Black-Box Optimization Benchmarking of Two Variants of the POEMS Algorithm on the Noiseless Testbed

Jiří Kubalík
Department of Cybernetics
CTU Prague
Technická 2, 166 27 Prague 6
Czech Republic
kubalik@labe.felk.cvut.cz

ABSTRACT

This paper presents benchmarking of a stochastic local search algorithm called Prototype Optimization with Evolved Improvement Steps (POEMS) on the BBOB 2010 noise-free functions testbed. An original version of the POEMS algorithm presented at BBOB 2009 workshop is compared to a new variant using a pool of candidate prototypes. Experiments for 2D, 3D, 5D, 10D and 20D were done. Experimental results show that the new variant of POEMS performs better on several functions for lower dimensions. Both variants perform equally on the 20D problems.

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, Evolutionary algorithms

1. INTRODUCTION

Prototype Optimization with Evolved Improvement Steps (POEMS) optimization algorithm is a stochastic local search algorithm that uses an evolutionary algorithm for searching the neighborhood of the current best solution. The moves in the search space can be thought of as so-called *evolved hypermutations*. The concept of the evolved hypermutations has been shown to outperform other mutation-based evolutionary algorithms that use pure random hypermutations for generating new points in the search space on several combinatorial optimization problems [5, 6, 7].

This paper compares two variants of the POEMS on the BBOB 2010 noiseless functions testbed. The two variants of POEMS are:

- The original version of the POEMS algorithm (denoted as oPOEMS) that has already been tested on the BBOB 2009 noise-free functions testbed [4].
- An extended version of the POEMS algorithm that uses a pool of candidate prototypes, denoted pPOEMS.

Series of experiments were carried out on the noise-free functions for 2, 3, 5 and 10D. The comparison is made using the new BBOB 2010 post-processing scripts and templates.

In the next section, both POEMS variants are shortly described along with the parameter setting used in the presented experiments. Section 3 presents all the results used to compare the algorithms and their discussions. Time demands of of both compared algorithms are presented in Section 4. Section 5 concludes the paper.

2. POEMS

2.1 Original POEMS

Original version of POEMS is described in [4]. It uses hypermutations composed of actions of one type denoted as $changeVariable(i, value)$. This action changes the value of the current prototype's variable i by adding the value $value$. The parameter $value$ can be positive or negative number sampled from the normal distribution $N(0, \sigma_i^2)$. Moreover, the $value$ is always chosen so that the constraint

$$lbound \leq prototype[i] + value \leq ubound$$

is satisfied.

The parameters σ_i^2 are initialized to

$$\sigma_i^2 = 0.25 * (ubound - lbound)$$

at the beginning of the POEMS run.

During the course of the run the values of σ_i^2 are adapted between iteration $k-1$ and iteration k according to the following rule

$$\sigma_{i,k}^2 = \sigma_{i,k-1}^2 * (1 - \alpha) + \delta_i * \alpha, \tag{1}$$

where

$$\delta_i = prototype[i]^{(k)} - prototype[i]^{(k-1)} \tag{2}$$

and α is a weighting factor that takes values from the interval $(0, 1)$. Thus, if the prototype's variable i does not change from iteration $k-1$ to iteration k then the corresponding $\sigma^2_{i,k}$ decreases to maximally possible extent. In the opposite case, the $\sigma^2_{i,k}$ is decreased less or it can even increase.

This can be interpreted so that if for the given value of σ^2_i an improving action sequence that includes a modification of the variable i has been found then there is perhaps no need for decreasing a value of σ^2_i. On the contrary, an absence of an action modifying the variable i in the improving action sequence or if no improving action sequence has been found in the current iteration can indicate that the interval determined by σ^2_i is too wide. Thus, the search should focus to a closer neighborhood of the current prototype's value of the variable i.

Restarted strategy. The algorithm stops either when the maximum number of function evaluations has been exceeded or when a solution of a quality equal to or better than the target function value $f_{target} = f_{opt} + 10^{-8}$ has been found. Additionally, if the values of σ^2_i for $i = 1 \dots D$ fall below 10^{-11} then they are reinitialized to the original values $0.25 * (ubound - lbound)$ while **the current prototype remains unchanged**.

2.2 POEMS with Pool of Prototypes

The second variant, denoted as pPOEMS, uses a pool of candidate prototypes of size M from which one prototype is chosen in each iteration. Each candidate prototype maintains its own σ^2_i values. Thus, the size of the neighborhood to be searched is different for each candidate prototype. The best modification of the current prototype is sought by an evolutionary algorithm and the resulting solution replaces one of the candidate prototypes in the pool according to the following rules:

1. If the modified prototype is better than the current prototype then the modified prototype replaces the current prototype in the pool of prototypes (option A in Figure 1). The values of σ^2_i of the current prototype are adapted according to Eq. (1) based on the differences between the modified modified and the current prototype variable values. Finally, this prototype remains the current prototype for the next iteration.

2. If the modified prototype is equally good as the current prototype then it replaces the current prototype in the pool of prototypes (option B in Figure 1) and the values of σ^2_i of the current prototype are adapted in the same way as in the rule nb. 1. However, since no improvement to the current prototype has been achieved in this iteration the next prototype from the pool of prototypes (meaning the prototype with index $(i+1)\%M$, where i is the index of the current prototype) becomes the current prototype for the next iteration.

3. If the modified prototype is worse than the current prototype then the most similar (according to the Euclidean distance) candidate prototype out of the prototypes that has worse fitness than the modified prototype is sought in the pool of prototypes. If such a prototype exists then it is replaced (option C in Figure 1) by the modified prototype. The values of σ^2_i of the replacement prototype are adapted according to Eq. (1) based on the differences between the modified and the replacement prototype variable values.

   If such a replacement does not exist then the modified prototype is thrown away and the values of σ^2_i of the current prototype are adapted so that the deltas $\delta_i = 0$ are used. Thus, the values of σ^2_i are maximally decreased.

   In both cases, next prototype from the pool of prototypes becomes the current prototype for the next iteration.

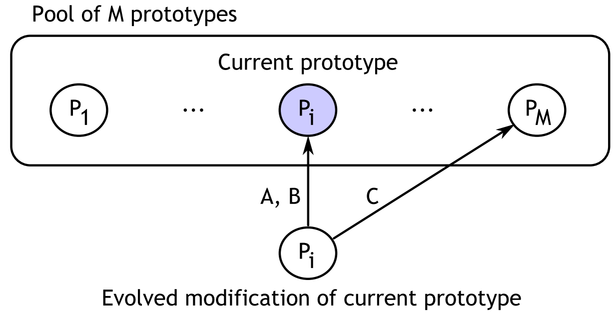

Figure 1: Pool of prototypes used in pPOEMS.

In all cases 1–3, if for some candidate prototype all its σ^2_i values drop below 10^{-11} then they are reinitialized to $0.25 * (ubound - lbound)$. The prototype itself remains unchanged.

2.3 Experimental Setup

No tuning of POEMS control parameters was done. The configuration was parameterized solely by the dimension of the current problem. The parameter setting was identical for all functions so the crafting effort $CrE = 0$. Both of the POEMS algorithms were configured as follows:

- $MaxGenes = D$, $NicheSize = 20$,
- $PopSize = MaxGenes * NicheSize$,
- $P_{cross} = 0.75$, $P_{mutate} = 0.25$, $\alpha = 0.2$,
- Tournament selection with $n = 2$,
- $lbound = -5.0$, $ubound = 5.0$,
- Number of fitness evaluations calculated in each iteration: $10 * PopSize$,
- Maximal number of fitness evaluations: $D \times 3 \cdot 10^5$,
- Pool size M: 7.

The simulations for 2, 3, 5, 10 and 20D were done with a maximum of $D \times 3 \cdot 10^5$ function evaluations.

3. RESULTS

Results from experiments according to [2] on the benchmark functions given in [1, 3] are presented in Figures 2 and 4 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 [2, 8]. **Statistical significance** is tested with the rank-sum test for a given target Δf_t (10^{-8} in Figure 2) 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.

In general, the POEMS algorithm successfully solves the separable functions while it has difficulties with solving ill-conditioned, multi-modal and weak structure functions, especially in 20D.

Figure 4 shows that in lower dimensions the pPOEMS algorithm outperforms the oPOEMS on ill-conditioned functions, multi-modal functions, and weak structure function. On separable functions and moderate functions no significant differences between the algorithms can be observed. In higher dimensions the pPOEMS outperforms oPOEMS.

Looking at Table 1, it can be stated that in 5D the pPOEMS outperforms oPOEMS on functions 10, 12, 18, 21, and 22 with respect to #succ. In 20D, there are no significant differences between the two algorithms with respect to #succ. There the pPOEMS finds orders of magnitude better results on functions 7 and 9 than oPOEMS. On the other hand, the oPOEMS clearly beats pPOEMS on function 11.

4. CPU TIMING EXPERIMENTS

For the timing experiment the algorithms were run for about 60 seconds. The experiments have been conducted with an Intel Pentium-M 1400 MHz under MS Windows using the C-code provided. No difference between the two algorithms has been observed. The time per function evaluation was 2.4×10^{-6}, 2.7×10^{-6}, 4.0×10^{-6}, 8.5×10^{-6}, 13×10^{-6} seconds in dimensions 2, 3, 5, 10 and 20 respectively. This means that the overhead related to the selection and replacement of the prototypes can be considered neglectable.

5. CONCLUSIONS

Original POEMS algorithm and its extension using a pool of candidate prototypes were compared. Results show that for on some functions in lower dimensions the extended version works better than the original one. However, in higher dimension (e.g. 20D) no significant improvement has been observed.

Another general observation is, that the POEMS algorithm successfully solves the separable functions while it has difficulties with solving ill-conditioned, multi-modal and weak structure functions, especially in 20D. Perhaps, this is an implication of the mechanism for sampling the solution space used in the algorithm. The evolved actions operate on 1 dimension only (making axis-parallel modifications only), nevertheless, the whole action sequences (hypermutations) result in non-axis-paralel steps. However, it is easier for this algorithm to optimize separable functions where the evolved hypermutations are composed of actions, each affecting the current prototype in an isolated way. On the other hand, the ill-conditioned, multi-modal and weak structure functions require the hypermutations being composed of actions that have a synergic effect on the current prototype.

The utilization of the pool of prototypes in the pPOEMS has been proposed with the aim to make the algorithm more resistant against stagnation and getting stuck in a local optimum. Our analyses show that in the latter stages of the run the pool of candidate prototypes serves as a buffer of high-quality solutions that sample one particular region of the search space. If the algorithm fails to find an improving modification to one of them another prototype is picked from the pool for the subsequent iteration and the pool is updated accordingly. The prototypes are tried one by one till the algorithm escapes hopefully by finding an improvement modification to one of them. Then the search continues by improving this prototype. This is perhaps what makes the pPOEMS better than the oPOEMS on some functions. However, a detailed investigation of the pPOEMS behaviour remains for future work.

Acknowledgement

This work has been supported by the research program No. MSM 6840770012 "Transdisciplinary Research in Biomedical Engineering II" of the CTU in Prague. The author would like to acknowledge the great and hard work of the BBOB 2010 team.

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] J. Kubalik. Black-box optimization benchmarking of prototype optimization with evolved improvement steps for noiseless function testbed. In F. Rothlauf, editor, *GECCO (Companion)*, pages 2303–2308. ACM, 2009.

[5] J. Kubalík. Solving the sorting network problem using iterative optimization with evolved hypermutations. In *GECCO '09: Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 301–308, New York, NY, USA, 2009. ACM.

[6] J. Kubalík. Solving the multiple sequence alignment problem using prototype optimization with evolved improvement steps. In *ICANNGA*, pages 183–192, 2009.

[7] J. Kubalík. Efficient stochastic local search algorithm for solving the shortest common supersequence problem. In *Accepted for presentation at GECCO '10*. ACM, 2010.

[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.

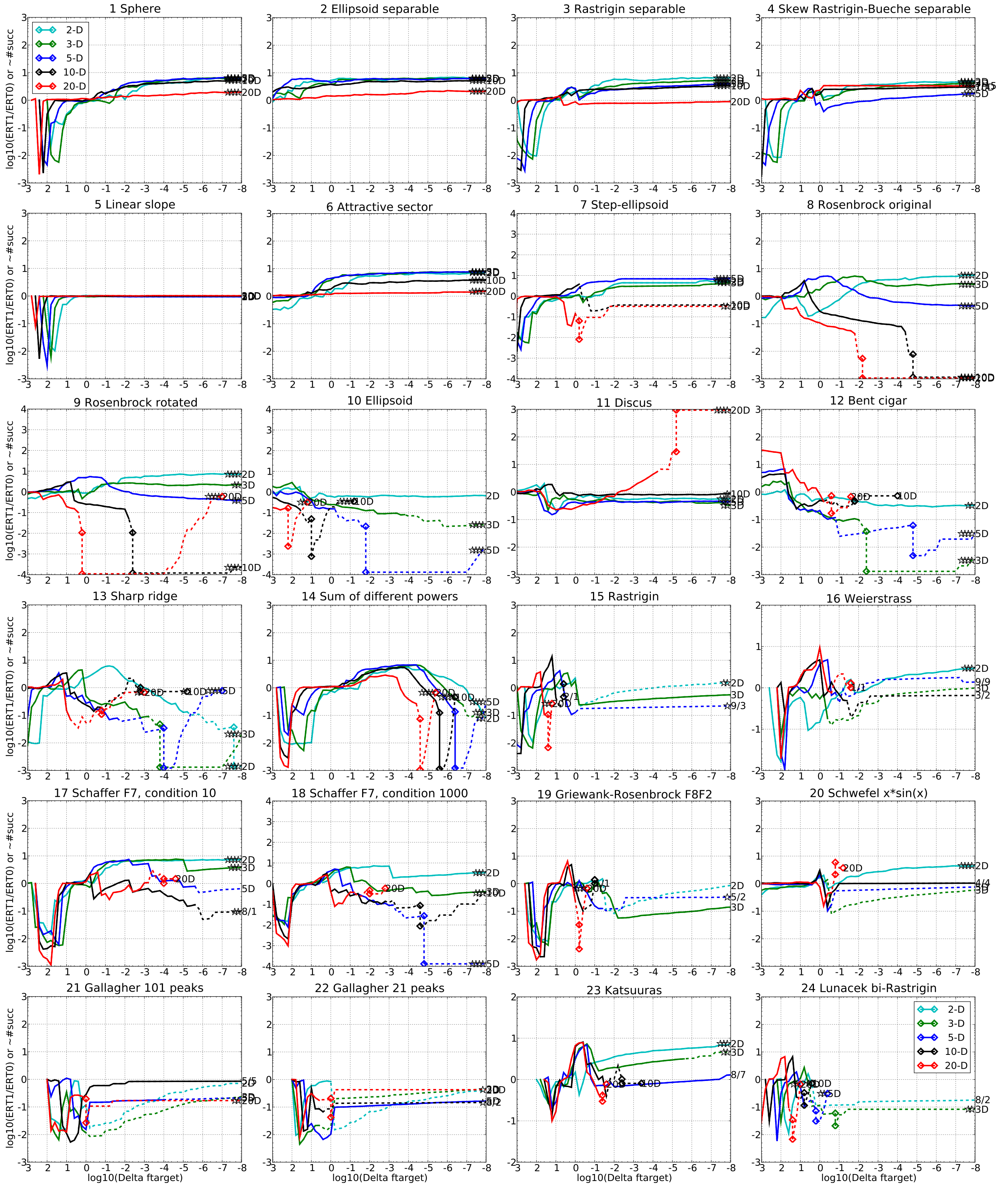

Figure 2: ERT ratio of pPOEMS divided by oPOEMS versus $\log_{10}(\Delta f)$ for f_1–f_{24} in 2, 3, 5, 10, 20, 40-D. Ratios $< 10^0$ indicate an advantage of pPOEMS, smaller values are always better. The line gets dashed when for any algorithm the ERT exceeds thrice the median of the trial-wise overall 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 pPOEMS. The line ends when no algorithm reaches Δf anymore. The number of successful trials is given, only if it was in $\{1 \dots 9\}$ for pPOEMS (1st number) and non-zero for oPOEMS (2nd number). Results are significant with $p = 0.05$ for one star and $p = 10^{-\#\star}$ otherwise, with Bonferroni correction within each figure.

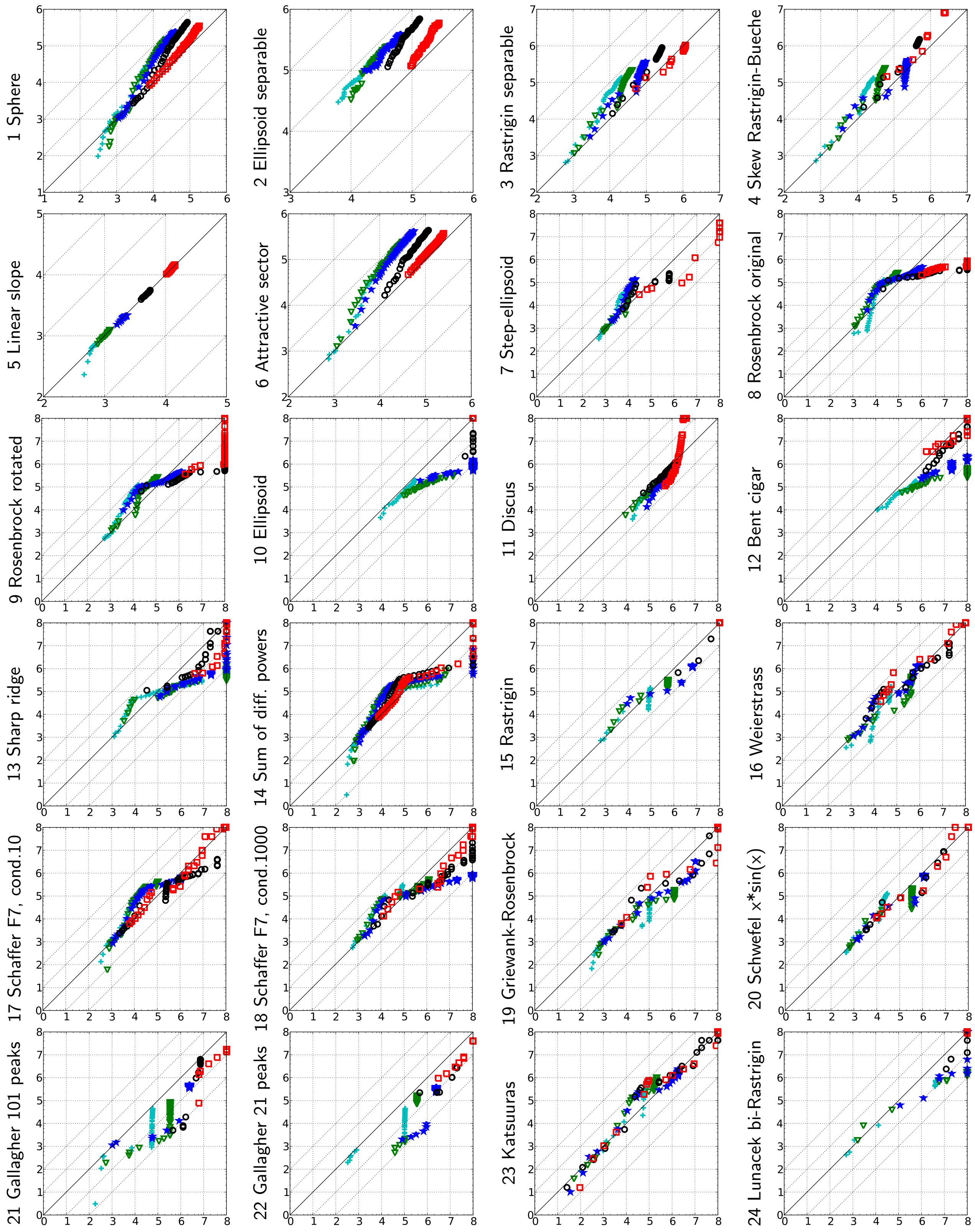

Figure 3: Expected running time (ERT in log10 of number of function evaluations) of pPOEMS versus oPOEMS for 46 target values $\Delta f \in [10^{-8}, 10]$ in each dimension for functions f_1–f_{24}. Markers on the upper or right egde indicate that the target value was never reached by pPOEMS or oPOEMS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

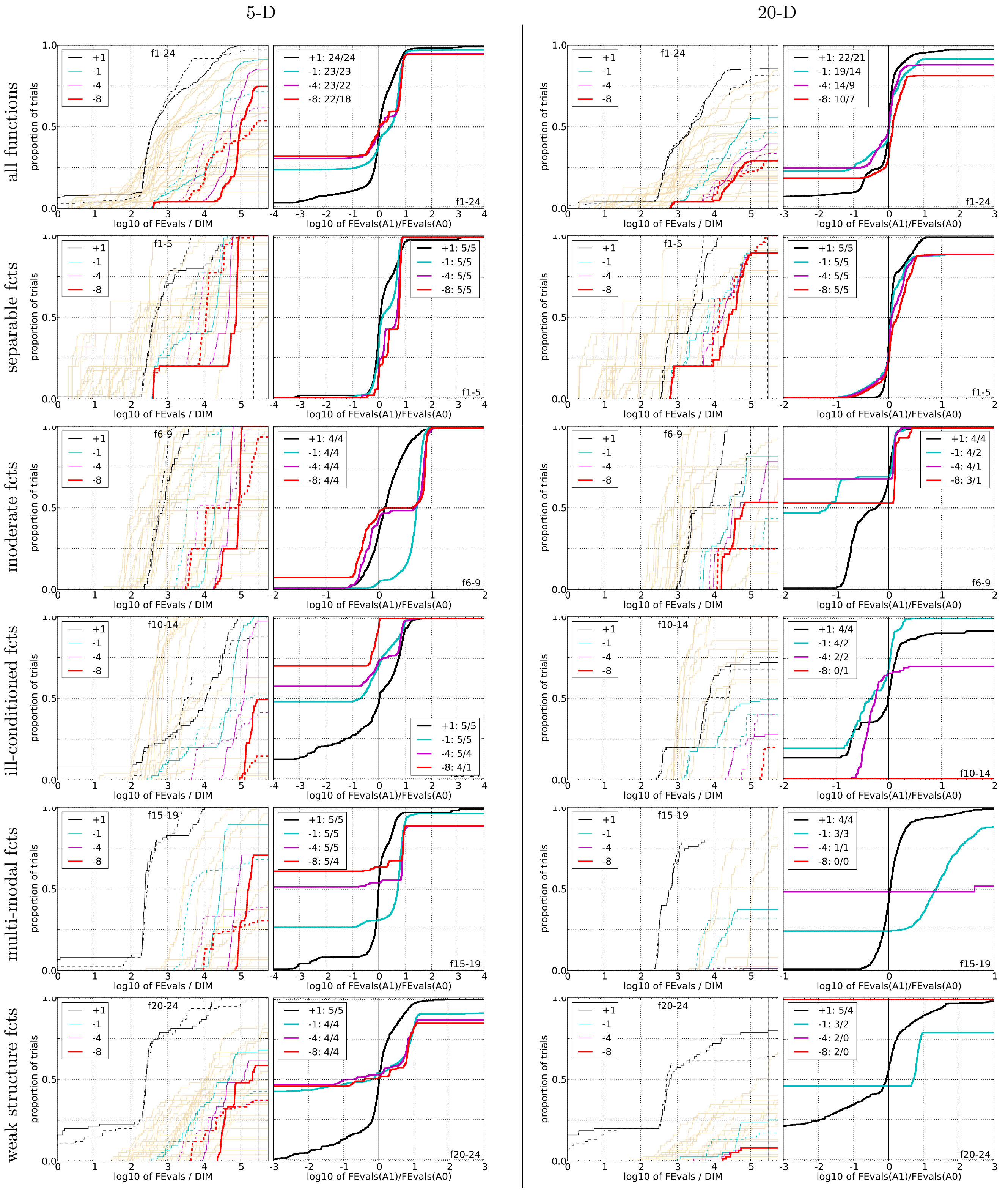

Figure 4: Empirical cumulative distributions (ECDF) of run lengths and speed-up ratios in 5-D (left) and 20-D (right). Left sub-columns: ECDF of the number of function evaluations divided by dimension D (FEvals/D) to reach a target value $f_{\mathrm{opt}} + \Delta f$ with $\Delta f = 10^k$, where $k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

5-D

Δf	1e+1	1e+0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathbf{f_1}$	11	12	12	12	12	12	15/15
0: org	100	140	330	**1.2e3**^{*3}	**2.0e3**^{*3}	**2.8e3**^{*3}	15/15
1: poo	97	130	610	5.8e3	1.2e4	1.8e4	15/15
$\mathbf{f_2}$	83	87	88	90	92	94	15/15
0: org	**200**^{*3}	**300**^{*3}	**340**^{*3}	**440**^{*3}	**550**^{*3}	**640**^{*3}	15/15
1: poo	1.2e3	1.3e3	1.7e3	2.5e3	3.1e3	3.9e3	15/15
$\mathbf{f_3}$	720	1600	1600	1600	1700	1700	15/15
0: org	4	**11**^{*2}	36	**42**^{*3}	**48**^{*3}	**54**^{*3}	15/15
1: poo	4.8	30	64	120	160	200	15/15
$\mathbf{f_4}$	810	1600	1700	1800	1900	1900	15/15
0: org	4.9	43	120	120	**120**^{*2}	**120**^{*2}	15/15
1: poo	6.8	36	79	120	160	200	15/15
$\mathbf{f_5}$	10	10	10	10	10	10	15/15
0: org	160	200	220	230	230	230	15/15
1: poo	150	200	210	220	220	220	15/15
$\mathbf{f_6}$	110	210	280	580	1000	1300	15/15
0: org	26	**46**^{*3}	**54**^{*3}	**45**^{*3}	**36**^{*3}	**37**^{*3}	15/15
1: poo	31	200	310	310	270	280	15/15
$\mathbf{f_7}$	24	320	1200	1600	1600	1600	15/15
0: org	83	17	**9**^{*3}	**11**^{*3}	**11**^{*3}	**12**^{*3}	15/15
1: poo	89	26	43	76	76	79	15/15
$\mathbf{f_8}$	73	270	340	390	410	420	15/15
0: org	58	**52**^{*3}	**120**^{*2}	690	1.5e3	2.2e3	15/15
1: poo	84	260	370	580	**800**^{*3}	**1.1e3**^{*3}	15/15
$\mathbf{f_9}$	35	130	210	300	340	370	15/15
0: org	**110***	**120**^{*3}	**180**^{*2}	1.0e3	1.9e3	2.8e3	11/15
1: poo	290	560	600	720	990	1.2e3	15/15
$\mathbf{f_{10}}$	350	500	570	630	830	880	15/15
0: org	1.4e3	4.1e3	1.7e4	∞	∞	∞ *1.5e6*	0/15
1: poo	550	640	**740***	**1.0e3**^{*3}	**1.0e3**^{*3}	**1.2e3**^{*3}	11/15
$\mathbf{f_{11}}$	140	200	760	1200	1500	1700	15/15
0: org	560	920	470	570	680	740	11/15
1: poo	95	390	200	250	290	340	15/15
$\mathbf{f_{12}}$	110	270	370	460	1300	1500	15/15
0: org	9.5e3	6.5e3	5.7e4	4.6e4	∞	∞ *1.5e6*	0/15
1: poo	2.1e3	1.4e3	**1.4e3**^{*2}	**1.9e3**^{*2}	**1.0e3**^{*3}	**1.4e3**^{*3}	8/15
$\mathbf{f_{13}}$	130	190	250	1300	1800	2300	15/15
0: org	900	2.9e3	9.1e3	1.6e4	∞	∞ *1.5e6*	0/15
1: poo	450	790	1.1e3	**390**^{*2}	**1.3e3**^{*3}	**1.0e4**^{*3}	0/15
$\mathbf{f_{14}}$	9.8	41	58	140	250	480	15/15
0: org	110	43	79	**140**^{*3}	500	∞ *1.5e6*	0/15
1: poo	61	47	140	800	1.1e3	**2.1e3**^{*3}	3/15
$\mathbf{f_{15}}$	510	9300	1.9e4	2.0e4	2.1e4	2.1e4	14/15
0: org	**18***	230	340	320	310	310	3/15
1: poo	57	29	60	62	64	66	9/15
$\mathbf{f_{16}}$	120	610	2700	1.0e4	1.2e4	1.2e4	15/15
0: org	11	14	92	74	68	66	9/15
1: poo	9.6	52	40	120	110	120	9/15
$\mathbf{f_{17}}$	5.2	210	900	3700	6400	7900	15/15
0: org	220	17	**15**^{*3}	**14**^{*3}	98	140	9/15
1: poo	170	22	88	71	73	82	15/15
$\mathbf{f_{18}}$	100	380	4000	9300	1.1e4	1.2e4	15/15
0: org	19	**23**^{*3}	**33**^{*2}	450	∞	∞ *1.5e6*	0/15
1: poo	18	95	34	39	**55**^{*3}	**69**^{*3}	15/15
$\mathbf{f_{19}}$	1	1	240	1.2e5	1.2e5	1.2e5	15/15
0: org	1.0e3	3.1e4	1.1e4	87	86	86	2/15
1: poo	980	1.8e4	1.6e3	27	27	27	5/15
$\mathbf{f_{20}}$	16	850	3.8e4	5.4e4	5.5e4	5.5e4	14/15
0: org	84	8.4	30	21	21	21	9/15
1: poo	81	17	17	13	14	15	11/15
$\mathbf{f_{21}}$	41	1200	1700	1700	1700	1800	14/15
0: org	25	740	1.4e3	1.4e3	1.4e3	1.4e3	6/15
1: poo	28	11	210	240	260	280	12/15
$\mathbf{f_{22}}$	71	390	940	1000	1000	1100	14/15
0: org	1.1e3	2.3e3	2.6e3	2.4e3	2.3e3	2.3e3	6/15
1: poo	29	24	270	280	330	360	13/15
$\mathbf{f_{23}}$	3	520	1.4e4	3.2e4	3.3e4	3.4e4	15/15
0: org	12	21	41	57	55	54	7/15
1: poo	3.4	71	29	42	47	53	8/15
$\mathbf{f_{24}}$	1600	2.2e5	6.4e6	9.6e6	1.3e7	1.3e7	3/15
0: org	72	∞	∞	∞	∞	∞ *1.5e6*	0/15
1: poo	39	**7.2***	∞	∞	∞	∞ *1.5e6*	0/15

20-D

Δf	1e+1	1e+0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathbf{f_1}$	43	43	43	43	43	43	15/15
0: org	180	410	850	**1.8e3**^{*3}	**2.8e3**^{*3}	**3.7e3**^{*3}	15/15
1: poo	200	470	1.1e3	2.6e3	4.5e3	7.1e3	15/15
$\mathbf{f_2}$	380	390	390	390	390	390	15/15
0: org	250	290	**340**^{*2}	**450**^{*3}	**560**^{*3}	**660**^{*3}	15/15
1: poo	310	420	510	740	1.2e3	1.4e3	15/15
$\mathbf{f_3}$	5100	7600	7600	7600	7600	7700	15/15
0: org	10	64	130	140	150	150	15/15
1: poo	13	57	100	110	120	130	15/15
$\mathbf{f_4}$	4700	7600	7700	7700	7800	1.4e5	9/15
0: org	14	110	310	310	310	18	15/15
1: poo	31	250	1.0e3	1.1e3	1.1e3	59	7/15
$\mathbf{f_5}$	41	41	41	41	41	41	15/15
0: org	250	310	330	350	350	350	15/15
1: poo	250	310	340	350	360	360	15/15
$\mathbf{f_6}$	1300	2300	3400	5200	6700	8400	15/15
0: org	32	**28**^{*3}	**26**^{*3}	**26**^{*3}	**27**^{*3}	**27**^{*3}	15/15
1: poo	36	32	32	34	35	38	15/15
$\mathbf{f_7}$	1400	4300	9500	1.7e4	1.7e4	1.7e4	15/15
0: org	23	2.0e3	∞	∞	∞	∞ *6.0e6*	0/15
1: poo	22	280	**1.8e3***	**2.4e3***	**2.4e3***	**2.3e3***	2/15
$\mathbf{f_8}$	2000	3900	4000	4200	4400	4500	15/15
0: org	500	900	1.6e3	∞	∞	∞ *6.0e6*	0/15
1: poo	**100**^{*2}	**97**^{*3}	**110**^{*3}	**120**^{*3}	**150**^{*3}	**190**^{*3}	15/15
$\mathbf{f_9}$	1700	3100	3300	3500	3600	3700	15/15
0: org	1.1e3	∞	∞	∞	∞	∞ *6.0e6*	0/15
1: poo	**210**^{*3}	**330**^{*3}	**440**^{*3}	**970**^{*3}	**2.5e3**^{*3}	**2.4e4**^{*3}	0/15
$\mathbf{f_{10}}$	7400	8700	1.1e4	1.5e4	1.7e4	1.7e4	15/15
0: org	∞	∞	∞	∞	∞	∞ *6.0e6*	0/15
1: poo	∞	∞	∞	∞	∞	∞ *6.0e6*	0/15
$\mathbf{f_{11}}$	1000	2200	6300	9800	1.2e4	1.5e4	15/15
0: org	480	400	220	220	**240**^{*2}	**260**^{*3}	15/15
1: poo	**110**^{*2}	**110**^{*3}	**84**^{*2}	360	7.1e3	∞ *6.0e6*	0/15
$\mathbf{f_{12}}$	1000	1900	2700	4100	1.2e4	1.4e4	15/15
0: org	**1.6e3***	4.7e3	∞	∞	∞	∞ *6.0e6*	0/15
1: poo	3.4e3	3.9e3	9.3e3	∞	∞	∞ *6.0e6*	0/15
$\mathbf{f_{13}}$	650	2000	2800	1.9e4	2.4e4	3.0e4	15/15
0: org	6.3e3	4.2e4	∞	∞	∞	∞ *6.0e6*	0/15
1: poo	940	2.2e3	**6.2e3***	**4.6e3***	∞	∞ *6.0e6*	0/15
$\mathbf{f_{14}}$	75	240	300	930	1600	1.6e4	15/15
0: org	98	65	120	**130**^{*3}	∞	∞ *6.0e6*	0/15
1: poo	100	72	140	350	**4.5e3**^{*3}	∞ *6.0e6*	0/15
$\mathbf{f_{15}}$	3.0e4	1.5e5	3.1e5	3.2e5	4.5e5	4.6e5	15/15
0: org	∞	∞	∞	∞	∞	∞ *6.0e6*	0/15
1: poo	∞	∞	∞	∞	∞	∞ *6.0e6*	0/15
$\mathbf{f_{16}}$	1400	2.7e4	7.7e4	1.9e5	2.0e5	2.2e5	15/15
0: org	14	**2.6**^{*2}	510	∞	∞	∞ *6.0e6*	0/15
1: poo	26	24	1.1e3	∞	∞	∞ *6.0e6*	0/15
$\mathbf{f_{17}}$	63	1000	4000	3.1e4	5.6e4	8.0e4	15/15
0: org	98	27	130	300	∞	∞ *6.0e6*	0/15
1: poo	98	31	54	310	∞	∞ *6.0e6*	0/15
$\mathbf{f_{18}}$	620	4000	2.0e4	6.8e4	1.3e5	1.5e5	15/15
0: org	21	120	270	∞	∞	∞ *6.0e6*	0/15
1: poo	21	51	110	∞	∞	∞ *6.0e6*	0/15
$\mathbf{f_{19}}$	1	1	3.4e5	6.2e6	6.7e6	6.7e6	15/15
0: org	6.1e3	6.1e6	∞	∞	∞	∞ *6.0e6*	0/15
1: poo	6.3e3	1.4e6	∞	∞	∞	∞ *6.0e6*	0/15
$\mathbf{f_{20}}$	82	4.6e4	3.1e6	5.5e6	5.6e6	5.6e6	14/15
0: org	120	2.4	8.6	∞	∞	∞ *6.0e6*	0/15
1: poo	130	1.8	∞	∞	∞	∞ *6.0e6*	0/15
$\mathbf{f_{21}}$	560	6500	1.4e4	1.5e4	1.6e4	1.8e4	15/15
0: org	1.1e4	6.0e3	∞	∞	∞	∞ *6.0e6*	0/15
1: poo	140	1.2e3	960	1.2e3	1.1e3	970	4/15
$\mathbf{f_{22}}$	470	5600	2.3e4	2.5e4	2.7e4	1.3e5	12/15
0: org	6.5e3	7.0e3	∞	∞	∞	∞ *6.0e6*	0/15
1: poo	2.0e3	1.4e3	1.7e3	1.6e3	1.5e3	290	2/15
$\mathbf{f_{23}}$	3.2	1600	6.7e4	4.9e5	8.1e5	8.4e5	15/15
0: org	29	**48**^{*2}	47	∞	∞	∞ *6.0e6*	0/15
1: poo	4.9	320	36	∞	∞	∞ *6.0e6*	0/15
$\mathbf{f_{24}}$	1.3e6	7.5e6	5.2e7	5.2e7	5.2e7	5.2e7	3/15
0: org	∞	∞	∞	∞	∞	∞ *6.0e6*	0/15
1: poo	**64***	∞	∞	∞	∞	∞ *6.0e6*	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: org is oPOEMS and 1: poo is pPOEMS. 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.