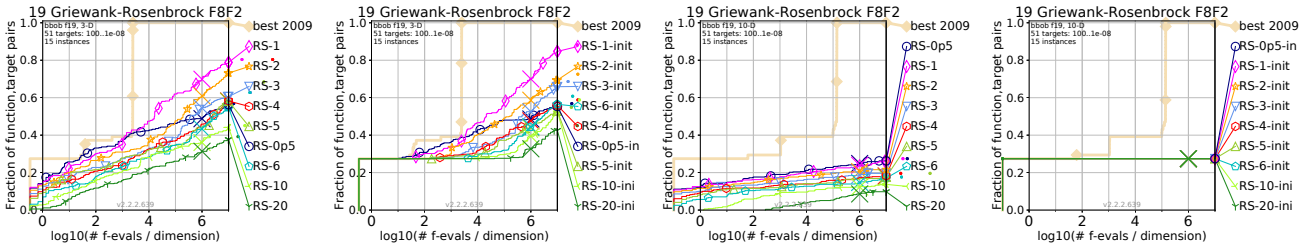

Figure 6: Empirical cumulative distribution functions of the runtime to reach certain targets on the Griewank-Rosenbrock function for the random search variants without (first and third plot) and with evaluating the origin first (second and forth) in dimension (left two plots) and dimension 10 (right two plots).

LMH, in a joint call with Gaspard Monge Program for optimization, operations research and their interactions with data sciences.

REFERENCES

[1] Anne Auger, Dimo Brockhoff, Nikolaus Hansen, Dejan Tušar, Tea Tušar, and Tobias Wagner. 2016. The impact of search volume on the performance of RANDOMSEARCH on the bi-objective BBOB-2016 test suite. In *Genetic and Evolutionary Computation Conference Companion Proceedings (GECCO 2016)*. ACM, 1257–1264.

[2] Anne Auger and Raymond Ros. 2009. Benchmarking the pure random search on the BBOB-2009 testbed. In *GECCO (Companion)*, Franz Rothlauf (Ed.). ACM, 2479–2484.

[3] Ouassim Elhara, Konstantinos Varelas, Duc Nguyen, Tea Tusar, Dimo Brockhoff, Nikolaus Hansen, and Anne Auger. 2019. COCO: The Large Scale Black-Box Optimization Benchmarking (bbob-largescale) Test Suite. *arXiv preprint arXiv:1903.06396* (2019).

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

[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, S. Finck, R. Ros, and A. Auger. 2009. *Real-Parameter Black-Box Optimization Benchmarking 2009: Noiseless Functions Definitions.* Technical Report RR-6829. INRIA. http://coco.lri.fr/downloads/download15.03/bbobdocfunctions.pdf Updated February 2010.

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

[9] Tea Tušar, Dimo Brockhoff, Nikolaus Hansen, and Anne Auger. 2016. COCO: the bi-objective black box optimization benchmarking (BBOB-BIOBJ) test suite. *ArXiv e-prints* arXiv:1604.00359 (2016).