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

Forename Name

ABSTRACT

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

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 [6, 5]. The original version of the POEMS algorithm has been tested on the noise-free BBOB 2009 testbed [4].

This paper compares two variants of the POEMS on the noiseless testbed where series of experiments were carried out on the noise-free problems for 2, 3, 5 and 10 D.

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

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

where

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

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 $\sigma_{i,k}^2$ decreases to maximally possible extent. In the opposite case, the $\sigma_{i,k}^2$ is decreased less or it can even increase.

This can be interpreted so that if for the given value of σ_i^2 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_i^2. 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_i^2 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 $10^5 \times D$ 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_i^2 for $i = 1 \ldots D$ fall below 10^{-11} then they are reinitialized to the original values $0.25 * (ubound - lbound)$ while **the current prototype remains unchanged**.

2.2 Variant rPOEMS

The first variant called rPOEMS uses a simple restarting strategy that generates a new prototype whenever there is no chance that there could be generated an improving move (hypermutation) for the current prototype. This variant differs from the original one in a way that once the values of

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

σ_i^2 for $i = 1 \ldots D$ fall below 10^{-11} then **a completely new prototype is generated by random** and the values of σ_i^2 are reinitialized to the original values $0.25 * (ubound - lbound)$. Thus, once the stagnation of the run is detected the search is restarted from a new randomly chosen point of the search space.

2.3 Variant pPOEMS

The second variant, denoted as pPOEMS, uses a pool of candidate prototypes of size $PoolSize$ from which one prototype is chosen in each iteration. Each candidate prototype maintains its own σ_i^2 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 1). The values of σ_i^2 of the current prototype are adapted accordingly where the deltas δ_i are calculated based on the modified and the current prototypes' 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 1) and the values of σ_i^2 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 $(i+1)\%PoolSize$, 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 1) by the modified prototype. The values of σ_i^2 of the replacement prototype are adapted accordingly where the deltas δ_i are calculated based on the modified and the replacement prototypes' variable values.

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

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

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

Note, the important difference between the two POEMS variants is that rPOEMS dispose of the ability to restart the search in a new randomly generated point of the search space whenever the stagnation is detected while the pPOEMS can only operate on the prototype pool of limited size. No restarts are allowed for pPOEMS.

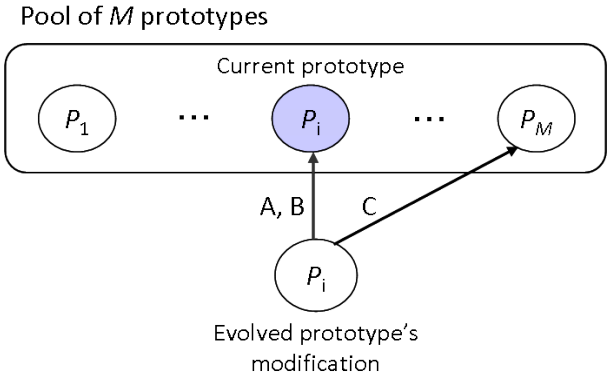

Figure 1: Schema of the pool of prototypes used in pPOEMS.

3. EXPERIMENTAL PROCEDURE

No tuning of POEMS control parameters was done. The configuration was parameterized solely by the dimension of the problem at hand. The parameter setting was identical for all functions so the *crafting effort* is zero. The POEMS algorithm was configured as follows:

- $MaxGenes = D$, $NicheSize = 20$,
- $PopSize = MaxGenes * NicheSize$,
- $P_{cross} = 0.75$, $P_{mutate} = 0.5$, $\alpha = 0.2$,
- 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: $D \times 10^5$.

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

4. RESULTS

Results from experiments according to [2] on the benchmark functions given in [1, 3] are presented in Figures 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 [2, 7]. **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.

I used the runcomp2.py script to generate the plots however it failed to generate the scatter plots. Perhaps, this is due to the fact that the data from experiments with 20D were not available.

5. CONCLUSIONS

Both the tested variants of the POEMS algorithm extends the original approach in order to make it more resistant against stagnation and getting stuck in a local optimum.

An interesting observation is that pPOEMS performs comparably to the rPOEMS on most of the problems and clearly outperforms rPOEMS on problems $\{f_8, \ldots, f_{14}$ and f_{18}. Our first analyses reveal an interesting aspect of the utilization of the pool of candidate prototypes in the pPOEMS. It turns out that in the latter stages of the run the pool serves as let say a buffer of high-quality prototypes 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. Simultaneously, the pool is being updated in each iteration. 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. It seems that this mechanism is much more robust than the pure restarting strategy used in rPOEMS.

Final version of this paper will contain detailed analysis of the pPOEMS behavior based on experiments carried out for 2, 3, 5, 10 and 20 D.

Acknowledgments

The author would like to acknowledge the great and hard work of the BBOB 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 *GECCO '09: Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference*, pages 2303–2308, New York, NY, USA, 2009. ACM.

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

[6] 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.

[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.

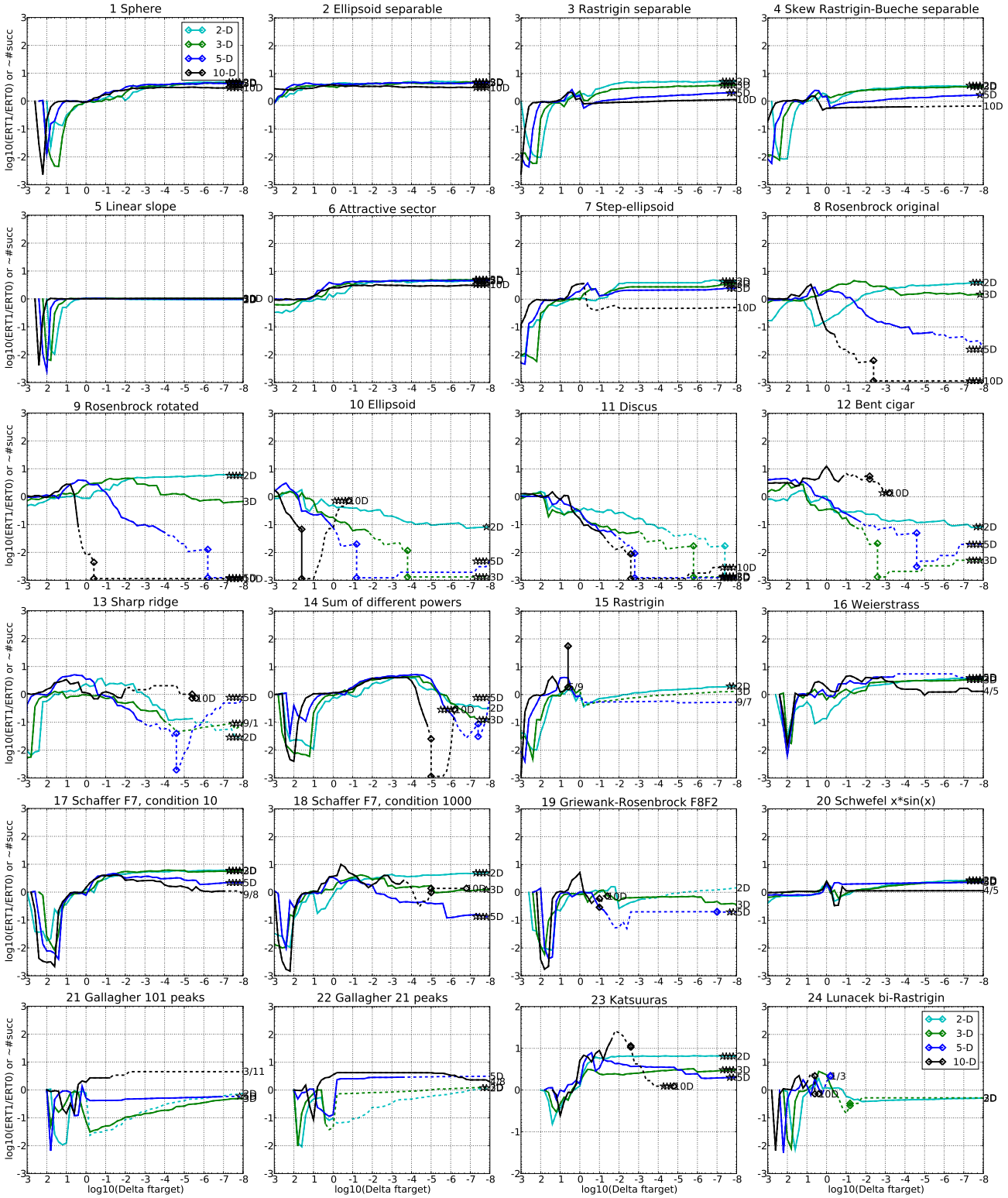

Figure 2: ERT ratio of pPOEMS divided by rPOEMS versus $\log_{10}(\Delta f)$ for f_1–f_{24} in 2, 3, 5, 10, 20, 40-D. Ratios $< 10^0$ indicate an advantage of pPOEMS, smaller values are always better. The line gets dashed when for any algorithm the ERT exceeds thrice the median of the trial-wise overall 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 rPOEMS (2nd number). Results are significant with $p = 0.05$ for one star and $p = 10^{-\#\star}$ otherwise, with Bonferroni correction within each figure.

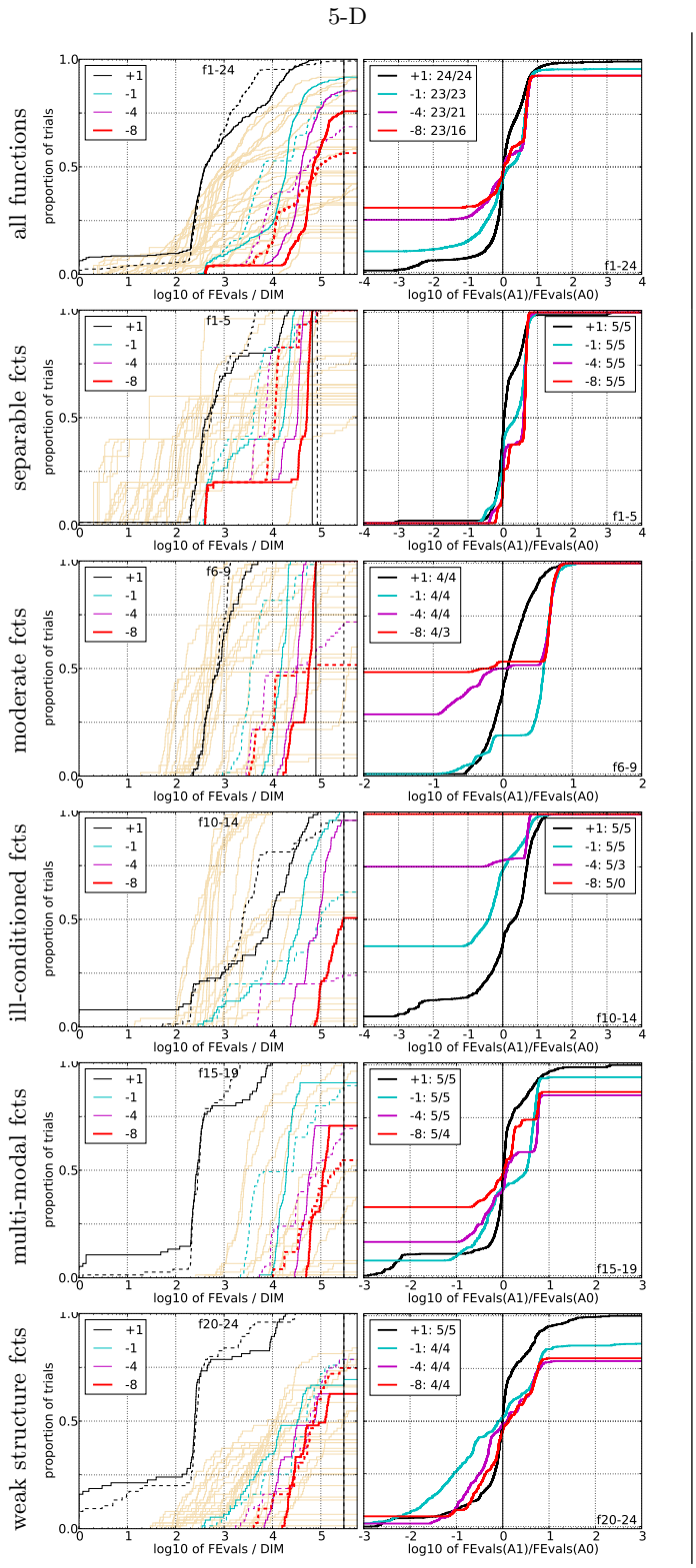

Figure 3: Empirical cumulative distributions (ECDF) of run lengths and speed-up ratios in 5-D (left) and 20-D (right). Left sub-columns: ECDF of the number of function evaluations divided by dimension D (FEvals/D) to reach a target value $f_{\text{opt}} + \Delta f$ with $\Delta f = 10^k$, where $k \in \{1, -1, -4, -8\}$ is given by the first value in the legend, for pPOEMS (solid) and rPOEMS (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 rPOEMS, 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 20-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: poe	95	140	300	**1.2e3**$^{\star 3}$	**2.0e3**$^{\star 3}$	**2.9e3**$^{\star 3}$	15/15
1: poe	97	130	610	4.6e3	8.3e3	1.2e4	15/15
f_2	83	87	88	90	92	94	15/15
0: poe	**220**$^{\star 3}$	**260**$^{\star 3}$	**310**$^{\star 3}$	**420**$^{\star 3}$	**520**$^{\star 3}$	**640**$^{\star 3}$	15/15
1: poe	920	1.1e3	1.3e3	1.9e3	2.4e3	2.9e3	15/15
f_3	720	1600	1600	1600	1700	1700	15/15
0: poe	4.6	**16**$^{\star 2}$	56	62	69	**73***	15/15
1: poe	4.8	25	56	87	110	140	15/15
f_4	810	1600	1700	1800	1900	1900	15/15
0: poe	5.4	22	75	76	79	83	15/15
1: poe	6.8	34	66	88	110	140	15/15
f_5	10	10	10	10	10	10	15/15
0: poe	160	200	210	220	220	220	15/15
1: poe	150	200	210	220	220	220	15/15
f_6	110	210	280	580	1000	1300	15/15
0: poe	26	**45**$^{\star 3}$	**55**$^{\star 3}$	**45**$^{\star 3}$	**36**$^{\star 3}$	**37**$^{\star 3}$	15/15
1: poe	31	160	240	210	170	170	15/15
f_7	24	320	1200	1600	1600	1600	15/15
0: poe	84	14	**29***	**26***	**26***	**26***	15/15
1: poe	89	26	38	56	56	60	15/15
f_8	73	270	340	390	410	420	15/15
0: poe	66	**110***	260	3.3e3	9.2e3	1.6e4	1/15
1: poe	84	200	290	**420**$^{\star 2}$	**550**$^{\star 3}$	**700**$^{\star 3}$	15/15
f_9	35	130	210	300	340	370	15/15
0: poe	140	**120**$^{\star 3}$	300	5.2e3	3.2e4	*∞1.5e6*	0/15
1: poe	290	460	430	**520**$^{\star 2}$	**660**$^{\star 3}$	**840**$^{\star 3}$	15/15
f_{10}	350	500	570	630	830	880	15/15
0: poe	1.9e3	7.0e3	3.7e4	∞	∞	*∞1.5e6*	0/15
1: poe	420	**550**$^{\star 2}$	**700**$^{\star 2}$	**900**$^{\star 3}$	**950**$^{\star 3}$	**1.2e3**$^{\star 3}$	12/15
f_{11}	140	200	760	1200	1500	1700	15/15
0: poe	110	970	2.4e3	∞	∞	*∞1.5e6*	0/15
1: poe	97	290	**140**$^{\star 2}$	**170**$^{\star 3}$	**210**$^{\star 3}$	**240**$^{\star 3}$	15/15
f_{12}	110	270	370	460	1300	1500	15/15
0: poe	**590**$^{\star 2}$	1.6e3	5.3e3	2.2e4	∞	*∞1.5e6*	0/15
1: poe	1.7e3	1.3e3	1.3e3	**1.7e3**$^{\star 2}$	**930**$^{\star 3}$	**1.2e3**$^{\star 3}$	9/15
f_{13}	130	190	250	1300	1800	2300	15/15
0: poe	**83**$^{\star 3}$	**160**$^{\star 2}$	880	3.6e3	∞	*∞1.5e6*	0/15
1: poe	360	540	760	**310**$^{\star 2}$	**810**$^{\star 3}$	**4.8e3**$^{\star 3}$	1/15
f_{14}	9.8	41	58	140	250	480	15/15
0: poe	100	41	74	**130**$^{\star 3}$	**220**$^{\star 3}$	4.7e4	0/15
1: poe	61	47	140	620	780	**1.9e3**$^{\star 3}$	1/15
f_{15}	510	9300	1.9e4	2.0e4	2.1e4	2.1e4	14/15
0: poe	**13**$^{\star 3}$	43	110	100	120	110	7/15
1: poe	52	38	58	59	59	60	9/15
f_{16}	120	610	2700	1.0e4	1.2e4	1.2e4	15/15
0: poe	11	45	28	15	15	20	15/15
1: poe	11	42	69	65	81	84	10/15
f_{17}	5.2	210	900	3700	6400	7900	15/15
0: poe	230	18	**16**$^{\star 3}$	**14**$^{\star 3}$	**19**$^{\star 2}$	**27**$^{\star 3}$	15/15
1: poe	120	14	64	52	52	58	15/15
f_{18}	100	380	4000	9300	1.1e4	1.2e4	15/15
0: poe	18	**27**$^{\star 3}$	**9.2***	47	98	320	4/15
1: poe	17	93	25	29	40	**49**$^{\star 3}$	15/15
f_{19}	1	1	240	1.2e5	1.2e5	1.2e5	15/15
0: poe	1.0e3	1.6e4	2.0e3	180	180	180	0/15
1: poe	930	1.7e4	**550**$^{\star 2}$	36	36	36	4/15
f_{20}	16	850	3.8e4	5.4e4	5.5e4	5.5e4	14/15
0: poe	84	9.7	6.1	4.5	4.6	4.8	15/15
1: poe	67	20	12	9.3	10	11	12/15
f_{21}	41	1200	1700	1700	1700	1800	14/15
0: poe	30	200	320	320	320	320	14/15
1: poe	28	100	130	150	170	180	13/15
f_{22}	71	390	940	1000	1000	1100	14/15
0: poe	25	130	150	150	150	150	15/15
1: poe	27	17	380	420	440	460	12/15
f_{23}	3	520	1.4e4	3.2e4	3.3e4	3.4e4	15/15
0: poe	9.2	30	8.9	9.2	12	20	12/15
1: poe	9.2	42	30	34	37	40	10/15
f_{24}	1600	2.2e5	6.4e6	9.6e6	1.3e7	1.3e7	3/15
0: poe	23	23	∞	∞	∞	*∞1.5e6*	0/15
1: poe	33	46	∞	∞	∞	*∞1.5e6*	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: poe is rPOEMS and 1: poe 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.