Comparison of Cauchy EDA and Rosenbrock's Algorithms on the BBOB Noiseless Testbed

Petr Pošík
Czech Technical University in Prague
Faculty of Electrical Engineering, Department of Cybernetics
Technická 2, 166 27 Prague 6, Czech Republic
posik@labe.felk.cvut.cz

ABSTRACT

Estimation-of-distribution algorithm equipped with Cauchy distribution (Cauchy EDA) is compared with Rosenbrock's local search algorithm. Both algorithms were already presented at the 2009 black-box optimization benchmarking workshop where Cauchy EDA usually ranked better than Rosenbrock's algorithm. This paper compares them in more detail and adds to the understanding of their key differences.

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

General Terms

Algorithms

Keywords

Benchmarking, Black-box optimization, Estimation-of-distribution algorithm, Cauchy distribution, Rosenbrock's algorithm

1. INTRODUCTION

During the black-box optimization benchmarking (BBOB) workshop in 2009 many diverse algorithm were benchmarked on a well-prepared set of functions using common conditions. The BBOB 2010 methodology [2] provides additional means to compare two algorithms in more detail. In this paper, two algorithms which took part in the BBOB 2009 workshop are further compared. Data for both algorithms were taken from the 2009 benchmarking, but the comparison is made using the new post-processing scripts and templates for BBOB 2010.

The two algorithms chosen for the comparison are:

- Rosenbrock's algorithm introduced in [8]. It could be described as an adaptive pattern search. Its performance on the BBOB 2009 noiseless test suite was reported in [6].
- The estimation-of-distribution algorithm (EDA) with Cauchy sampling distribution (Cauchy EDA) [5]. It represents the class of evolutionary optimization algorithms.

Despite their different origins, both algorithms maintain the model of local neighborhood. It is interesting to see if there exists any systematic difference resulting from the different adaptation mechanisms and other differences between the algorithms, or if the similar principle of maintaining the model of the local neighborhood also unifies the performance of both algorithms.

In the next section, both algorithms are shortly described and their differences are emphasized. Sec. 3 contains all the results used to compare the algorithms and their discussions. After the presentation of the time demands of both algorithms in Sec. 4, Sec. 5 concludes the paper.

2. ALGORITHM PRESENTATION

The exact descriptions of the algorithms along with the parameter settings can be found in [6] and [5], respectively. The main differences between the algorithms are:

- The Cauchy EDA is a population based algorithm, while Rosenbrock's algorithm maintains the best-so-far solution only.
- The Cauchy EDA updates the model of the local neighborhood on the basis of (a relatively large set of) selected individuals each generation. Rosenbrock's algorithm updates the model (orthonormal basis) only in strictly defined situations; the time periods which use the same model may last varying number of iterations.
- The Cauchy EDA uses the Cauchy distribution to sample new candidate solutions. Rosenbrock's algorithm uses strictly defined pattern to sample new candidates; it iterates over all axes of the orthonormal basis and generates one solution in the respective direction in a particular iteration.
- To fight the premature convergence, it uses a constant multiplier to enlarge the variance of the distribution (as suggested in [4]). Rosenbrock's algorithm needs no such modification.

For both algorithms, the crafting effort CrE= 0.

3. RESULTS

Results from experiments according to [2] on the benchmark functions given in [1, 3] are presented in Figures 1, 2 and 3 and in Table 1. The **expected running time (ERT)**, used in the figures and table, depends on a given target function value, $f_\mathrm{t} = f_\mathrm{opt} + \Delta f$, and is computed over all relevant trials as the number of function evaluations executed during each trial while the best function value did not reach f_t, summed over all trials and divided by the number of trials that actually reached f_t [2, 7]. **Statistical significance** is tested with the rank-sum test for a given target Δf_t (10^{-8} in Figure 1) using, for each trial, either the number of needed function evaluations to reach Δf_t (inverted and multiplied by -1), or, if the target was not reached, the best Δf-value achieved, measured only up to the smallest number of overall function evaluations for any unsuccessful trial under consideration.

Rosenbrock's algorithm ouperforms Cauchy EDA on functions 1, 2, 5, 6, 20, 21, 22, and 23 (it is often faster, but Cauchy EDA can solve many of these functions as well), while Cauchy EDA beats G3PCX on functions 7, 10, 11, 13, 17, and 18 (where Rosenbrock's often fails to find the target level), i.e. in this small competition Rosenbrock's algorithm wins 8:6. (The results on the other functions are mixed, or neither algorithm solved the problem successfully.)

In Fig. 1, we can sometimes see a peak (upward or downward) at the beginning of the ERT ratio lines (functions 1, 5, 13, 14). In the beginning of the evolution, both algorithms need to adapt their models to the fitness landscape. The peak means that Cauchy EDA (upward) or Rosenbrock's algorithm (downward) probably needs more time for this adaptation.

Interesting situation can be seen when comparing results for functions 2 and 10 (separable and non-separable version of the ellipsoid function). In the first case, the model of Rosenbrock's algorithm is actually initialized to the correct one, and it does not need to be adapted (however, it is not clear if any adaptation takes place or not). Thanks to this correct initialization, Rosenbrock's algorithm is faster for the separable problem. The picture for non-separable ellipsoid is, however, completely different. For 2- and 3-dimensional version, Rosebrock's algorithm is still about 2 times faster than Cauchy EDA, but for dimensions 5 and higher Rosenbrock's adaptation mechanism loses its efficiency and the algorithm becomes very slow.

Rosenbrock's algorithm failed on functions 7 (Step-ellipsoid) and on both Schaffer's functions 17 and 18. It seems that for these functions the batch model adaptation using tens or hundreds samples is a better approach than sequential adaptation after each sampled point.

Interesting results may be found for functions 13 (sharp ridge problem) and 14 (sum of different powers): Rosenbrock's algorithm is orders of magnitude faster than Cauchy EDA for a broad range of target levels. But for target levels at about 10^{-5} and tighter, Cauchy EDA takes over and its results are much better. It seems that Rosenbrock's algorithm is not even able to find some of the tighter target levels.

Another note can be made on the variance enlargment constant used by Cauchy EDA. It was set to be approximately optimal for Rosenbrock's function. However, such setting may be too large for other functions. The ERT ratio for sphere function shows that with increasing problem dimensionality the gap between the algorithms gets larger. Also the results for function 21 and 22 (and possibly for function 23) suggest, that the slow convergence of Cauchy EDA prevents it to be restarted more often which is the key to solve these problems; on the contrary, Rosenbrock's algorithm converges probably much faster and is thus restarted more often which gives it a chance to have higher success rate.

Looking at Fig. 3, it can be stated that Rosenbrock's algorithm beats CauchyEDA mainly on the separable functions and on weak structure functions. On the other hand, Cauchy EDA wins on moderate, ill-conditioned and multimodal functions. It can be stated that if Rosenbrock's algorithm solves a problem, it solves it very quickly, while Cauchy EDA is usually much slower but more robust (is able to solve broader range of problems).

Table 2: The average time demands per function evaluation (in microseconds) of the two compared algorithms.

Dim	2	3	5	10	20	40
Rosenbrock	310	312	320	334	350	370
CauchyEDA	51	17	9	9	11	NA

4. CPU TIMING EXPERIMENTS

The time requirements of both algorithms are taken from the respective articles, [6] and [5]. The multistart algorithm was run with the maximal number of evaluations set to 10^5, the basic algorithm was restarted for at least 30 seconds. The experiment was conducted on Intel Core 2 CPU, T5600, 1.83 GHz, 1 GB RAM with Windows XP SP3 in MATLAB R2007b. The comparison of the average time demands per function evaluation are shown in Table 2.

The differences in the average time needed for function evaluation are caused by the frequency of calling the evaluation function and by the size of the solution set to be evaluated. While Rosenbrock's algorithm evaluates the solutions one by one, Cauchy EDA uses batches of tens or hundreds of solutions which means that the evaluation routine is called less often and can take advantage of the MATLAB matrix processing capabilities to much larger extent.

5. CONCLUSIONS

The results indicate that neither algorithm dominates the other. In cases when Rosenbrock's algorithm is able to find the solution, it finds it quickly. The Cauchy EDA is slower, but can find the solution for a broader set of functions. Of course, this can be attributed to the fact that Cauchy EDA uses a population, while Rosenbrock's algorithm maintains only single point. This observation can be expected for many comparisons of single-point vs. population-based methods—the fact that both algorithms use a model of the local neighborhood does not change this expectation

Acknowledgements

The author is supported by the Grant Agency of the Czech Republic with the grant no. 102/08/P094 entitled "Machine learning methods for solution construction in evolutionary algorithms".

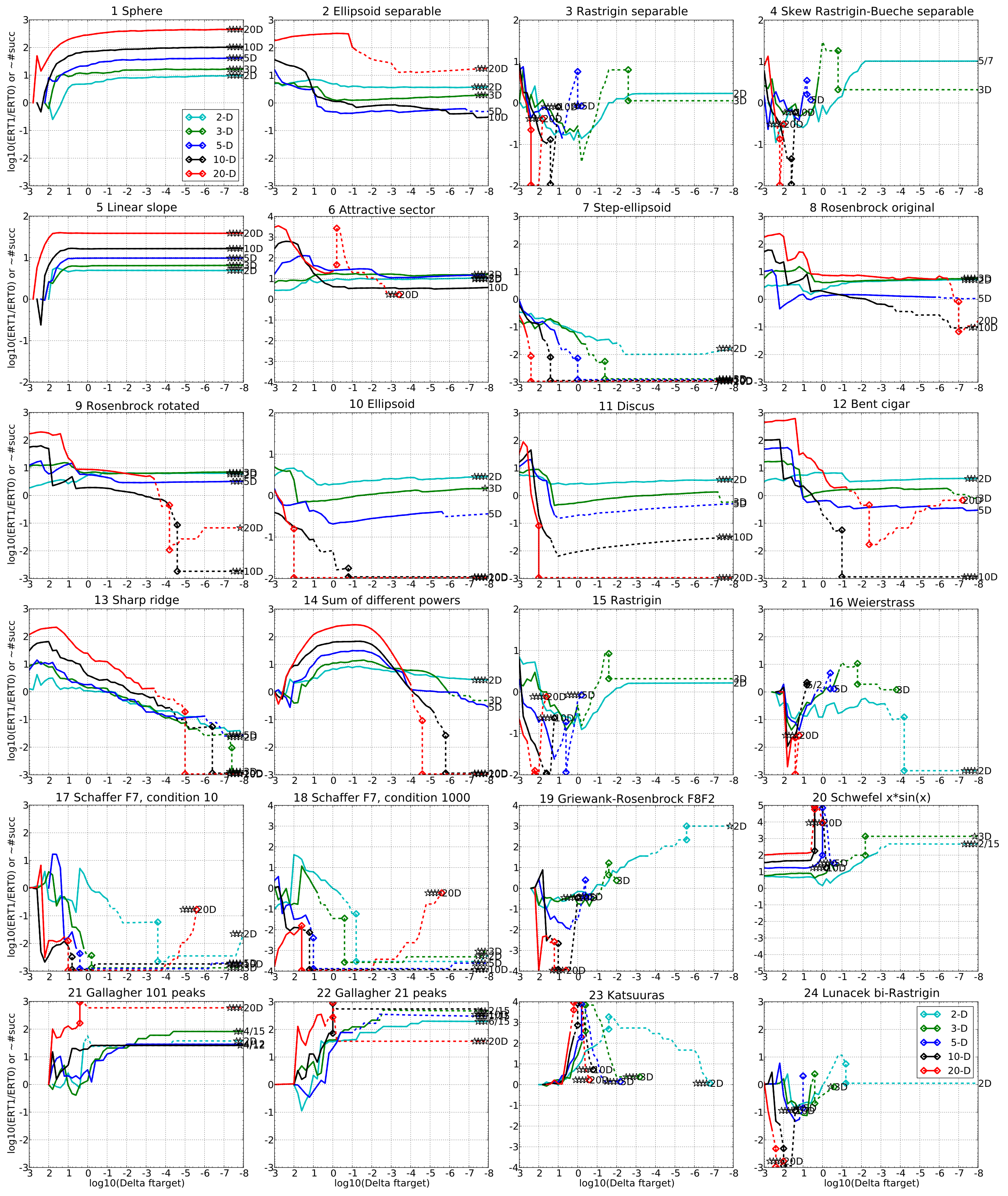

Figure 1: ERT ratio of CauchyEDA divided by Rosenbrock versus $\log_{10}(\Delta f)$ for f_1–f_{24} in 2, 3, 5, 10, 20, 40-D. Ratios $< 10^0$ indicate an advantage of CauchyEDA, smaller values are always better. The line gets dashed when for any algorithm the ERT exceeds thrice the median of the trial-wise overall number of f-evaluations for the same algorithm on this function. Symbols indicate the best achieved Δf-value of one algorithm (ERT gets undefined to the right). The dashed line continues as the fraction of successful trials of the other algorithm, where 0 means 0% and the y-axis limits mean 100%, values below zero for CauchyEDA. The line ends when no algorithm reaches Δf anymore. The number of successful trials is given, only if it was in $\{1 \ldots 9\}$ for CauchyEDA (1st number) and non-zero for Rosenbrock (2nd number). Results are significant with $p = 0.05$ for one star and $p = 10^{-\#\star}$ otherwise, with Bonferroni correction within each figure.

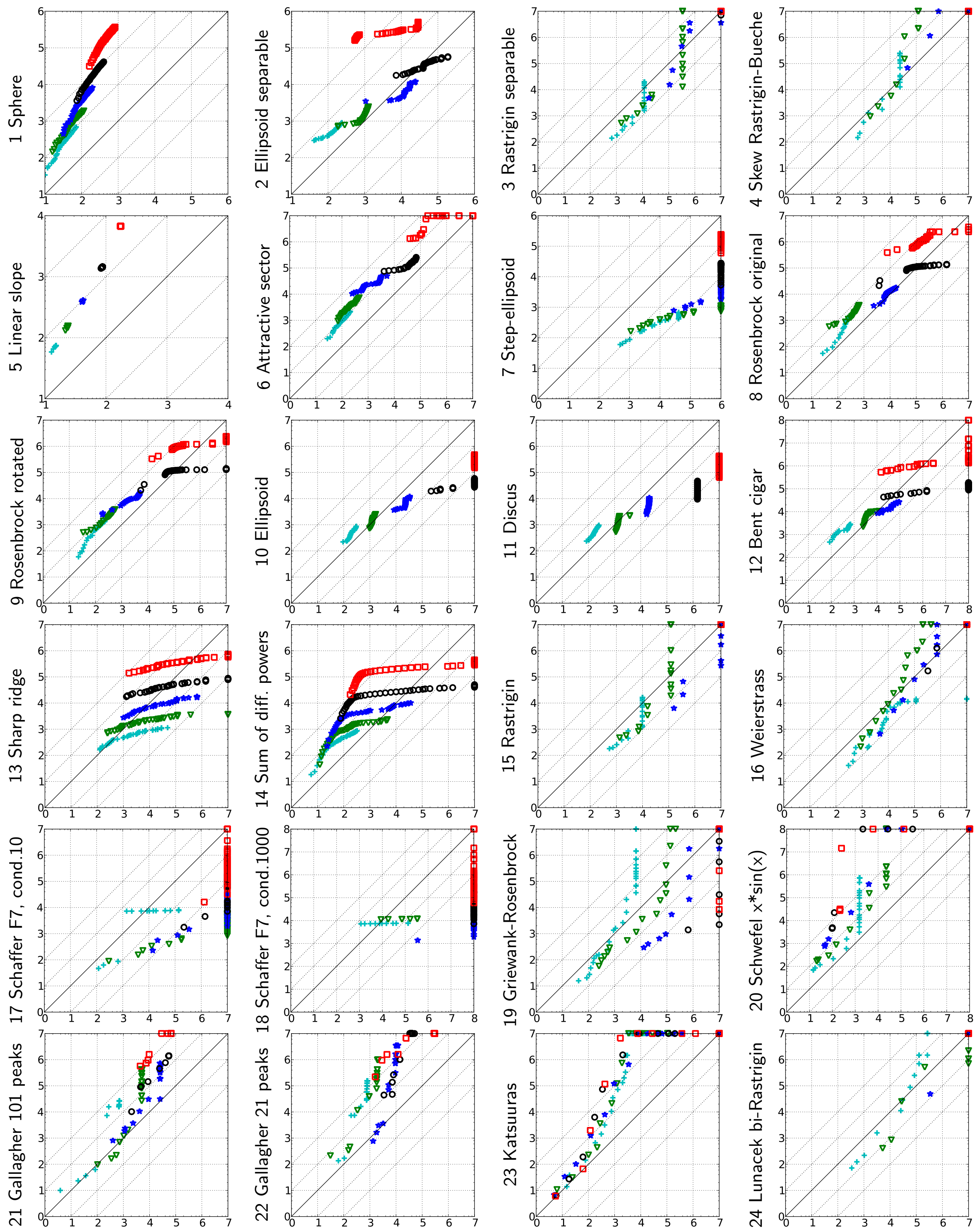

Figure 2: Expected running time (ERT in log10 of number of function evaluations) of CauchyEDA versus Rosenbrock for 46 target values $\Delta f \in [10^{-8}, 10]$ in each dimension for functions f_1–f_{24}. Markers on the upper or right egde indicate that the target value was never reached by CauchyEDA or Rosenbrock respectively. Markers represent dimension: 2:+, 3:▽, 5:⋆, 10:○, 20:□, 40:◇.

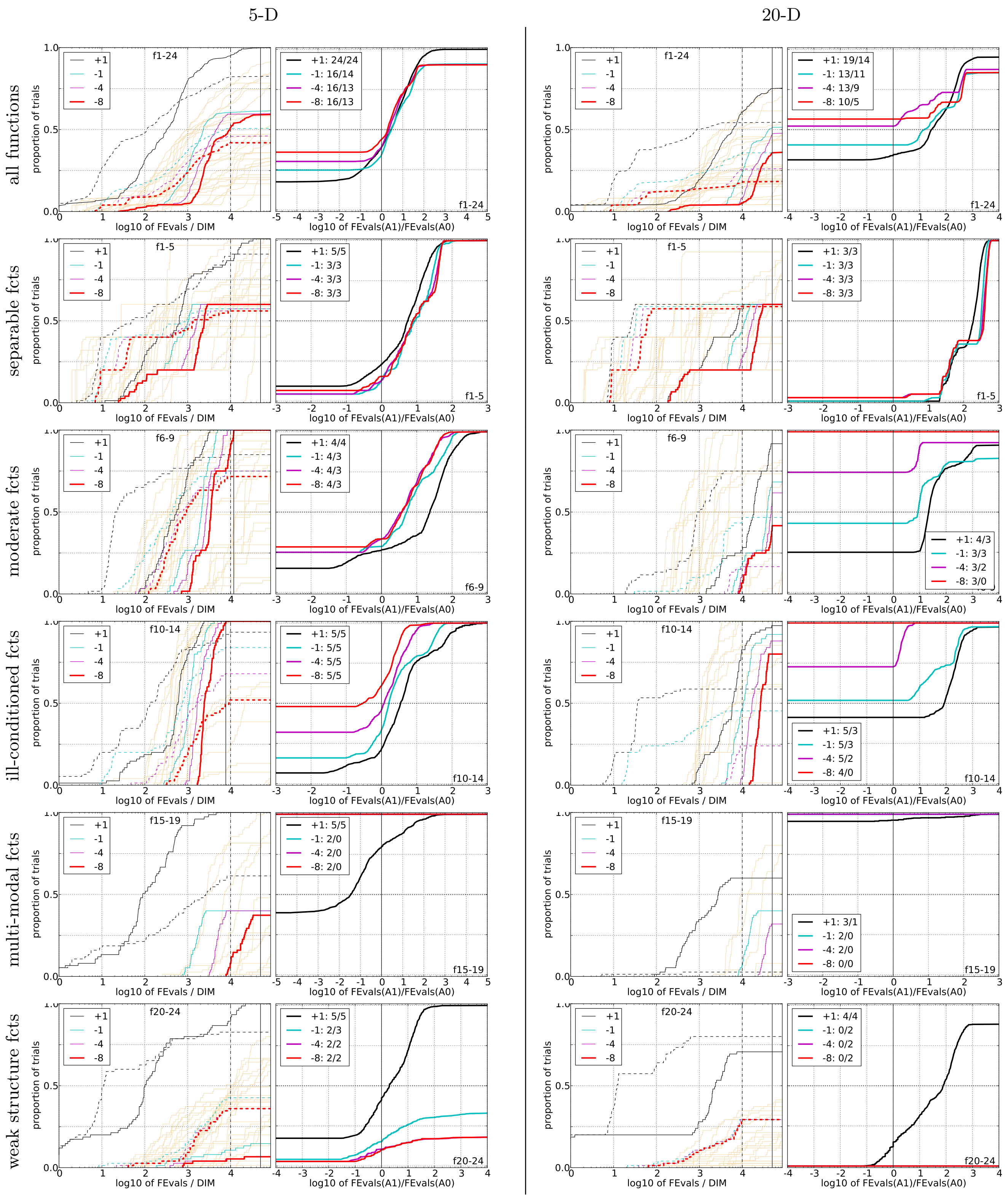

Figure 3: Empirical cumulative distributions (ECDF) of run lengths and speed-up ratios in 5-D (left) and 20-D (right). Left sub-columns: ECDF of the number of function evaluations divided by dimension D (FEvals/D) to reach a target value $f_{\text{opt}} + \Delta f$ with $\Delta f = 10^k$, where $k \in \{1, -1, -4, -8\}$ is given by the first value in the legend, for CauchyEDA (solid) and Rosenbrock (dashed). Light beige lines show the ECDF of FEvals for target value $\Delta f = 10^{-8}$ of algorithms benchmarked during BBOB-2009. Right sub-columns: ECDF of FEval ratios of CauchyEDA divided by Rosenbrock, all trial pairs for each function. Pairs where both trials failed are disregarded, pairs where one trial failed are visible in the limits being > 0 or < 1. The legends indicate the number of functions that were solved in at least one trial (CauchyEDA first).

5-D

Δf	1e+1	1e+0	1e-1	1e-3	1e-5	1e-7	#succ
f$_1$	11	12	12	12	12	12	15/15
0: Ros	**2.9**^{*3}	**4.2**^{*3}	**5.5**^{*3}	**8.7**^{*3}	**12**^{*3}	**15**^{*3}	15/15
1: Cau	41	90	170	310	460	600	15/15
f$_2$	83	87	88	90	92	94	15/15
0: Ros	**13***	100	140	150	190	240	12/15
1: Cau	42	49	58	80	100	120	15/15
f$_3$	720	1600	1600	1600	1700	1700	15/15
0: Ros	24	390	∞	∞	∞	∞ *4.6e4*	0/15
1: Cau	6.7	2.2e3	∞	∞	∞	∞ *2.5e5*	0/15
f$_4$	810	1600	1700	1800	1900	1900	15/15
0: Ros	57	∞	∞	∞	∞	∞ *5.0e4*	0/15
1: Cau	85	∞	∞	∞	∞	∞ *2.5e5*	0/15
f$_5$	10	10	10	10	10	10	15/15
0: Ros	**4**^{*3}	**4.2**^{*3}	**4.2**^{*3}	**4.2**^{*3}	**4.2**^{*3}	**4.2**^{*3}	15/15
1: Cau	39	41	41	41	41	41	15/15
f$_6$	110	210	280	580	1000	1300	15/15
0: Ros	**2.2**^{*3}	**2.8**^{*3}	**2.4**^{*3}	**4.3**^{*3}	**2.8**^{*3}	**2.4**^{*3}	15/15
1: Cau	92	69	68	47	35	34	15/15
f$_7$	24	320	1200	1600	1600	1600	15/15
0: Ros	1.2e3	670	∞	∞	∞	∞ *1.5e4*	0/15
1: Cau	**33**^{*3}	**4.9**^{*2}	**2.4**^{*3}	**2.9**^{*3}	**2.9**^{*3}	**3.4**^{*3}	15/15
f$_8$	73	270	340	390	410	420	15/15
0: Ros	**32**^{*2}	23	**22***	25	30	36	13/15
1: Cau	49	31	33	34	37	40	15/15
f$_9$	35	130	210	300	340	370	15/15
0: Ros	**5.3**^{*3}	**9.8**^{*3}	**10**^{*3}	**14**^{*2}	**14**^{*2}	**14**^{*2}	15/15
1: Cau	71	54	45	41	42	43	15/15
f$_{10}$	350	500	570	630	830	880	15/15
0: Ros	24	44	40	37	29	37	10/15
1: Cau	11	9	9.4	12	11	13	15/15
f$_{11}$	140	200	760	1200	1500	1700	15/15
0: Ros	120	88	26	18	14	13	12/15
1: Cau	18	17	6	5.3	5.6	5.9	15/15
f$_{12}$	110	270	370	460	1300	1500	15/15
0: Ros	98	63	91	95	42	48	6/15
1: Cau	79	41	35	38	17	17	15/15
f$_{13}$	130	190	250	1300	1800	2300	15/15
0: Ros	**7.6**^{*3}	**13***	26	39	63	290	1/15
1: Cau	21	24	25	7.4	**7.3***	**7.3**^{*3}	15/15
f$_{14}$	9.8	41	58	140	250	480	15/15
0: Ros	**2.4**^{*2}	**1.2**^{*3}	**1.3**^{*3}	**4.6**^{*3}	**26***	43	10/15
1: Cau	23	29	40	33	28	19	15/15
f$_{15}$	510	9300	1.9e4	2.0e4	2.1e4	2.1e4	14/15
0: Ros	310	∞	∞	∞	∞	∞ *5.0e4*	0/15
1: Cau	**12**^{*2}	**190***	∞	∞	∞	∞ *2.5e5*	0/15
f$_{16}$	120	610	2700	1.0e4	1.2e4	1.2e4	15/15
0: Ros	40	1.2e3	∞	∞	∞	∞ *5.0e4*	0/15
1: Cau	5.6	1.2e3	∞	∞	∞	∞ *2.5e5*	0/15
f$_{17}$	5.2	210	900	3700	6400	7900	15/15
0: Ros	2.7e3	∞	∞	∞	∞	∞ *5.0e4*	0/15
1: Cau	44	**13**^{*3}	**7**^{*3}	**4.3**^{*3}	**5.3**^{*3}	**13**^{*3}	14/15
f$_{18}$	100	380	4000	9300	1.1e4	1.2e4	15/15
0: Ros	3.4e3	∞	∞	∞	∞	∞ *5.0e4*	0/15
1: Cau	**13**^{*3}	**12**^{*3}	**2.4**^{*3}	**2.7**^{*3}	**3.7**^{*3}	**8.6**^{*3}	14/15
f$_{19}$	1	1	240	1.2e5	1.2e5	1.2e5	15/15
0: Ros	1.3e4	7.1e5	∞	∞	∞	∞ *5.0e4*	0/15
1: Cau	300	**2.1e4**^{*2}	∞	∞	∞	∞ *2.5e5*	0/15
f$_{20}$	16	850	3.8e4	5.4e4	5.5e4	5.5e4	14/15
0: Ros	**2.9**^{*3}	**4.6**^{*3}	∞	∞	∞	∞ *5.0e4*	0/15
1: Cau	48	460	∞	∞	∞	∞ *2.5e5*	0/15
f$_{21}$	41	1200	1700	1700	1700	1800	14/15
0: Ros	9.7	7.9	15	15	15	15	12/15
1: Cau	20	27	190	420	420	410	4/15
f$_{22}$	71	390	940	1000	1000	1100	14/15
0: Ros	19	13	**10***	**10**^{*2}	**10**^{*2}	**11**^{*2}	15/15
1: Cau	11	280	780	3.5e3	3.4e3	3.3e3	1/15
f$_{23}$	3	520	1.4e4	3.2e4	3.3e4	3.4e4	15/15
0: Ros	1.6	**1.8**^{*3}	**4.6**^{*3}	∞	∞	∞ *2.5e4*	0/15
1: Cau	2.2	230	∞	∞	∞	∞ *2.5e5*	0/15
f$_{24}$	1600	2.2e5	6.4e6	9.6e6	1.3e7	1.3e7	3/15
0: Ros	210	∞	∞	∞	∞	∞ *5.0e4*	0/15
1: Cau	30	∞	∞	∞	∞	∞ *2.5e5*	0/15

20-D

Δf	1e+1	1e+0	1e-1	1e-3	1e-5	1e-7	#succ
f$_1$	43	43	43	43	43	43	15/15
0: Ros	**3.8**^{*3}	**5.8**^{*3}	**7.2**^{*3}	**11**^{*3}	**14**^{*3}	**17**^{*3}	15/15
1: Cau	730	1.6e3	2.5e3	4.3e3	6.1e3	7.8e3	15/15
f$_2$	380	390	390	390	390	390	15/15
0: Ros	**1.4**^{*3}	**1.6**^{*3}	**5.8**^{*3}	**29**^{*3}	**73**^{*3}	**73**^{*3}	14/15
1: Cau	410	510	610	800	990	1.2e3	15/15
f$_3$	5100	7600	7600	7600	7600	7700	15/15
0: Ros	∞	∞	∞	∞	∞	∞ *1.4e5*	0/15
1: Cau	∞	∞	∞	∞	∞	∞ *1.0e6*	0/15
f$_4$	4700	7600	7700	7700	7800	1.4e5	9/15
0: Ros	∞	∞	∞	∞	∞	∞ *1.6e5*	0/15
1: Cau	∞	∞	∞	∞	∞	∞ *1.0e6*	0/15
f$_5$	41	41	41	41	41	41	15/15
0: Ros	**4.2**^{*3}	**4.3**^{*3}	**4.3**^{*3}	**4.3**^{*3}	**4.3**^{*3}	**4.3**^{*3}	15/15
1: Cau	160	170	170	170	170	170	15/15
f$_6$	1300	2300	3400	5200	6700	8400	15/15
0: Ros	**31**^{*3}	**56**^{*3}	**150**^{*3}	**570**^{*3}	∞	∞ *2.0e5*	0/15
1: Cau	1.0e3	1.3e3	∞	∞	∞	∞ *1.0e6*	0/15
f$_7$	1400	4300	9500	1.7e4	1.7e4	1.7e4	15/15
0: Ros	∞	∞	∞	∞	∞	∞ *6.7e4*	0/15
1: Cau	**44**^{*3}	**29**^{*3}	**18**^{*3}	**14**^{*3}	**14**^{*3}	**14**^{*3}	15/15
f$_8$	2000	3900	4000	4200	4400	4500	15/15
0: Ros	**3.8**^{*3}	**24**^{*3}	**28**^{*3}	**42**^{*3}	**62**^{*3}	670	0/15
1: Cau	190	180	210	260	360	540	4/15
f$_9$	1700	3100	3300	3500	3600	3700	15/15
0: Ros	**8.4**^{*3}	**31**^{*3}	**37**^{*3}	**63**^{*3}	∞	∞ *2.0e5*	0/15
1: Cau	190	270	290	310	470	630	6/15
f$_{10}$	7400	8700	1.1e4	1.5e4	1.7e4	1.7e4	15/15
0: Ros	∞	∞	∞	∞	∞	∞ *2.0e5*	0/15
1: Cau	**20**^{*3}	**22**^{*3}	**20**^{*3}	**20**^{*3}	**21**^{*3}	**25**^{*3}	15/15
f$_{11}$	1000	2200	6300	9800	1.2e4	1.5e4	15/15
0: Ros	∞	∞	∞	∞	∞	∞ *2.0e5*	0/15
1: Cau	**64**^{*3}	**44**^{*3}	**22**^{*3}	**22**^{*3}	**25**^{*3}	**26**^{*3}	15/15
f$_{12}$	1000	1900	2700	4100	1.2e4	1.4e4	15/15
0: Ros	**14**^{*3}	**56**^{*3}	**210**^{*3}	∞	∞	∞ *2.0e5*	0/15
1: Cau	510	440	420	380	390	1.1e3	0/15
f$_{13}$	650	2000	2800	1.9e4	2.4e4	3.0e4	15/15
0: Ros	**2.5**^{*3}	**4.2**^{*3}	**8.3**^{*3}	**17**^{*3}	120	∞ *2.0e5*	0/15
1: Cau	210	100	100	23	23	23	15/15
f$_{14}$	75	240	300	930	1600	1.6e4	15/15
0: Ros	**2.4**^{*3}	**1.2**^{*3}	**1.3**^{*3}	**7.4**^{*3}	∞	∞ *2.0e5*	0/15
1: Cau	280	270	350	210	180	25	15/15
f$_{15}$	3.0e4	1.5e5	3.1e5	3.2e5	4.5e5	4.6e5	15/15
0: Ros	∞	∞	∞	∞	∞	∞ *2.0e5*	0/15
1: Cau	∞	∞	∞	∞	∞	∞ *1.0e6*	0/15
f$_{16}$	1400	2.7e4	7.7e4	1.9e5	2.0e5	2.2e5	15/15
0: Ros	∞	∞	∞	∞	∞	∞ *2.0e5*	0/15
1: Cau	∞	∞	∞	∞	∞	∞ *1.0e6*	0/15
f$_{17}$	63	1000	4000	3.1e4	5.6e4	8.0e4	15/15
0: Ros	2.1e4	∞	∞	∞	∞	∞ *2.0e5*	0/15
1: Cau	**260***	**120**^{*3}	**62**^{*3}	**16**^{*3}	**23**^{*3}	∞ *1.0e6*	0/15
f$_{18}$	620	4000	2.0e4	6.8e4	1.3e5	1.5e5	15/15
0: Ros	∞	∞	∞	∞	∞	∞ *2.0e5*	0/15
1: Cau	**96**^{*3}	**42**^{*3}	**15**^{*3}	**12**^{*3}	**38**^{*3}	∞ *1.0e6*	0/15
f$_{19}$	1	1	3.4e5	6.2e6	6.7e6	6.7e6	15/15
0: Ros	∞	∞	∞	∞	∞	∞ *2.0e5*	0/15
1: Cau	**8.4e3**^{*3}	∞	∞	∞	∞	∞ *1.0e6*	0/15
f$_{20}$	82	4.6e4	3.1e6	5.5e6	5.6e6	5.6e6	14/15
0: Ros	**2.6**^{*3}	**2.9**^{*3}	∞	∞	∞	∞ *2.0e5*	0/15
1: Cau	340	∞	∞	∞	∞	∞ *1.0e6*	0/15
f$_{21}$	560	6500	1.4e4	1.5e4	1.6e4	1.8e4	15/15
0: Ros	**7.8**^{*3}	**7.6**^{*3}	**4.7**^{*3}	**4.5**^{*3}	**4.3**^{*3}	**3.8**^{*3}	14/15
1: Cau	1.0e3	∞	∞	∞	∞	∞ *1.0e6*	0/15
f$_{22}$	470	5600	2.3e4	2.5e4	2.7e4	1.3e5	12/15
0: Ros	**3.4**^{*3}	**4.3**^{*3}	**12**^{*3}	**12**^{*3}	**11**^{*3}	**2.2**^{*3}	8/15
1: Cau	470	1.2e3	∞	∞	∞	∞ *1.0e6*	0/15
f$_{23}$	3.2	1600	6.7e4	4.9e5	8.1e5	8.4e5	15/15
0: Ros	1.7	**4.6**^{*3}	∞	∞	∞	∞ *8.4e4*	0/15
1: Cau	1.9	∞	∞	∞	∞	∞ *1.0e6*	0/15
f$_{24}$	1.3e6	7.5e6	5.2e7	5.2e7	5.2e7	5.2e7	3/15
0: Ros	∞	∞	∞	∞	∞	∞ *2.0e5*	0/15
1: Cau	∞	∞	∞	∞	∞	∞ *1.0e6*	0/15

Table 1: Expected running time (ERT in number of function evaluations) divided by the best ERT measured during BBOB-2009 (given in the respective first row) for different Δf values for functions f_1–f_{24}. The median number of conducted function evaluations is additionally given in *italics*, if $\mathrm{ERT}(10^{-7}) = \infty$. #succ is the number of trials that reached the final target $f_{\mathrm{opt}} + 10^{-8}$. 0: Ros is Rosenbrock and 1: Cau is CauchyEDA. Bold entries are statistically significantly better compared to the other algorithm, with $p = 0.05$ or $p = 10^{-k}$ where $k > 1$ is the number following the $\star$ symbol, with Bonferroni correction of 48.

6. REFERENCES

[1] S. Finck, N. Hansen, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2009: Presentation of the noiseless functions. Technical Report 2009/20, Research Center PPE, 2009. Updated February 2010.

[2] N. Hansen, A. Auger, S. Finck, and R. Ros. Real-parameter black-box optimization benchmarking 2010: Experimental setup. Technical Report RR-7215, INRIA, 2010.

[3] N. Hansen, S. Finck, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions. Technical Report RR-6829, INRIA, 2009. Updated February 2010.

[4] P. Pošík. Preventing premature convergence in a simple EDA via global step size setting. In G. Rudolph, editor, *Parallel Problem Solving from Nature – PPSN X*, volume 5199 of *Lecture Notes in Computer Science*, pages 549–558. Springer, 2008.

[5] P. Pošík. BBOB-benchmarking a simple estimation-of-distribution algorithm with Cauchy distribution. In *GECCO '09: Proceedings of the 11th annual conference companion on Genetic and evolutionary computation conference*, pages 2309–2314, New York, NY, USA, 2009. ACM.

[6] P. Pošík. BBOB-benchmarking the Rosenbrock's local search algorithm. In *GECCO '09: Proceedings of the 11th annual conference companion on Genetic and evolutionary computation conference*, pages 2337–2342, New York, NY, USA, 2009. ACM.

[7] 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.

[8] H. H. Rosenbrock. An automatic method for finding the greatest or least value of a function. *The Computer Journal*, 3(3):175–184, March 1960.