BBOB-Benchmarking the Rosenbrock's Local Search Algorithm

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

ABSTRACT

The restarted Rosenbrock's optimization algorithm is tested on the BBOB 2009 testbed. The algorithm turned out to be very efficient for functions with simple structure (independently of dimensionality), but is not reliable for multimodal or ill-conditioned functions.

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, Experimentation, Performance, Reliability

Keywords

Benchmarking, Black-box optimization, Local search, Rosenbrock's algorithm, Evolutionary computation

1. INTRODUCTION

The Rosenbrock's algorithm [6] is a classical local search technique—it maintains the best-so-far solution and searches in its neighborhood for improvements. What distinguishes this algortihm from other local search techniques is the fact that it also maintains a model of the current local neighborhood—it adapts the model orientation and size. This feature can be observed in many recent successful optimization techniques, e.g. in CMA-ES [4].

2. ALGORITHM DECRIPTION

The Rosenbrock's local search technique is depicted as Alg. 1. The model of the local neighborhood consists of D vectors $\mathbf{e}_1, \ldots, \mathbf{e}_D$ forming the orthonormal basis, and of D multipliers (or step lengths, if you want) $d_1, \ldots, d_D$, where D is the dimensionality of the search space. In each iteration, the algorithm performs a kind of pattern line search along the directions given by the orthornormal basis. If in one direction $\mathbf{e}_i$ an improvement is found, next time (after trying all other directions) a point α times further in that direction is sampled; if no improvement is found in the $\mathbf{e}_i$ direction, next time a closer point on the other side is sampled (governed by the β parameter). Usually, the values of parameters are $\alpha = 2$ and $\beta = \frac{1}{2}$.

Algorithm 1: Rosenbrock's Algorithm

Input: $\alpha > 1$, $\beta \in (0, 1)$
begin
    $\mathbf{x} \leftarrow$ `Initialize()`; $\mathbf{x}_o \leftarrow \mathbf{x}$
    $\{\mathbf{e}_1, \ldots, \mathbf{e}_D\} \leftarrow$ `InitOrthoBasis()`
    $\{d_1, \ldots, d_D\} \leftarrow$ `InitMultipliers()`
    while not `TerminationCondition()` **do**
        for *i=1...D* **do**
            $\mathbf{y} \leftarrow \mathbf{x} + d_i \mathbf{e}_i$
            if `BetterThan(`y,x`)` **then**
                $\mathbf{x} \leftarrow \mathbf{y}$
                $d_i \leftarrow \alpha \cdot d_i$
            else
                $d_i \leftarrow -\beta \cdot d_i$
        if `AtLeastOneSuccessInAllDirs()` **and** `AtLeastOneFailInAllDirs()` **then**
            $\{\mathbf{e}_1, \ldots, \mathbf{e}_D\} \leftarrow$ `UpdOrthoBasis(`$\mathbf{x} - \mathbf{x}_o$`)`
            $\{d_1, \ldots, d_D\} \leftarrow$ `InitMultipliers()`
            $\mathbf{x}_o \leftarrow \mathbf{x}$
end

As soon as at least one successful and one unsuccessful move in each direction was carried out, the algorithm updates its orthonormal basis to reflect the cumulative effect of all successful steps in all directions. It also resets the multipliers to their original values. The update of the orthonormal basis is done by the Palmer's orthogonalization method [5] so that the first basis vector is always parallel to the last vector $\mathbf{x} - \mathbf{x}_0$.

The demonstration of the Rosenbrock's algorithm behaviour on 2D sphere and 2D Rosenbrock's function can be seen in Fig. 1.

To improve performance on multimodal functions, a restarting strategy (see Fig. 2) was used. Each restart begins with an initial point uniformly chosen from the interval $\langle -5, 5 \rangle^D$.

GECCO'09, July 8–12, 2009, Montréal Québec, Canada.

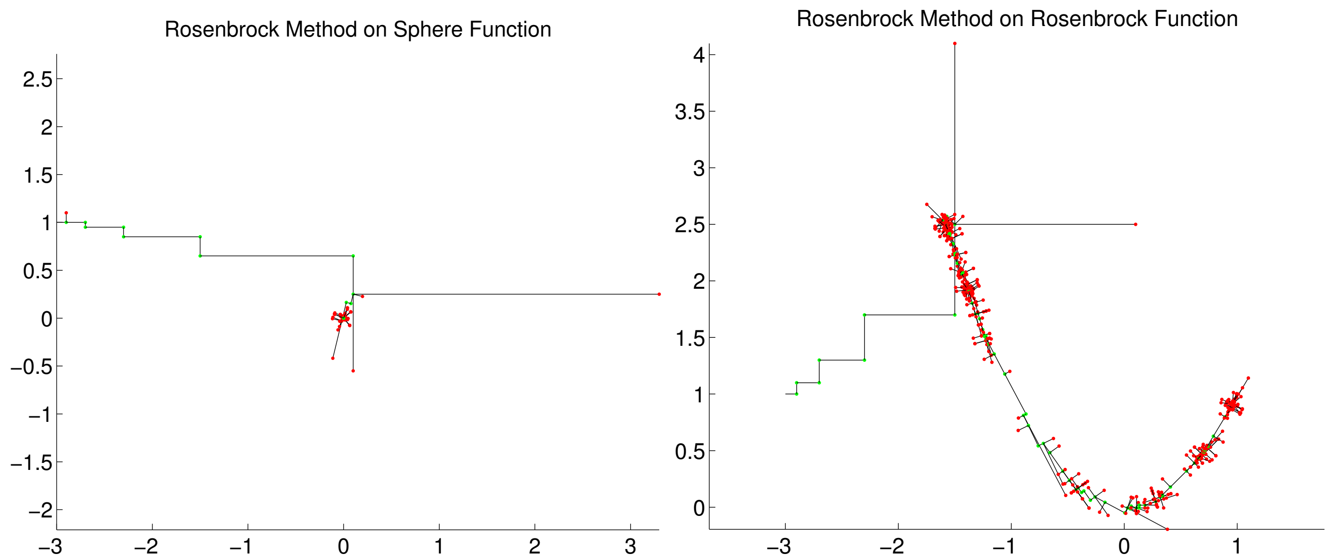

Figure 1: The behaviour of the Rosenbrock's optimization algorithm on the sphere (left) and on the Rosenbrock's (right) function

```
function [x, ilaunch] = bbobRestRosenbrockLS(FUN, ...
    DIM, ftarget, outermaxfunevals, innermaxfunevals, varargin)

options = struct('MaxFunEvals', min(1e8*DIM, outermaxfunevals), ...
    'MaxIter', inf, ...
    'Tolfun', 1e-11, ...
    'TolX', 1e-9, ...
    'StopFitness', ftarget, ...
    'Display', 'off' ...
    );

for ilaunch = 1:100; % relaunch optimizer up to 100 times
    % set initial conditions
    ropts = options;
    ropts.MaxFunEvals = min( innermaxfunevals, outermaxfunevals - feval(FUN,'evaluations'));
    ropts.dInit = 0.1;

    xstart = -5 + 10 * rand(1,DIM); % random start solution

    % try the Rosenbrock's algorithm
    x = minNDRosenbrockMod(FUN, xstart, ropts);
    if feval(FUN, 'fbest') < ftarget || feval(FUN, 'evaluations') >= options.MaxFunEvals,
        break;
    end
end
```

Figure 2: Restarting procedure for the Rosenbrock's Algorithm

2.1 Implementation Modifications

The original Rosenbrock's algorithm resets the multipliers d_i at each stage of the algorithm, i.e. each time the orthonormal basis is updated. In the particular implementation used in this article, the multipliers are not reset. It was observed that this modification improves the results on many benchmark problems—it spares some function evaluation needed to adapt the multipliers at each stage. Instead, it converges faster, allowing for more algorithm restarts.

2.2 Parameter Settings

The algorithm has 2 paramemeters, α and β. As already stated, the default values of $\alpha = 2$ and $\beta = \frac{1}{2}$ were used.

2.3 Box Constraints? No

The algorithm was run in unconstrained setting. In the BBOB comparison, some global search algorithms are present and these usually need the box constraints to be able to work. In this sense, the comparison is not quite fair since algorithms run in the unconstrained setting may spend some part of the available evaluation budget evaluating solutions lying far away from the area of the global optimum.

2.4 The Crafting Effort

The algorithm has no additional parameters, so that no further tuning is necessary; the crafting effort $\mathrm{CrR} = 0$.

2.5 Invariance Properties

With the exception of initialization, the algorithm is *invariant with respect to translation and rotation* which is demonstrated in the next section on the unrotated and rotated Rastrigin functions (3 and 15), and the Rosenbrock function (8 and 9). Looking at the separable and non-separable version of the ellipsoid function (2 and 10), we can see rather unexpected phenomenon: a different pattern which would suggest that the algorithm is not rotationally invariant. For the time being, we do not have any sound explanation for this effect, but we believe it is caused by the initialization which always initializes the algorithm with the best possible model in case of separable Ellipsoid function, while it initializes the algorithm with a bad model in case of non-separable Ellipsoid function.

The algorithm is also *invariant with respect to order-preserving transformations of the fitness function* since it uses only comparisons between two individuals.

3. EXPERIMENTAL PROCEDURE

The standard experimental procedure of BBOB was adopted: the algorithm was run on 24 test functions, 5 instances each, 3 runs on each instance. Each run was finished

- after finding a solution with fitness difference $\Delta f \leq 10^{-8}$, or
- after performing more than $10^4 \times D$ function evaluations.

Each individual launch of the basic Rosenbrock's algorithm was interrupted (and the algorithm was restarted)

- after finding a solution with fitness difference $\Delta f \leq 10^{-8}$, or
- after performing more than the allowed number of function evaluations, or
- after the model converged too much, i.e. when $\max_i |d_i| < 10^{-9}$.

4. RESULTS

Results from experiments according to [2] on the benchmark functions given in [1, 3] are presented in Figures 3 and 4 and in Table 1. The method solves at least once (out of 15 runs) 20 ($2D$), 16 ($3D$), 13 ($5D$), 8 ($10D$), and 5 ($20D$) out of 24 functions.

5. CPU TIMING EXPERIMENT

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 algorithm takes on average from $3.1 \cdot 10^{-4}$ to $3.7 \cdot 10^{-4}$ seconds per function evaluation for all tested dimensionalities (2–40) of the search space. The experiment was conducted on Intel Core 2 CPU, T5600, 1.83 GHz, 1 GB RAM with Windows XP SP3 in MATLAB R2007b.

6. CONCLUSIONS

The restarted Rosenbrock's optimization algorithm is an optimization technique suitable for functions with simple structure and low number of local optima. For the most simple functions (Sphere, Linear slope), the algorithm finds the solution very quickly and exhibits only linear scaling. The more complex functions (especially highly multimodal or ill-conditioned) are very hard for this algorithm and it cannot be considered a reliable solver for them.

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".

7. 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.

[2] N. Hansen, A. Auger, S. Finck, and R. Ros. Real-parameter black-box optimization benchmarking 2009: Experimental setup. Technical Report RR-6828, INRIA, 2009.

[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.

[4] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[5] J. R. Palmer. An improved procedure for orthogonalising the search vectors in Rosenbrock's and Swann's direct search optimisation methods. *The Computer Journal*, 12(1):69–71, February 1969.

[6] 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.

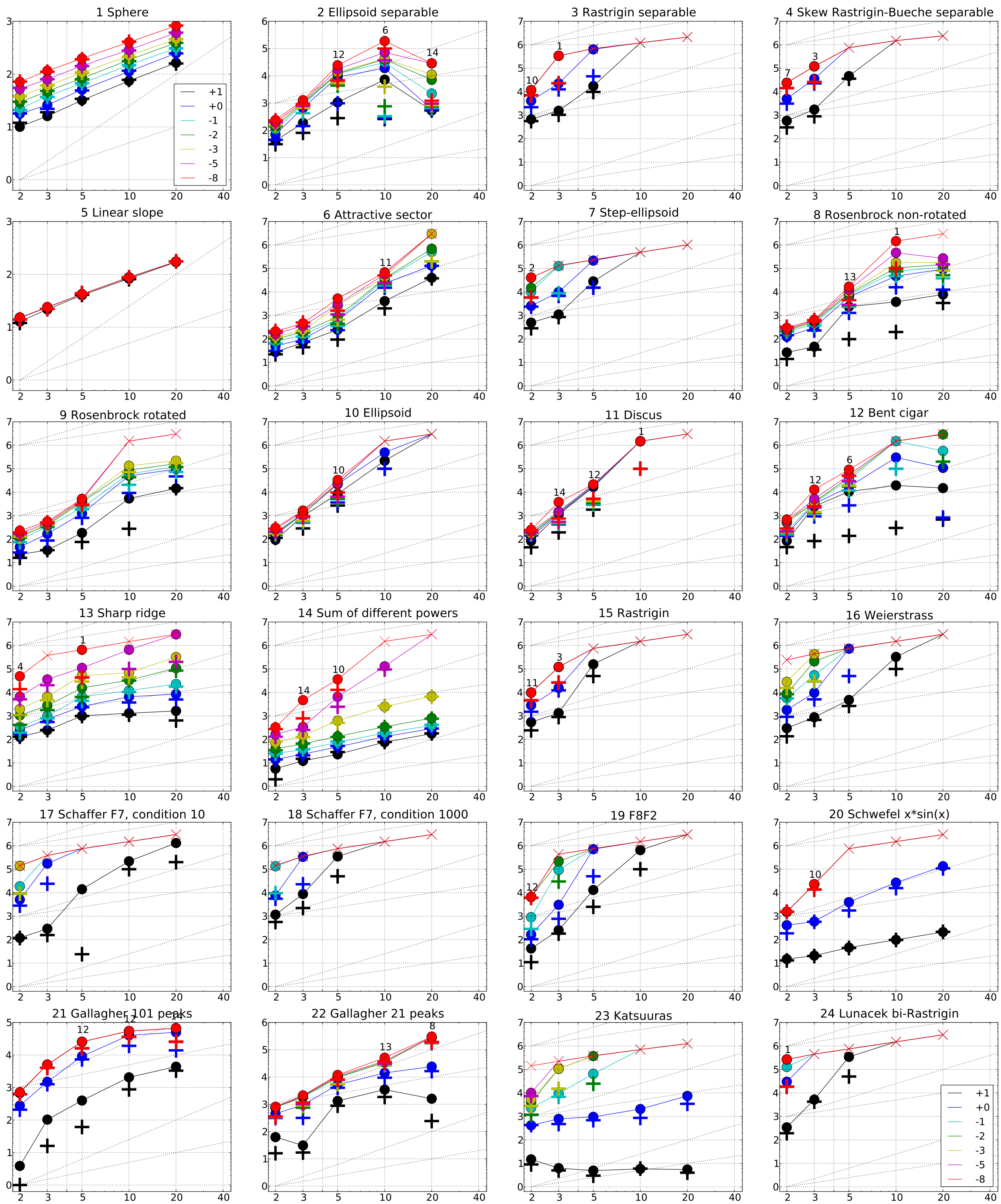

Figure 3: Expected Running Time (ERT, ●) **to reach** $f_{\mathrm{opt}} + \Delta f$ **and median number of function evaluations of successful trials** (+)**, shown for** $\Delta f = 10, 1, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-5}, 10^{-8}$ **(the exponent is given in the legend of** f_1 **and** f_{24}**) versus dimension in log-log presentation. The** $\mathrm{ERT}(\Delta f)$ **equals to** $\#\mathrm{FEs}(\Delta f)$ **divided by the number of successful trials, where a trial is successful if** $f_{\mathrm{opt}} + \Delta f$ **was surpassed during the trial. The** $\#\mathrm{FEs}(\Delta f)$ **are the total number of function evaluations while** $f_{\mathrm{opt}} + \Delta f$ **was not surpassed during the trial from all respective trials (successful and unsuccessful), and** f_{opt} **denotes the optimal function value. Crosses** ($\times$) **indicate the total number of function evaluations** $\#\mathrm{FEs}(-\infty)$**. Numbers above ERT-symbols indicate the number of successful trials. Annotated numbers on the ordinate are decimal logarithms. Additional grid lines show linear and quadratic scaling.**

Δf	f_1 in 5-D, N=15, mFE=261 #	ERT	10%	90%	RT_{succ}	f_1 in 20-D, N=15, mFE=901 #	ERT	10%	90%	RT_{succ}
10	15	3.2e1	3.0e1	3.5e1	3.2e1	15	1.6e2	1.6e2	1.7e2	1.6e2
1	15	5.1e1	4.5e1	5.6e1	5.1e1	15	2.5e2	2.4e2	2.6e2	2.5e2
1e−1	15	6.7e1	6.2e1	7.3e1	6.7e1	15	3.1e2	3.0e2	3.2e2	3.1e2
1e−3	15	1.1e2	9.8e1	1.1e2	1.1e2	15	4.6e2	4.5e2	4.7e2	4.6e2
1e−5	15	1.4e2	1.4e2	1.5e2	1.4e2	15	6.0e2	5.9e2	6.1e2	6.0e2
1e−8	15	2.0e2	1.9e2	2.1e2	2.0e2	15	8.2e2	8.1e2	8.4e2	8.2e2

Δf	f_2 in 5-D, N=15, mFE=50002 #	ERT	10%	90%	RT_{succ}	f_2 in 20-D, N=15, mFE=200001 #	ERT	10%	90%	RT_{succ}
10	15	1.1e3	5.9e2	1.6e3	1.1e3	15	5.3e2	5.1e2	5.5e2	5.3e2
1	14	9.0e3	4.3e3	1.4e4	8.9e3	15	6.0e2	5.9e2	6.2e2	6.0e2
1e−1	13	1.2e4	6.2e3	1.8e4	1.1e4	15	2.2e3	6.5e2	3.8e3	2.2e3
1e−3	13	1.4e4	7.6e3	2.0e4	1.2e4	15	1.1e4	8.3e2	2.2e4	1.1e4
1e−5	13	1.7e4	1.1e4	2.4e4	1.6e4	14	2.9e4	1.1e3	5.5e4	1.4e4
1e−8	12	2.4e4	1.7e4	3.2e4	2.0e4	14	2.9e4	1.4e3	5.5e4	1.5e4

Δf	f_3 in 5-D, N=15, mFE=48525 #	ERT	10%	90%	RT_{succ}	f_3 in 20-D, N=15, mFE=156740 #	ERT	10%	90%	RT_{succ}
10	13	1.7e4	1.1e4	2.4e4	1.6e4	0	*23e+1*	*17e+1*	*29e+1*	6.3e4
1	1	6.4e5	5.9e5	6.9e5	4.5e4	.	.	.	.	.
1e−1	0	*70e-1*	*20e-1*	*14e+0*	1.6e4	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	f_4 in 5-D, N=15, mFE=50004 #	ERT	10%	90%	RT_{succ}	f_4 in 20-D, N=15, mFE=194980 #	ERT	10%	90%	RT_{succ}
10	10	4.6e4	3.7e4	5.6e4	3.0e4	0	*20e+1*	*16e+1*	*31e+1*	4.5e4
1	0	*99e-1*	*50e-1*	*18e+0*	3.2e4	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	f_5 in 5-D, N=15, mFE=46 #	ERT	10%	90%	RT_{succ}	f_5 in 20-D, N=15, mFE=181 #	ERT	10%	90%	RT_{succ}
10	15	4.0e1	3.9e1	4.2e1	4.0e1	15	1.7e2	1.7e2	1.7e2	1.7e2
1	15	4.2e1	4.1e1	4.4e1	4.2e1	15	1.7e2	1.7e2	1.8e2	1.7e2
1e−1	15	4.2e1	4.1e1	4.4e1	4.2e1	15	1.7e2	1.7e2	1.8e2	1.7e2
1e−3	15	4.2e1	4.1e1	4.4e1	4.2e1	15	1.7e2	1.7e2	1.8e2	1.7e2
1e−5	15	4.2e1	4.1e1	4.4e1	4.2e1	15	1.7e2	1.7e2	1.8e2	1.7e2
1e−8	15	4.2e1	4.1e1	4.4e1	4.2e1	15	1.7e2	1.7e2	1.8e2	1.7e2

Δf	f_6 in 5-D, N=15, mFE=29922 #	ERT	10%	90%	RT_{succ}	f_6 in 20-D, N=15, mFE=200001 #	ERT	10%	90%	RT_{succ}
10	15	2.5e2	9.5e1	4.2e2	2.5e2	15	4.0e4	3.4e4	4.6e4	4.0e4
1	15	6.0e2	3.5e2	8.9e2	6.0e2	14	1.3e5	1.1e5	1.5e5	1.2e5
1e−1	15	6.8e2	4.1e2	9.4e2	6.8e2	5	5.1e5	4.6e5	5.6e5	1.8e5
1e−3	15	2.5e3	8.4e2	4.3e3	2.5e3	1	3.0e6	3.0e6	3.0e6	2.0e5
1e−5	15	3.0e3	1.2e3	4.8e3	3.0e3	0	*21e-2*	*16e-4*	*75e-2*	2.0e5
1e−8	15	5.2e3	1.8e3	8.5e3	5.2e3	.	.	.	.	.

Δf	f_7 in 5-D, N=15, mFE=16325 #	ERT	10%	90%	RT_{succ}	f_7 in 20-D, N=15, mFE=74760 #	ERT	10%	90%	RT_{succ}
10	6	2.8e4	2.4e4	3.3e4	1.3e4	0	*38e+1*	*26e+1*	*59e+1*	4.0e4
1	1	2.2e5	2.0e5	2.3e5	1.5e4	.	.	.	.	.
1e−1	0	*13e+0*	*20e-1*	*23e+0*	6.3e3	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	f_8 in 5-D, N=15, mFE=50001 #	ERT	10%	90%	RT_{succ}	f_8 in 20-D, N=15, mFE=200003 #	ERT	10%	90%	RT_{succ}
10	15	2.4e3	1.5e2	4.6e3	2.4e3	15	7.8e3	4.4e3	1.1e4	7.8e3
1	15	6.2e3	2.1e3	1.0e4	6.2e3	11	9.4e4	5.9e4	1.3e5	3.9e4
1e−1	15	7.5e3	3.5e3	1.2e4	7.5e3	11	1.1e5	7.3e4	1.5e5	5.8e4
1e−3	15	9.7e3	5.8e3	1.4e4	9.7e3	10	1.8e5	1.5e5	2.1e5	1.1e5
1e−5	15	1.2e4	7.6e3	1.8e4	1.2e4	9	2.7e5	2.5e5	2.9e5	1.6e5
1e−8	13	1.7e4	9.9e3	2.5e4	1.7e4	0	*87e-8*	*19e-8*	*40e-1*	2.0e5

Δf	f_9 in 5-D, N=15, mFE=26886 #	ERT	10%	90%	RT_{succ}	f_9 in 20-D, N=15, mFE=200001 #	ERT	10%	90%	RT_{succ}
10	15	1.8e2	7.7e1	2.9e2	1.8e2	15	1.4e4	1.3e4	1.6e4	1.4e4
1	15	1.2e3	8.1e2	1.7e3	1.2e3	12	9.6e4	7.0e4	1.2e5	8.4e4
1e−1	15	2.2e3	1.5e3	2.8e3	2.2e3	12	1.2e5	1.0e5	1.4e5	1.0e5
1e−3	15	4.2e3	3.3e3	5.3e3	4.2e3	12	2.2e5	2.1e5	2.3e5	1.8e5
1e−5	15	4.6e3	3.3e3	6.1e3	4.6e3	0	*35e-5*	*18e-5*	*40e-1*	2.0e5
1e−8	15	5.2e3	3.3e3	7.2e3	5.2e3	.	.	.	.	.

Δf	f_{10} in 5-D, N=15, mFE=50002 #	ERT	10%	90%	RT_{succ}	f_{10} in 20-D, N=15, mFE=200001 #	ERT	10%	90%	RT_{succ}
10	14	8.4e3	3.4e3	1.3e4	8.4e3	0	*27e+1*	*74e+0*	*38e+1*	2.0e5
1	11	2.2e4	1.3e4	3.0e4	2.1e4	.	.	.	.	.
1e−1	11	2.3e4	1.4e4	3.2e4	2.2e4	.	.	.	.	.
1e−3	11	2.3e4	1.4e4	3.2e4	2.2e4	.	.	.	.	.
1e−5	11	2.4e4	1.5e4	3.3e4	2.3e4	.	.	.	.	.
1e−8	10	3.3e4	2.3e4	4.3e4	2.6e4	.	.	.	.	.

Δf	f_{11} in 5-D, N=15, mFE=50001 #	ERT	10%	90%	RT_{succ}	f_{11} in 20-D, N=15, mFE=200001 #	ERT	10%	90%	RT_{succ}
10	13	1.7e4	9.1e3	2.5e4	1.3e4	0	*11e+1*	*75e+0*	*19e+1*	2.0e5
1	13	1.8e4	1.0e4	2.6e4	1.4e4	.	.	.	.	.
1e−1	12	2.0e4	1.1e4	2.8e4	1.5e4	.	.	.	.	.
1e−3	12	2.1e4	1.3e4	2.9e4	1.6e4	.	.	.	.	.
1e−5	12	2.1e4	1.3e4	2.9e4	1.6e4	.	.	.	.	.
1e−8	12	2.1e4	1.4e4	3.0e4	1.7e4	.	.	.	.	.

Δf	f_{12} in 5-D, N=15, mFE=50004 #	ERT	10%	90%	RT_{succ}	f_{12} in 20-D, N=15, mFE=200004 #	ERT	10%	90%	RT_{succ}
10	13	1.1e4	4.6e3	1.7e4	3.0e3	14	1.5e4	7.2e2	2.9e4	1.5e4
1	12	1.7e4	9.6e3	2.5e4	8.6e3	10	1.1e5	6.1e4	1.6e5	8.0e4
1e−1	10	3.4e4	2.4e4	4.3e4	1.8e4	4	5.7e5	4.6e5	6.7e5	1.2e5
1e−3	9	4.4e4	3.3e4	5.5e4	2.6e4	0	*70e-2*	*11e-3*	*20e-1*	1.0e5
1e−5	8	5.5e4	4.2e4	6.8e4	2.8e4	.	.	.	.	.
1e−8	6	9.1e4	7.5e4	1.1e5	4.0e4	.	.	.	.	.

Δf	f_{13} in 5-D, N=15, mFE=46360 #	ERT	10%	90%	RT_{succ}	f_{13} in 20-D, N=15, mFE=200019 #	ERT	10%	90%	RT_{succ}
10	15	1.0e3	8.0e2	1.2e3	1.0e3	15	1.6e3	1.0e3	2.3e3	1.6e3
1	15	2.5e3	2.0e3	3.0e3	2.5e3	15	8.5e3	5.6e3	1.1e4	8.5e3
1e−1	15	6.6e3	5.2e3	7.9e3	6.6e3	15	2.3e4	1.8e4	2.8e4	2.3e4
1e−3	8	5.2e4	4.2e4	6.1e4	2.5e4	7	3.2e5	2.7e5	3.7e5	1.7e5
1e−5	5	1.1e5	9.9e4	1.2e5	3.0e4	1	3.0e6	3.0e6	3.0e6	2.0e5
1e−8	1	6.5e5	6.5e5	6.6e5	4.1e4	0	*18e-4*	*13e-6*	*11e-3*	1.3e5

Δf	f_{14} in 5-D, N=15, mFE=50004 #	ERT	10%	90%	RT_{succ}	f_{14} in 20-D, N=15, mFE=200002 #	ERT	10%	90%	RT_{succ}
10	15	2.3e1	1.8e1	2.8e1	2.3e1	15	1.8e2	1.7e2	1.9e2	1.8e2
1	15	4.8e1	4.3e1	5.3e1	4.8e1	15	2.9e2	2.8e2	3.0e2	2.9e2
1e−1	15	7.4e1	6.8e1	8.0e1	7.4e1	15	4.0e2	3.8e2	4.1e2	4.0e2
1e−3	15	6.3e2	4.7e2	8.2e2	6.3e2	15	6.9e3	6.3e3	7.4e3	6.9e3
1e−5	14	6.5e3	2.7e3	1.0e4	6.2e3	0	*71e-6*	*39e-6*	*88e-6*	2.0e5
1e−8	10	3.6e4	2.7e4	4.8e4	2.9e4	.	.	.	.	.

Δf	f_{15} in 5-D, N=15, mFE=50004 #	ERT	10%	90%	RT_{succ}	f_{15} in 20-D, N=15, mFE=200019 #	ERT	10%	90%	RT_{succ}
10	4	1.6e5	1.4e5	1.8e5	3.8e4	0	*35e+1*	*24e+1*	*62e+1*	7.1e4
1	0	*16e+0*	*30e-1*	*34e+0*	2.8e4	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	f_{16} in 5-D, N=15, mFE=50004 #	ERT	10%	90%	RT_{succ}	f_{16} in 20-D, N=15, mFE=200018 #	ERT	10%	90%	RT_{succ}
10	15	4.8e3	3.0e3	6.6e3	4.8e3	0	*29e+0*	*19e+0*	*33e+0*	7.1e4
1	1	7.3e5	7.1e5	7.5e5	2.9e4	.	.	.	.	.
1e−1	0	*36e-1*	*11e-1*	*65e-1*	2.2e4	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	f_{17} in 5-D, N=15, mFE=50004 #	ERT	10%	90%	RT_{succ}	f_{17} in 20-D, N=15, mFE=200001 #	ERT	10%	90%	RT_{succ}
10	12	1.4e4	5.7e3	2.2e4	8.4e3	2	1.3e6	1.1e6	1.5e6	2.0e5
1	0	*57e-1*	*24e-1*	*14e+0*	2.0e4	0	*19e+0*	*97e-1*	*90e+0*	2.0e5
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	f_{18} in 5-D, N=15, mFE=50004 #	ERT	10%	90%	RT_{succ}	f_{18} in 20-D, N=15, mFE=200002 #	ERT	10%	90%	RT_{succ}
10	2	3.5e5	3.2e5	3.8e5	5.0e4	0	*97e+0*	*33e+0*	*52e+1*	2.0e5
1	0	*16e+0*	*96e-1*	*25e+0*	2.5e4	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	f_{19} in 5-D, N=15, mFE=50004 #	ERT	10%	90%	RT_{succ}	f_{19} in 20-D, N=15, mFE=200001 #	ERT	10%	90%	RT_{succ}
10	13	1.3e4	6.1e3	2.0e4	9.1e3	0	*33e+0*	*16e+0*	*10e+1*	2.0e5
1	1	7.1e5	6.8e5	7.5e5	5.0e4	.	.	.	.	.
1e−1	0	*38e-1*	*11e-1*	*13e+0*	1.6e4	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	f_{20} in 5-D, N=15, mFE=50004 #	ERT	10%	90%	RT_{succ}	f_{20} in 20-D, N=15, mFE=200016 #	ERT	10%	90%	RT_{succ}
10	15	4.6e1	4.3e1	4.9e1	4.6e1	15	2.1e2	2.0e2	2.2e2	2.1e2
1	15	4.0e3	2.7e3	5.3e3	4.0e3	12	1.3e5	1.0e5	1.6e5	1.2e5
1e−1	0	*47e-2*	*24e-2*	*67e-2*	1.8e4	0	*97e-2*	*85e-2*	*10e-1*	1.1e5
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	f_{21} in 5-D, N=15, mFE=50003 #	ERT	10%	90%	RT_{succ}	f_{21} in 20-D, N=15, mFE=200004 #	ERT	10%	90%	RT_{succ}
10	15	4.0e2	1.9e2	6.4e2	4.0e2	15	4.4e3	3.1e3	5.7e3	4.4e3
1	15	9.2e3	6.7e3	1.2e4	9.2e3	14	5.0e4	3.0e4	7.1e4	4.7e4
1e−1	12	2.5e4	1.8e4	3.3e4	2.3e4	14	6.6e4	4.1e4	9.1e4	6.3e4
1e−3	12	2.5e4	1.8e4	3.3e4	2.3e4	14	6.6e4	4.2e4	9.1e4	6.3e4
1e−5	12	2.6e4	1.8e4	3.3e4	2.3e4	14	6.6e4	4.2e4	9.4e4	6.3e4
1e−8	12	2.6e4	1.8e4	3.3e4	2.3e4	14	6.7e4	4.3e4	9.0e4	6.4e4

Δf	f_{22} in 5-D, N=15, mFE=42005 #	ERT	10%	90%	RT_{succ}	f_{22} in 20-D, N=15, mFE=200016 #	ERT	10%	90%	RT_{succ}
10	15	1.3e3	7.5e2	2.0e3	1.3e3	15	1.6e3	8.3e2	2.4e3	1.6e3
1	15	5.2e3	3.9e3	6.6e3	5.2e3	15	2.4e4	1.4e4	3.4e4	2.4e4
1e−1	15	9.5e3	6.3e3	1.3e4	9.5e3	8	2.8e5	2.4e5	3.2e5	1.6e5
1e−3	15	1.0e4	6.8e3	1.4e4	1.0e4	8	2.9e5	2.5e5	3.2e5	1.6e5
1e−5	15	1.1e4	7.8e3	1.4e4	1.1e4	8	2.9e5	2.6e5	3.3e5	1.6e5
1e−8	15	1.2e4	8.8e3	1.5e4	1.2e4	8	3.0e5	2.7e5	3.4e5	1.7e5

Δf	f_{23} in 5-D, N=15, mFE=28320 #	ERT	10%	90%	RT_{succ}	f_{23} in 20-D, N=15, mFE=86140 #	ERT	10%	90%	RT_{succ}
10	15	4.9e0	3.6e0	6.3e0	4.9e0	15	5.5e0	4.2e0	6.7e0	5.5e0
1	15	9.5e2	6.7e2	1.3e3	9.5e2	15	7.5e3	4.6e3	1.0e4	7.5e3
1e−1	5	6.5e4	5.9e4	7.1e4	2.0e4	0	*50e-2*	*29e-2*	*54e-2*	3.5e4
1e−3	0	*17e-2*	*43e-3*	*39e-2*	1.6e4	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	f_{24} in 5-D, N=15, mFE=50004 #	ERT	10%	90%	RT_{succ}	f_{24} in 20-D, N=15, mFE=200019 #	ERT	10%	90%	RT_{succ}
10	2	3.4e5	3.1e5	3.8e5	5.0e4	0	*37e+1*	*27e+1*	*44e+1*	1.0e5
1	0	*19e+0*	*93e-1*	*30e+0*	2.8e4	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Table 1: Shown are, for a given target difference to the optimal function value Δf: the number of successful trials (#); the expected running time to surpass $f_{\rm opt}+\Delta f$ (ERT, see Figure 3); the 10%-tile and 90%-tile of the bootstrap distribution of ERT; the average number of function evaluations in successful trials or, if none was successful, as last entry the median number of function evaluations to reach the best function value ($\mathbf{RT_{succ}}$). If $f_{\rm opt}+\Delta f$ was never reached, figures in *italics* denote the best achieved Δf-value of the median trial and the 10% and 90%-tile trial. Furthermore, N denotes the number of trials, and mFE denotes the maximum of number of function evaluations executed in one trial. See Figure 3 for the names of functions.

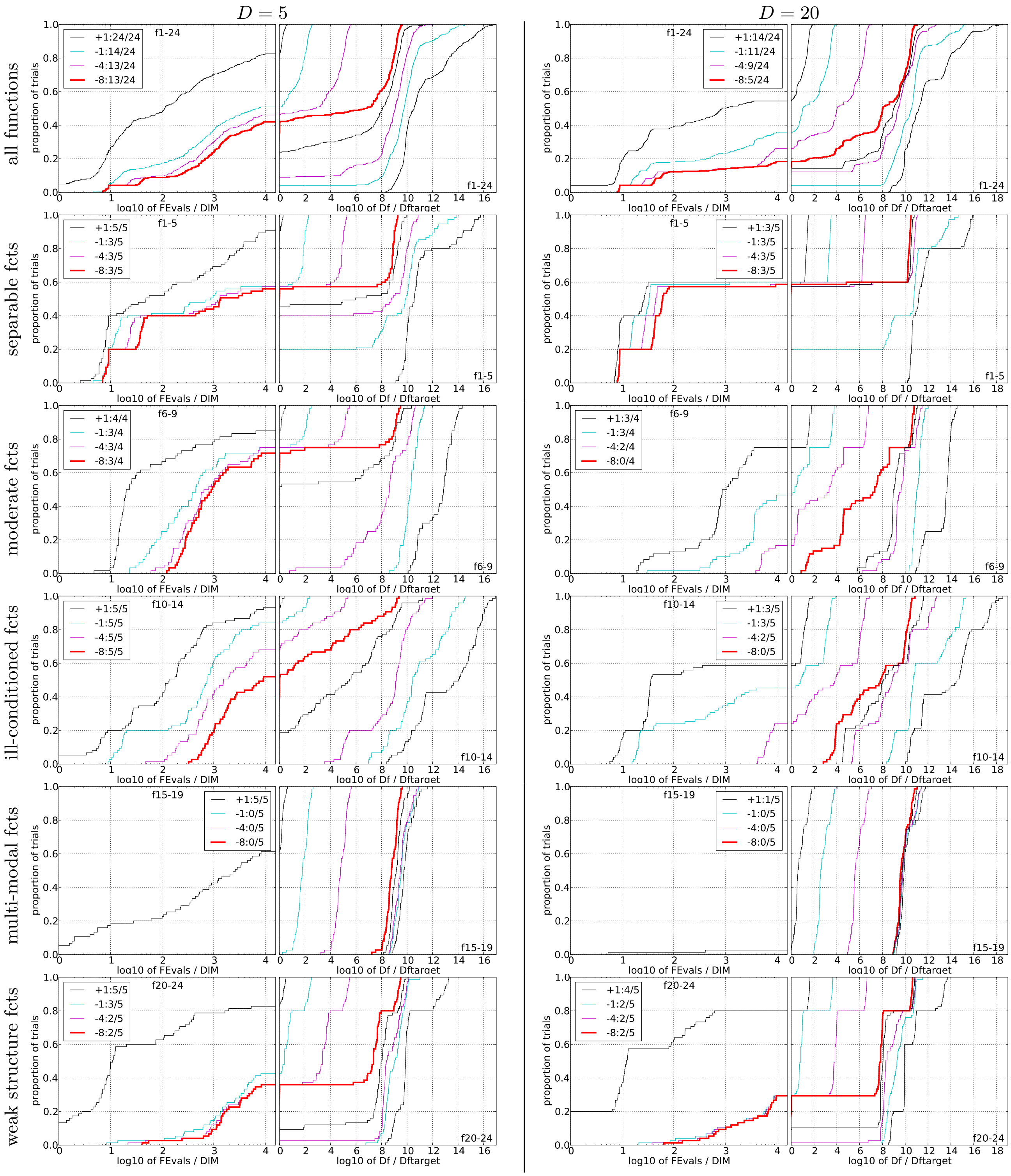

Figure 4: Empirical cumulative distribution functions (ECDFs), plotting the fraction of trials versus running time (left) or Δf. Left subplots: ECDF of the running time (number of function evaluations), divided by search space dimension D, to fall below $f_{\rm opt} + \Delta f$ with $\Delta f = 10^k$, where k is the first value in the legend. Right subplots: ECDF of the best achieved Δf divided by 10^k (upper left lines in continuation of the left subplot), and best achieved Δf divided by 10^{-8} for running times of $D, 10\,D, 100\,D\ldots$ function evaluations (from right to left cycling black-cyan-magenta). Top row: all results from all functions; second row: separable functions; third row: misc. moderate functions; fourth row: ill-conditioned functions; fifth row: multi-modal functions with adequate structure; last row: multi-modal functions with weak structure. The legends indicate the number of functions that were solved in at least one trial. FEvals denotes number of function evaluations, D and DIM denote search space dimension, and Δf and Df denote the difference to the optimal function value.