Comparison of G3PCX 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

Generalized generation gap algorithm with parent centric crossover is compared with Rosenbrock's optimization algorithm. Both algorithms were already presented at the BBOB 2009 workshop where they often showed similar performance. This paper compares them in more detail and adds to the understanding of their key features and 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, Generalized generation gap, Parent-centric crossover, Rosenbrock's algorithm

1. INTRODUCTION

Black-box optimization benchmarking (BBOB) workshop aims at comparing diverse optimization algorithms in a systematic manner. The methodology created for the BBOB 2010 workshop [4] allows to compare two algorithms in greater detail. This paper presents a comparison of two algorithms benchmarked in 2009, i.e. the data are taken from the 2009 benchmarking, but the comparison is made using the new post-processing scripts and templates made available for BBOB 2010.

The two algorithms chosen for the comparison are:

- Rosenbrock's algorithm introduced in [9]. It could be described as an adaptive pattern search. Its performance on the BBOB 2009 noiseless test suite was reported in [7].
- The generalized generation gap (G3) model introduced by Deb in [1] and used with the parent centric crossover operator (PCX) introduced in [2]. It falls to the class of steady-state evolutionary algorithms. The performance of the G3PCX algorithm on the BBOB 2009 noiseless test suite was reported in [6].

Both algorithms can be described as adaptive local-search methods[1], but the neighborhood is defined in a completely different way. We can expect their behaviour to be quite similar, nevertheless it is enlighting to study the cases where the performance of both methods differs.

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 discussion. 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 [7], respectively. Both algorithms search in the neighborhood of the best solution found so far and both algorithms work in an incremental manner, i.e. they evaluate a small number of candidate solutions each iteration.

The main differences between the algorithms are:

- Rosenbrock's algorithm is a single-point method, i.e. it iteratively updates a single point which represents the best solution found so far. G3PCX is a population-based algorithm, i.e. it maintains a whole set of candidate solutions.
- Rosenbrock's algorithm maintains a model of the local neigborhood (orthonormal basis). The updates are carried out only in strictly defined situations; the algorithm uses the same model for a certain time period and these periods may differ in length. The G3PCX algorithm uses the population as a mean to adapt to the local neigborhood. It is adapted continuously, every generation.

[1]Note that the local-search behaviour is not generally a feature of the G3PCX algorithm. Only the particular implementation of this algorithm used in this comparison is such local-search heavy.

GECCO'10, July 7–11, 2010, Portland, Oregon, USA.

- The neighborhood used in Rosenbrock's algorithm has the form of a pattern. The algorithm iterates over the individual axes of the orthonormal basis and samples a new candidate in a precisely defined distance in the direction of the respective axis. The G3PCX algorithm uses Gaussian distribution to sample new point. The parameters of the Gaussian are estimated from several (here 3) selected parents.

For both algorithms, the crafting effort CrE= 0.

3. RESULTS

Results from experiments according to [4] on the benchmark functions given in [3, 5] 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, $f_t = f_{opt} + \Delta f$, 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 [4, 8]. **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.

Although the algorithms use different principles, both can be described as local search methods. Their performance is often similar. Rosenbrock's algorithm ouperforms G3PCX on functions 1, 2, 5, while G3PCX beats Rosenbrock on functions 7, 16, 17, 18, i.e. in this small competition the G3PCX wins 4:3. The results on the other functions are mixed, or neither algorithm solved the problem successfully. Generally, we can say that for dimensions 2 and 3, Rosenbrock's algorithm is faster, while in higher dimensions Rosenbrock's algorithm often does not find good enough solution and G3PCX wins.

In Fig. 1, we can sometimes see a downward peak at the beginning of the ERT ratio lines (funtions 1, 5, etc.). The peak means that the model used by Rosenbrock's algorithm must first be adapted to the local neighborhood of the fitness landscape. G3PCX algorithm does not use any explicit model of the neighborhood and thus is able to improve the best-so-far solution almost instantly. After Rosenbrock's algorithm learns its model, it is often faster than G3PCX.

On functions 7 (Step-ellipsoid) and on both Schaffer's functions 17 and 18 neither algorithm works well, but despite that the results of G3PCX are better. The reason for this seems to be the population used by G3PCX.

Gallagher's functions 21 and 22 are known to be solvable by performing sufficient number of restarts. Both algorithms need similar number of function evaluations for target levels 10^{-1} and tighter. This suggests that both algorithms converge sufficiently quickly to perform the needed number of restarts.

Looking at Fig. 3, it can be stated that for 5D functions, the proportion of successful trials is similar for both algorithms, with Rosenbrock's algorithm being usually faster. With increasing search-space dimensionality, the adaptation mechanism of Rosenbrock's algorithm loses its efficiency, and G3PCX with its population approach is able to solve larger proportion of functions, and many of them also faster. Rosenbrock's algorithm beats G3PCX mainly on the separable functions (where G3PCX exhibits slightly more successful trials, but Rosebrock's algorithm is faster). G3PCX wins on moderate functions and ill-conditioned functions in 20D. On weak-structure functions both algorithms did comparably well, while both algorithms failed for multi-modal functions.

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
G3PCX	470	443	420	411	540	750

4. CPU TIMING EXPERIMENTS

The time requirements of both algorithms are taken from the respective articles, [6] and [7]. 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 time demands of both algorithms are similar. Both work in an incremental manner, evaluating candidate solution sets of similar sizes with similar frequency.

5. CONCLUSIONS

The results confirm that both algorithms behave like local-search methods—they completely fail on functions assigned to the "multimodal" group. For low-dimensional functions, Rosenbrock's algorithm is usually faster. In high dimensional spaces, however, the improved global search abilities of G3PCX show up and the algorithm is able to solve higher proportion of functions.

Acknowledgements

The project was supported by the Ministry of Education, Youth and Sport of the Czech Republic with the grant No. MSM6840770012 entitled "Transdisciplinary Research in Biomedical Engineering II".

6. REFERENCES

[1] K. Deb. A population-based algorithm-generator for real-parameter optimization. *Soft Computing*, 9(4):236–253, April 2005.

[2] K. Deb, A. Anand, and D. Joshi. A computationally efficient evolutionary algorithm for real-parameter optimization. Technical report, Indian Institute of Technology, April 2002.

[3] 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.

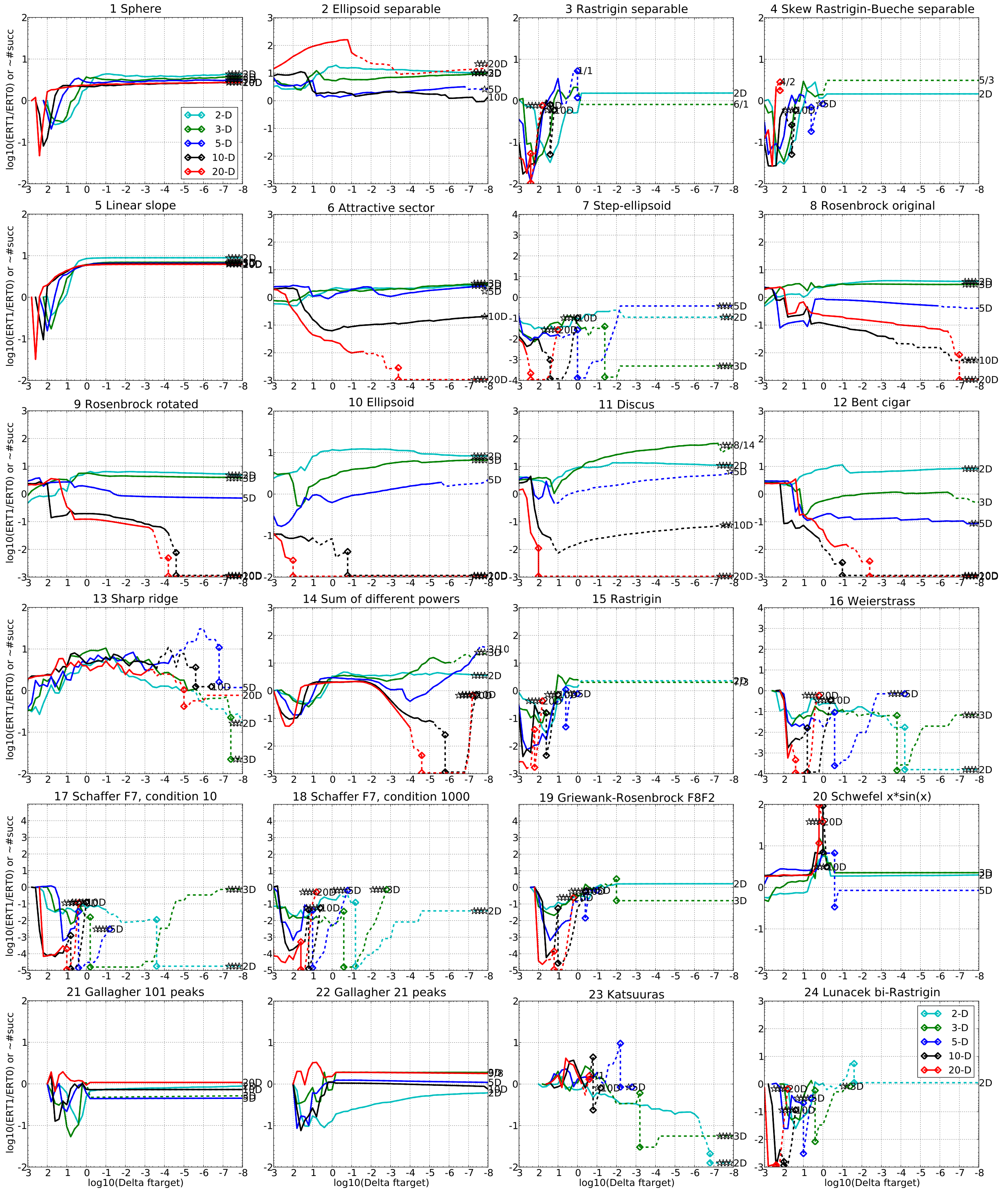

Figure 1: ERT ratio of G3PCX divided by Rosenbrock versus $\log_{10}(\Delta f)$ for f_1–f_{24} in 2, 3, 5, 10, 20, 40-D. Ratios $< 10^0$ indicate an advantage of G3PCX, smaller values are always better. The line gets dashed when for any algorithm the ERT exceeds thrice the median of the trial-wise overall 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 G3PCX. 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 G3PCX (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.

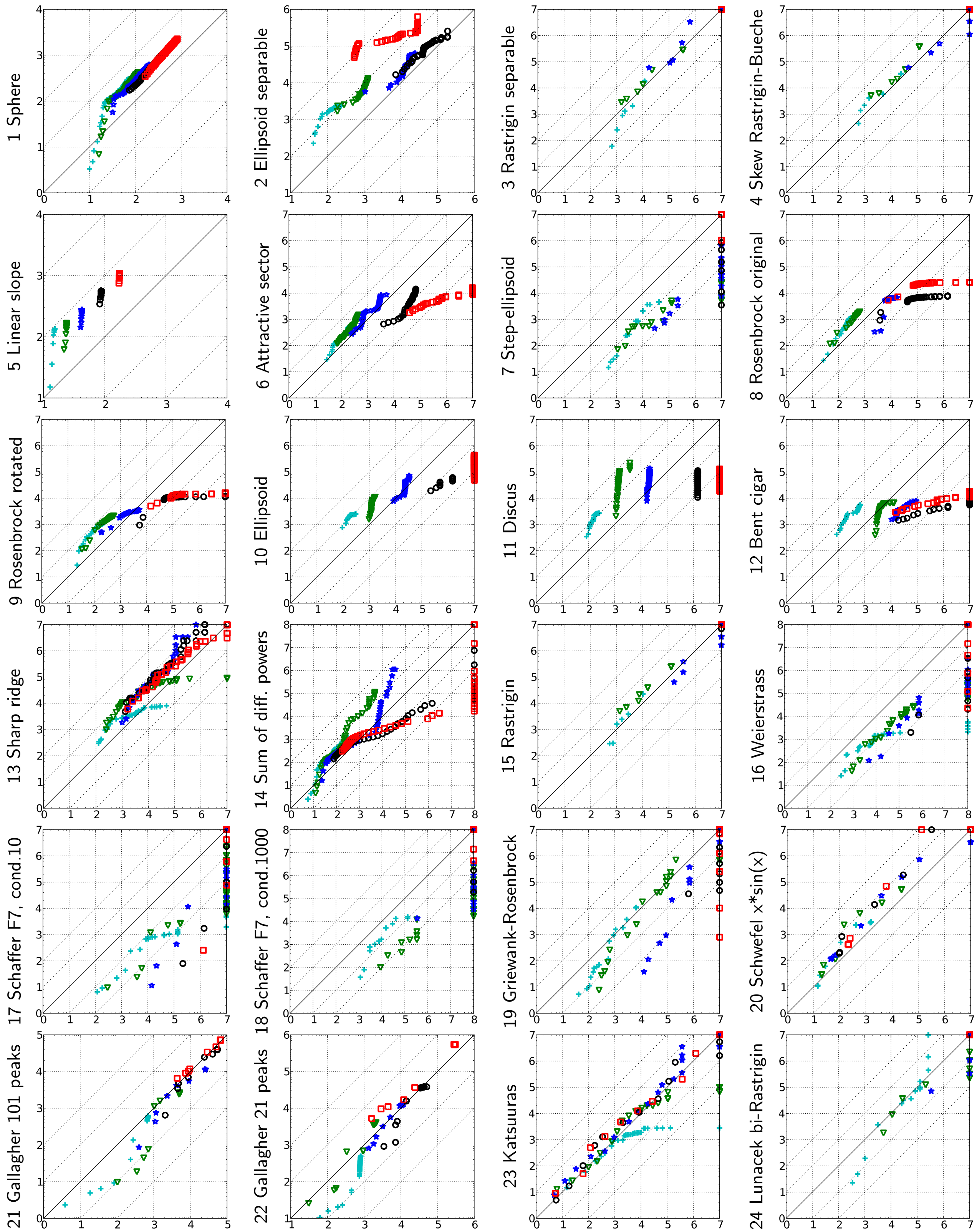

Figure 2: Expected running time (ERT in log10 of number of function evaluations) of G3PCX versus Rosenbrock for 46 target values $\Delta f \in [10^{-8}, 10]$ in each dimension for functions f_1–f_{24}. Markers on the upper or right egde indicate that the target value was never reached by G3PCX or Rosenbrock respectively. Markers represent dimension: 2:+, 3:▽, 5:⋆, 10:○, 20:□, 40:◇.

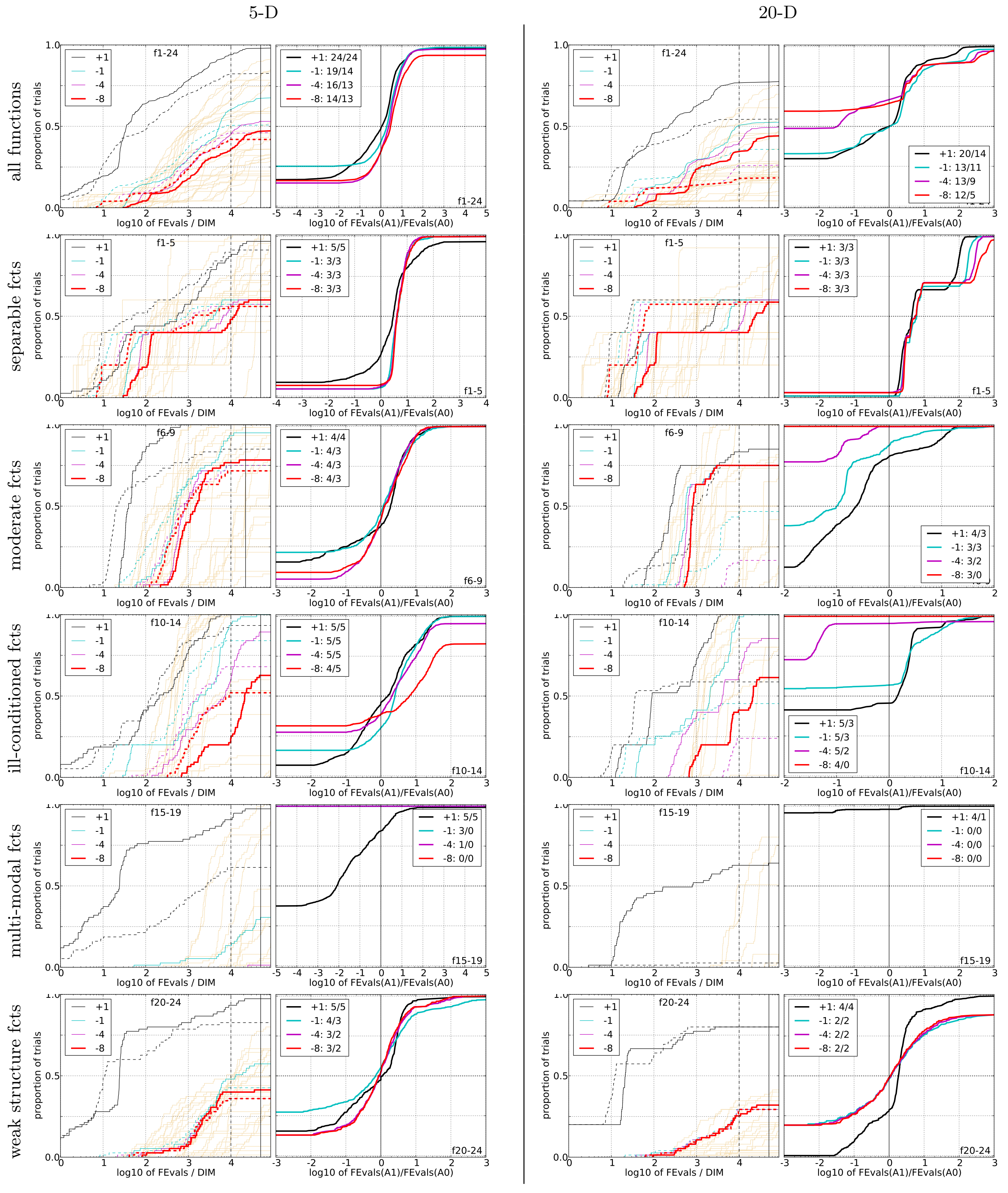

Figure 3: Empirical cumulative distributions (ECDF) of run lengths and speed-up ratios in 5-D (left) and 20-D (right). Left sub-columns: ECDF of the number of function evaluations divided by dimension D (FEvals/D) to reach a target value $f_{\rm opt} + \Delta f$ with $\Delta f = 10^k$, where $k \in \{1, -1, -4, -8\}$ is given by the first value in the legend, for G3PCX (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 G3PCX 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 (G3PCX first).

5-D

Δf	1e+1	1e+0	1e-1	1e-3	1e-5	1e-7	#succ
f1	11	12	12	12	12	12	15/15
0: Ros	2.9	**4.2**$^{\star 3}$	**5.5**$^{\star 3}$	**8.7**$^{\star 3}$	**12**$^{\star 3}$	**15**$^{\star 3}$	15/15
1: G3P	5.2	12	15	25	35	45	15/15
f2	83	87	88	90	92	94	15/15
0: Ros	**13**$^{\star 2}$	100	140	150	190	240	12/15
1: G3P	69	150	220	340	470	620	15/15
f3	720	1600	1600	1600	1700	1700	15/15
0: Ros	24	390	∞	∞	∞	∞ *4.6e4*	0/15
1: G3P	84	2.1e3	∞	∞	∞	∞ *2.5e5*	0/15
f4	810	1600	1700	1800	1900	1900	15/15
0: Ros	57	∞	∞	∞	∞	∞ *5.0e4*	0/15
1: G3P	76	2.2e3	∞	∞	∞	∞ *2.5e5*	0/15
f5	10	10	10	10	10	10	15/15
0: Ros	**4**$^{\star 3}$	**4.2**$^{\star 3}$	**4.2**$^{\star 3}$	**4.2**$^{\star 3}$	**4.2**$^{\star 3}$	**4.2**$^{\star 3}$	15/15
1: G3P	14	25	27	28	28	28	15/15
f6	110	210	280	580	1000	1300	15/15
0: Ros	2.2	2.8	2.4	4.3	**2.8***	**2.4***	15/15
1: G3P	2.4	3	4.8	4.7	5.1	5.5	15/15
f7	24	320	1200	1600	1600	1600	15/15
0: Ros	1.2e3	670	∞	∞	∞	∞ *1.5e4*	0/15
1: G3P	**19**$^{\star 3}$	**18**$^{\star 2}$	**45**$^{\star 3}$	**410**$^{\star 3}$	**410**$^{\star 3}$	**410**$^{\star 3}$	2/15
f8	73	270	340	390	410	420	15/15
0: Ros	32	23	22	25	30	36	13/15
1: G3P	4.6	20	18	17	16	16	15/15
f9	35	130	210	300	340	370	15/15
0: Ros	**5.3***	9.8	10	14	14	14	15/15
1: G3P	14	18	14	11	10	9.8	15/15
f10	350	500	570	630	830	880	15/15
0: Ros	24	44	40	37	29	37	10/15
1: G3P	22	27	36	49	51	64	15/15
f11	140	200	760	1200	1500	1700	15/15
0: Ros	120	88	26	18	14	13	12/15
1: G3P	55	110	45	49	56	61	14/15
f12	110	270	370	460	1300	1500	15/15
0: Ros	98	63	91	95	42	48	6/15
1: G3P	14	11	13	12	5.2	**5.1***	15/15
f13	130	190	250	1300	1800	2300	15/15
0: Ros	7.6	13	26	39	63	290	1/15
1: G3P	14	60	150	120	590	∞ *2.3e5*	0/15
f14	9.8	41	58	140	250	480	15/15
0: Ros	2.4	**1.2**$^{\star 3}$	**1.3**$^{\star 3}$	4.6	26	**43***	10/15
1: G3P	1.7	3.5	3.5	5	26	390	3/15
f15	510	9300	1.9e4	2.0e4	2.1e4	2.1e4	14/15
0: Ros	310	∞	∞	∞	∞	∞ *5.0e4*	0/15
1: G3P	130	370	∞	∞	∞	∞ *2.5e5*	0/15
f16	120	610	2700	1.0e4	1.2e4	1.2e4	15/15
0: Ros	40	1.2e3	∞	∞	∞	∞ *5.0e4*	0/15
1: G3P	**1***	**22**$^{\star 3}$	**44**$^{\star 3}$	**350**$^{\star 3}$	∞	∞ *2.5e5*	0/15
f17	5.2	210	900	3700	6400	7900	15/15
0: Ros	2.7e3	∞	∞	∞	∞	∞ *5.0e4*	0/15
1: G3P	2.2	**130**$^{\star 3}$	**290**$^{\star 3}$	∞	∞	∞ *2.5e5*	0/15
f18	100	380	4000	9300	1.1e4	1.2e4	15/15
0: Ros	3.4e3	∞	∞	∞	∞	∞ *5.0e4*	0/15
1: G3P	**130**$^{\star 2}$	**800**$^{\star 3}$	∞	∞	∞	∞ *2.5e5*	0/15
f19	1	1	240	1.2e5	1.2e5	1.2e5	15/15
0: Ros	1.3e4	7.1e5	∞	∞	∞	∞ *5.0e4*	0/15
1: G3P	39	**9.5e4**$^{\star 2}$	**1.4e4**$^{\star 2}$	∞	∞	∞ *2.5e5*	0/15
f20	16	850	3.8e4	5.4e4	5.5e4	5.5e4	14/15
0: Ros	**2.9**$^{\star 2}$	**4.6***	∞	∞	∞	∞ *5.0e4*	0/15
1: G3P	7.4	36	88	62	61	61	1/15
f21	41	1200	1700	1700	1700	1800	14/15
0: Ros	9.7	7.9	15	15	15	15	12/15
1: G3P	2.1	4.7	6.8	6.7	6.7	6.6	15/15
f22	71	390	940	1000	1000	1100	14/15
0: Ros	19	13	10	10	10	11	15/15
1: G3P	12	15	13	12	12	12	15/15
f23	3	520	1.4e4	3.2e4	3.3e4	3.4e4	15/15
0: Ros	1.6	1.8	4.6	∞	∞	∞ *2.5e4*	0/15
1: G3P	2.6	2.4	8.6	∞	∞	∞ *2.5e5*	0/15
f24	1600	2.2e5	6.4e6	9.6e6	1.3e7	1.3e7	3/15
0: Ros	210	∞	∞	∞	∞	∞ *5.0e4*	0/15
1: G3P	44	∞	∞	∞	∞	∞ *2.5e5*	0/15

20-D

Δf	1e+1	1e+0	1e-1	1e-3	1e-5	1e-7	#succ
f1	43	43	43	43	43	43	15/15
0: Ros	**3.8**$^{\star 3}$	**5.8**$^{\star 3}$	**7.2**$^{\star 3}$	**11**$^{\star 3}$	**14**$^{\star 3}$	**17**$^{\star 3}$	15/15
1: G3P	8	13	18	27	37	48	15/15
f2	380	390	390	390	390	390	15/15
0: Ros	**1.4**$^{\star 3}$	**1.6**$^{\star 3}$	**5.8**$^{\star 3}$	**29**$^{\star 3}$	**73**$^{\star 3}$	**73**$^{\star 3}$	14/15
1: G3P	130	210	320	550	760	990	14/15
f3	5100	7600	7600	7600	7600	7700	15/15
0: Ros	∞	∞	∞	∞	∞	∞ *1.4e5*	0/15
1: G3P	∞	∞	∞	∞	∞	∞ *1.0e6*	0/15
f4	4700	7600	7700	7700	7800	1.4e5	9/15
0: Ros	∞	∞	∞	∞	∞	∞ *1.6e5*	0/15
1: G3P	∞	∞	∞	∞	∞	∞ *1.0e6*	0/15
f5	41	41	41	41	41	41	15/15
0: Ros	**4.2**$^{\star 3}$	**4.3**$^{\star 3}$	**4.3**$^{\star 3}$	**4.3**$^{\star 3}$	**4.3**$^{\star 3}$	**4.3**$^{\star 3}$	15/15
1: G3P	19	25	26	27	27	27	15/15
f6	1300	2300	3400	5200	6700	8400	15/15
0: Ros	31	56	150	570	∞	∞ *2.0e5*	0/15
1: G3P	**1.4**$^{\star 3}$	**1.5**$^{\star 3}$	**1.5**$^{\star 3}$	**1.5**$^{\star 3}$	**1.7**$^{\star 3}$	**1.7**$^{\star 3}$	15/15
f7	1400	4300	9500	1.7e4	1.7e4	1.7e4	15/15
0: Ros	∞	∞	∞	∞	∞	∞ *6.7e4*	0/15
1: G3P	**760**$^{\star 3}$	∞	∞	∞	∞	∞ *2.7e5*	0/15
f8	2000	3900	4000	4200	4400	4500	15/15
0: Ros	3.8	24	28	42	62	670	0/15
1: G3P	2.6	5.4	5.5	**5.5**$^{\star 3}$	**5.5**$^{\star 3}$	**5.6**$^{\star 3}$	15/15
f9	1700	3100	3300	3500	3600	3700	15/15
0: Ros	8.4	31	37	63	∞	∞ *2.0e5*	0/15
1: G3P	**2.9**$^{\star 2}$	**3.8**$^{\star 3}$	**4**$^{\star 3}$	**4.1**$^{\star 3}$	**4.1**$^{\star 3}$	**4.2**$^{\star 3}$	15/15
f10	7400	8700	1.1e4	1.5e4	1.7e4	1.7e4	15/15
0: Ros	∞	∞	∞	∞	∞	∞ *2.0e5*	0/15
1: G3P	**6.5**$^{\star 3}$	**10**$^{\star 3}$	**12**$^{\star 3}$	**15**$^{\star 3}$	**18**$^{\star 3}$	**23**$^{\star 3}$	15/15
f11	1000	2200	6300	9800	1.2e4	1.5e4	15/15
0: Ros	∞	∞	∞	∞	∞	∞ *2.0e5*	0/15
1: G3P	**18**$^{\star 3}$	**14**$^{\star 3}$	**7**$^{\star 3}$	**7.1**$^{\star 3}$	**7.6**$^{\star 3}$	**8**$^{\star 3}$	15/15
f12	1000	1900	2700	4100	1.2e4	1.4e4	15/15
0: Ros	14	56	210	∞	∞	∞ *2.0e5*	0/15
1: G3P	2.7	2.8	3	**2.9**$^{\star 3}$	**1.2**$^{\star 3}$	**1.3**$^{\star 3}$	15/15
f13	650	2000	2800	1.9e4	2.4e4	3.0e4	15/15
0: Ros	2.5	4.2	**8.3**$^{\star 2}$	17	120	∞ *2.0e5*	0/15
1: G3P	9.3	17	43	47	130	330	1/15
f14	75	240	300	930	1600	1.6e4	15/15
0: Ros	**2.4**$^{\star 3}$	**1.2**$^{\star 3}$	**1.3**$^{\star 3}$	7.4	∞	∞ *2.0e5*	0/15
1: G3P	4.1	2.4	2.8	**2.7**$^{\star 3}$	**13**$^{\star 3}$	**59**$^{\star 3}$	0/15
f15	3.0e4	1.5e5	3.1e5	3.2e5	4.5e5	4.6e5	15/15
0: Ros	∞	∞	∞	∞	∞	∞ *2.0e5*	0/15
1: G3P	∞	∞	∞	∞	∞	∞ *1.0e6*	0/15
f16	1400	2.7e4	7.7e4	1.9e5	2.0e5	2.2e5	15/15
0: Ros	∞	∞	∞	∞	∞	∞ *2.0e5*	0/15
1: G3P	**17**$^{\star 3}$	∞	∞	∞	∞	∞ *1.0e6*	0/15
f17	63	1000	4000	3.1e4	5.6e4	8.0e4	15/15
0: Ros	2.1e4	∞	∞	∞	∞	∞ *2.0e5*	0/15
1: G3P	**4**$^{\star 2}$	∞	∞	∞	∞	∞ *1.0e6*	0/15
f18	620	4000	2.0e4	6.8e4	1.3e5	1.5e5	15/15
0: Ros	∞	∞	∞	∞	∞	∞ *2.0e5*	0/15
1: G3P	**7.1e3**$^{\star 3}$	∞	∞	∞	∞	∞ *1.0e6*	0/15
f19	1	1	3.4e5	6.2e6	6.7e6	6.7e6	15/15
0: Ros	∞	∞	∞	∞	∞	∞ *2.0e5*	0/15
1: G3P	**800**$^{\star 3}$	∞	∞	∞	∞	∞ *1.0e6*	0/15
f20	82	4.6e4	3.1e6	5.5e6	5.6e6	5.6e6	14/15
0: Ros	**2.6**$^{\star 3}$	**2.9**$^{\star 3}$	∞	∞	∞	∞ *2.0e5*	0/15
1: G3P	5	∞	∞	∞	∞	∞ *1.0e6*	0/15
f21	560	6500	1.4e4	1.5e4	1.6e4	1.8e4	15/15
0: Ros	7.8	7.6	4.7	4.5	4.3	3.8	14/15
1: G3P	12	7.2	5.1	4.9	4.6	4.1	15/15
f22	470	5600	2.3e4	2.5e4	2.7e4	1.3e5	12/15
0: Ros	3.4	4.3	12	12	11	2.2	8/15
1: G3P	11	6.6	23	22	20	4	9/15
f23	3.2	1600	6.7e4	4.9e5	8.1e5	8.4e5	15/15
0: Ros	1.7	4.6	∞	∞	∞	∞ *8.4e4*	0/15
1: G3P	2.8	7.8	∞	∞	∞	∞ *5.7e5*	0/15
f24	1.3e6	7.5e6	5.2e7	5.2e7	5.2e7	5.2e7	3/15
0: Ros	∞	∞	∞	∞	∞	∞ *2.0e5*	0/15
1: G3P	∞	∞	∞	∞	∞	∞ *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: G3P is G3PCX. 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.

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

[5] 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.

[6] P. Pošík. BBOB-benchmarking the generalized generation gap model with parent centric crossover. In *GECCO '09: Proceedings of the 11th annual conference companion on Genetic and evolutionary computation conference*, pages 2321–2328, New York, NY, USA, 2009. ACM.

[7] 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.

[8] 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.

[9] 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.