

Benchmarking Separable Natural Evolution Strategies on the Noiseless and Noisy Black-box Optimization Testbeds

Tom Schaul

Courant Institute of Mathematical Sciences, New York University
Broadway 715, New York, USA
schaul@cims.nyu.edu

ABSTRACT

Natural Evolution Strategies (NES) are a recent member of the class of real-valued optimization algorithms that are based on adapting search distributions. *Separable* NES (SNES) are a variant of NES that scale linearly with problem dimension and are particularly appropriate for large, separable problems. This report provides the the most extensive empirical results on that algorithm to date, on both the noise-free and noisy BBOB testbeds.

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

Evolution Strategies, Natural Gradient, Benchmarking

1. INTRODUCTION

Evolution strategies (ES), in contrast to traditional evolutionary algorithms, aim at repeating the type of mutation that led to those good individuals. We can characterize those mutations by an explicitly parameterized *search distribution* from which new candidate samples are drawn, akin to estimation of distribution algorithms (EDA). Covariance matrix adaptation ES (CMA-ES [8]) innovated the field by introducing a parameterization that includes the full covariance matrix, allowing them to solve highly non-separable problems.

A more recent variant, *natural evolution strategies* (NES [16, 4, 14, 15]) aims at a higher level of generality, providing a procedure to update the search distribution’s parameters for

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

GECCO’12, July 7–11, 2012, Philadelphia, USA.

Copyright 2012 ACM 978-1-4503-0073-5/10/07 ...\$10.00.

any type of distribution, by ascending the gradient towards higher expected fitness. Further, it has been shown [11, 10] that following the *natural gradient* to adapt the search distribution is highly beneficial, because it appropriately normalizes the update step with respect to its uncertainty and makes the algorithm scale-invariant.

Separable NES (SNES [13]), an instantiation of NES designed for when the problem dimensionality is too high for using a full covariance matrix parameterization, instead using only the diagonal for the search distribution. It is thus quite similar to sep-CMA-ES [9]. Given the relatively small problem dimensions of the BBOB benchmarks, and the fact that many are non-separable, SNES is not the most appropriate NES variants for this particular task. In this report, we retain the original formulation of SNES (including all parameter settings, except for an added stopping criterion) and describe the empirical performance on all 54 benchmark functions (both noise-free and noisy) of the BBOB 2012 workshop.

2. NATURAL EVOLUTION STRATEGIES

Natural evolution strategies (NES) maintain a search distribution π and adapt the distribution parameters θ by following the *natural gradient* [1] of expected fitness J , that is, maximizing

$$J(\theta) = \mathbb{E}_\theta[f(\mathbf{z})] = \int f(\mathbf{z}) \pi(\mathbf{z} | \theta) d\mathbf{z}$$

Just like their close relative CMA-ES [8], NES algorithms are invariant under monotone transformations of the fitness function and linear transformations of the search space. Each iteration the algorithm produces n samples $\mathbf{z}_i \sim \pi(\mathbf{z} | \theta)$, $i \in \{1, \dots, n\}$, i.i.d. from its search distribution, which is parameterized by θ . The gradient w.r.t. the parameters θ can be rewritten (see [16]) as

$$\nabla_\theta J(\theta) = \nabla_\theta \int f(\mathbf{z}) \pi(\mathbf{z} | \theta) d\mathbf{z} = \mathbb{E}_\theta [f(\mathbf{z}) \nabla_\theta \log \pi(\mathbf{z} | \theta)]$$

from which we obtain a Monte Carlo estimate

$$\nabla_\theta J(\theta) \approx \frac{1}{n} \sum_{i=1}^n f(\mathbf{z}_i) \nabla_\theta \log \pi(\mathbf{z}_i | \theta)$$

of the search gradient. The key step then consists in replacing this gradient by the natural gradient defined as $\mathbf{F}^{-1} \nabla_\theta J(\theta)$ where $\mathbf{F} = \mathbb{E} [\nabla_\theta \log \pi(\mathbf{z} | \theta) \nabla_\theta \log \pi(\mathbf{z} | \theta)^\top]$ is the Fisher information matrix. The search distribution is iteratively

updated using natural gradient ascent

$$\theta \leftarrow \theta + \eta \mathbf{F}^{-1} \nabla_{\theta} J(\theta)$$

with learning rate parameter η .

2.1 Separable NES

While the NES formulation is applicable to arbitrary parameterizable search distributions [16, 10], the most common variant employs multinormal search distributions. For that case, two helpful techniques were introduced in [4], namely an exponential parameterization of the covariance matrix, which guarantees positive-definiteness, and a novel method for changing the coordinate system into a “natural” one, which makes the algorithm computationally efficient. The resulting algorithm, NES with a multivariate Gaussian search distribution and using both these techniques is called *xNES*. Building on this work, a separable variant that parameterizes only the diagonal of the search distribution was introduced in [13]. The pseudocode is given in Algorithm 1.

Algorithm 1: Separable NES (SNES)

```

input:  $f, \mu_{\text{init}}$ 
initialize  $\mu \leftarrow \mu_{\text{init}}$ 
            $\sigma \leftarrow \mathbf{1}$ 

repeat
  for  $k = 1 \dots n$  do
    draw sample  $\mathbf{s}_k \sim \mathcal{N}(0, \mathbf{I})$ 
     $\mathbf{z}_k \leftarrow \mu + \sigma \mathbf{s}_k$ 
    evaluate the fitness  $f(\mathbf{z}_k)$ 
  end

  sort  $\{(\mathbf{s}_k, \mathbf{z}_k)\}$  with respect to  $f(\mathbf{z}_k)$ 
  and assign utilities  $u_k$  to each sample

  compute gradients  $\nabla_{\mu} J \leftarrow \sum_{k=1}^n u_k \cdot \mathbf{s}_k$ 
                    $\nabla_{\sigma} J \leftarrow \sum_{k=1}^n u_k \cdot (\mathbf{s}_k^2 - 1)$ 

  update parameters  $\mu \leftarrow \mu + \eta_{\mu} \cdot \sigma \cdot \nabla_{\mu} J$ 
                    $\sigma \leftarrow \sigma \cdot \exp(\eta_{\sigma} / 2 \cdot \nabla_{\sigma} J)$ 
until stopping criterion is met

```

Table 1: Default parameter values for xNES (including the utility function and adaptation sampling) as a function of problem dimension d .

parameter	default value
n	$4 + \lceil 3 \log(d) \rceil$
$\eta_{\sigma} = \eta_{\mathbf{B}}$	$\frac{3 + \log(d)}{5\sqrt{d}}$
u_k	$\frac{\max(0, \log(\frac{n}{2} + 1) - \log(k))}{\sum_{j=1}^n \max(0, \log(\frac{n}{2} + 1) - \log(j))} - \frac{1}{n}$

3. EXPERIMENTAL SETTINGS

We use identical default hyper-parameter values for all benchmarks (both noisy and noise-free functions), which are taken from [13, 10]. Table 1 summarizes all the hyper-parameters used.

In addition, we make use of the provided target fitness f_{opt} to trigger *independent* algorithm restarts¹, using a simple ad-hoc procedure: If the log-progress during the past $1000d$ evaluations is too small, i.e., if

$$\log_{10} \left| \frac{f_{\text{opt}} - f_t}{f_{\text{opt}} - f_{t-1000d}} \right| < (r+2)^2 \cdot m^{3/2} \cdot [\log_{10} |f_{\text{opt}} - f_t| + 8]$$

where m is the remaining budget of evaluations divided by $1000d$, f_t is the best fitness encountered until evaluation t and r is the number of restarts so far. The total budget is $10^5 d^{3/2}$ evaluations.

Implementations of this and other NES algorithm variants are available in Python through the PyBrain machine learning library [12], as well as in other languages at www.idsia.ch/~tom/nas.html.

4. CPU TIMING

A timing experiment was performed to determine the CPU-time per function evaluation, and how it depends on the problem dimension. For each dimension, the algorithm was restarted with a maximum budget of $10000/d$ evaluations, until at least 30 seconds had passed.

Our SNES implementation (in Python, stand-alone), running on an Intel Xeon with 2.67GHz, required an average time of 0.15, 0.16, 0.15, 0.15, 0.16, 0.18, 0.23, 0.38 milliseconds per function evaluation for dimensions 2, 5, 10, 20, 40, 80, 160, 320 respectively. Not that within that cost, the majority of computation is taken up by the function evaluations themselves, which last 0.11, 0.11, 0.12, 0.12, 0.12, 0.14, 0.17, 0.28 milliseconds each, for the same range of dimensions respectively.

5. RESULTS

Results of SNES on the noiseless testbed (from experiments according to [5] on the benchmark functions given in [2, 6]) are presented in Figures 1, 3 and 5 and in Tables 2 and 4.

Similarly, results of SNES on the testbed of noisy functions (from experiments according to [5] on the benchmark functions given in [3, 7]) are presented in Figures 2, 4 and 5 and in Tables 3, and 4.

6. DISCUSSION

Given the composition of the testbeds, with many non-separable problems, it does not come as a surprise that SNES only performs well on a subset of the benchmarks (e.g., functions 1, 2, 3, 5, 21, 22, 101, 102, 103, 107, 109, 128, 130). According to Table 3, the only conditions where SNES significantly outperforms *all* algorithms from the BBOB2009 competition in dimension 20 are on functions f_{109} and f_{124} (during the early phase), and f_{110} in dimension 5. The SNES parameters were chosen for large unimodal, separable benchmarks, but we still observe a graceful decay in performance when using the algorithm on multimodal and noisy benchmarks as well. As expected, the highly non-separable problems become too hard with the separability assumption.

¹It turns out that this use of f_{opt} is technically not permitted by the BBOB guidelines, so strictly speaking a different restart strategy should be employed, for example the one described in [10].

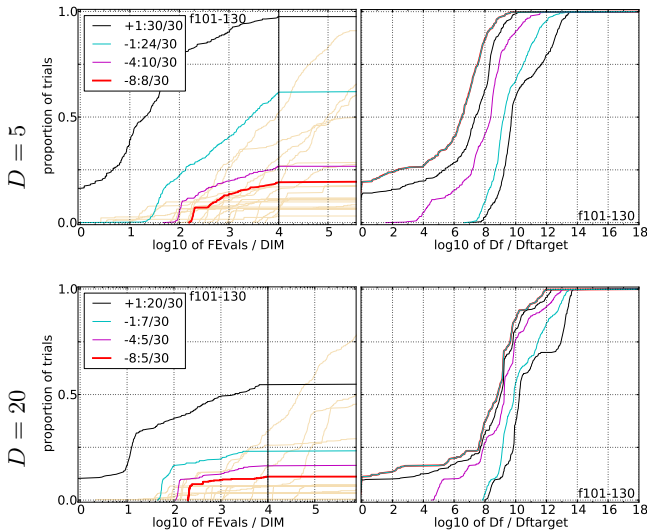


Figure 4: Empirical cumulative distribution functions (ECDFs) of the 30 noisy benchmark functions. Plotted is the fraction of trials versus running time (left subplots) or versus Δf (right subplots) (see Figure 3 for details).

Interestingly, from Table 4 we can see that in the early phase of convergence ($\#FEs \approx 100d$), SNES is still performing well, with a median loss ratio of only 2 to 7 across all benchmarks taken together. So it appears that initial progress can be made with SNES even on non-separable functions, and that estimating the full covariance becomes more important later on for fine-tuning.

Acknowledgements

The author wants to thank the organizers of the BBOB workshop for providing such a well-designed benchmark setup, and especially such high-quality post-processing utilities.

This work was funded in part through AFR postdoc grant number 2915104, of the National Research Fund Luxembourg.

7. REFERENCES

- [1] S. I. Amari. Natural Gradient Works Efficiently in Learning. *Neural Computation*, 10:251–276, 1998.
- [2] 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.
- [3] S. Finck, N. Hansen, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2010: Presentation of the noisy functions. Technical Report 2009/21, Research Center PPE, 2010.
- [4] T. Glasmachers, T. Schaul, Y. Sun, D. Wierstra, and J. Schmidhuber. Exponential Natural Evolution Strategies. In *Genetic and Evolutionary Computation Conference (GECCO)*, Portland, OR, 2010.
- [5] N. Hansen, A. Auger, S. Finck, and R. Ros. Real-parameter black-box optimization benchmarking

Table 4: ERT loss ratio compared to the respective best result from BBOB-2009 for budgets given in the first column (see also Figure 5). The last row RL_{US}/D gives the number of function evaluations in unsuccessful runs divided by dimension. Shown are the smallest, 10%-ile, 25%-ile, 50%-ile, 75%-ile and 90%-ile value (smaller values are better). The ERT Loss ratio equals to one for the respective best algorithm from BBOB-2009. Typical median values are between ten and hundred.

f_1-f_{24} in 5-D, $\max FE/D=200320$						
#FEs/D	best	10%	25%	med	75%	90%
2	1.5	2.4	4.8	7.0	9.2	10
10	2.1	2.3	2.7	3.4	4.6	14
100	0.93	2.0	4.3	7.1	14	42
1e3	1.3	3.9	7.6	29	65	80
1e4	5.9	7.9	13	69	2.5e2	4.4e2
1e5	5.2	14	38	1.2e2	1.4e3	2.1e3
1e6	12	15	33	1.8e2	5.5e3	1.2e4
RL_{US}/D	2e5	2e5	2e5	2e5	2e5	2e5
f_1-f_{24} in 20-D, $\max FE/D=400110$						
#FEs/D	best	10%	25%	med	75%	90%
2	1.0	1.9	11	31	40	40
10	0.79	1.7	2.3	3.5	5.9	27
100	0.64	1.3	2.6	5.8	31	71
1e3	1.1	4.0	7.4	22	76	2.6e2
1e4	6.1	9.0	23	83	1.3e2	7.6e2
1e5	12	24	43	2.2e2	6.5e2	2.0e3
1e6	12	15	1.9e2	5.9e2	4.6e3	1.7e4
1e7	12	51	3.5e2	3.6e3	4.2e4	1.4e5
RL_{US}/D	3e5	4e5	4e5	4e5	4e5	4e5
$f_{101}-f_{130}$ in 5-D, $\max FE/D=10152$						
#FEs/D	best	10%	25%	med	75%	90%
2	0.86	5.6	7.1	10	10	10
10	1.3	1.9	2.4	5.1	16	50
100	0.63	0.98	1.7	2.8	9.9	2.7e2
1e3	0.47	1.1	1.2	2.1	11	2.5e3
1e4	0.42	1.4	3.1	6.3	35	2.5e4
RL_{US}/D	1e4	1e4	1e4	1e4	1e4	1e4
$f_{101}-f_{130}$ in 20-D, $\max FE/D=10047$						
#FEs/D	best	10%	25%	med	75%	90%
2	1.0	2.6	29	40	40	40
10	0.58	0.68	1.0	4.2	2.0e2	2.0e2
100	0.62	1.1	1.3	2.1	16	2.0e3
1e3	0.19	1.0	2.8	7.0	20	2.0e4
1e4	0.75	4.5	6.6	18	54	2.0e5
1e5	2.8	5.4	32	68	1.7e2	1.0e6
RL_{US}/D	1e4	1e4	1e4	1e4	1e4	1e4

2012: Experimental setup. Technical report, INRIA, 2012.

- [6] 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.
- [7] N. Hansen, S. Finck, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2009: Noisy functions definitions. Technical Report RR-6869, INRIA, 2009. Updated February 2010.

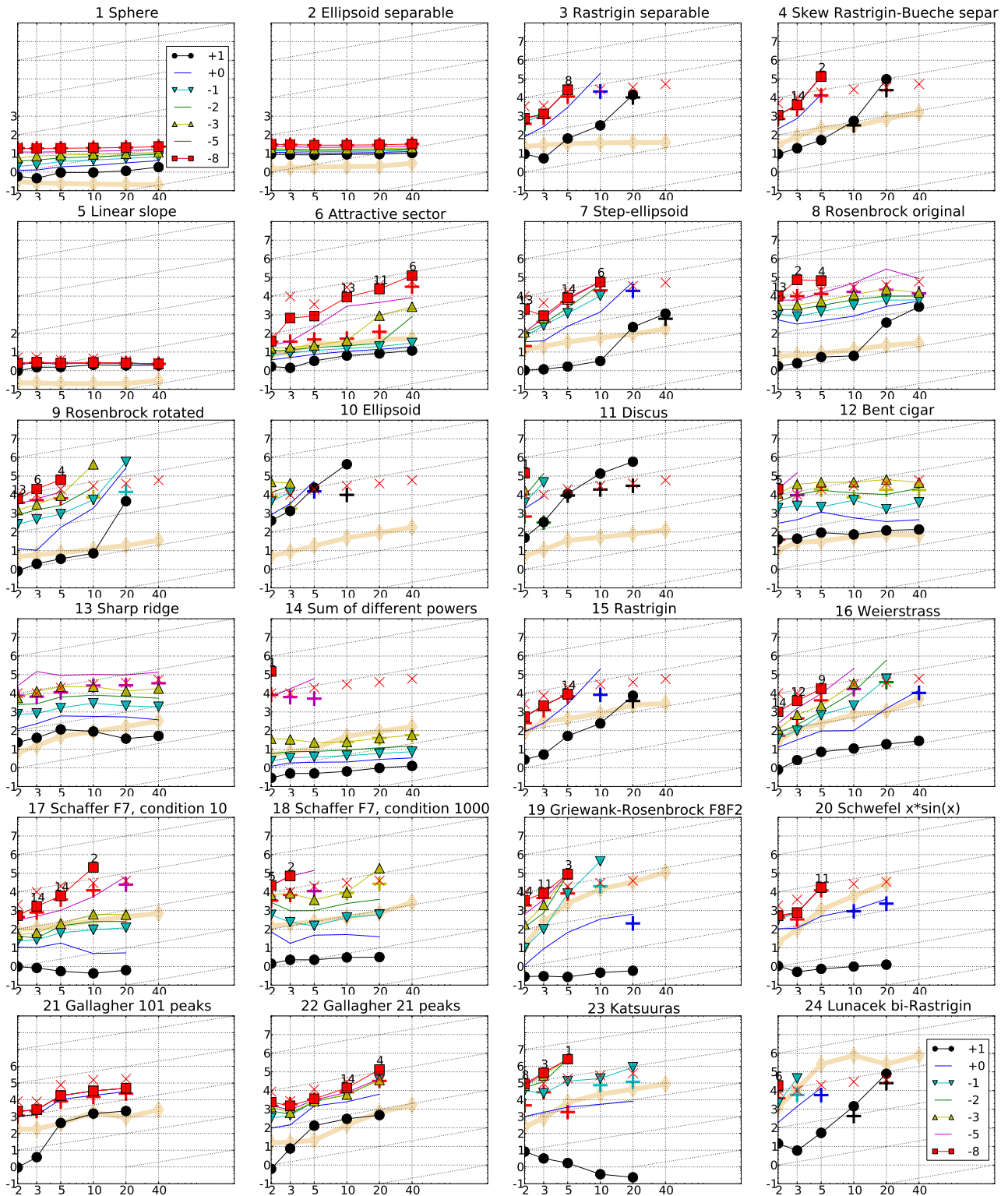


Figure 1: Expected number of f -evaluations (ERT, with lines, see legend) to reach $f_{\text{opt}} + \Delta f$, median number of f -evaluations to reach the most difficult target that was reached at least once (+) and maximum number of f -evaluations in any trial (\times), all divided by dimension and plotted as \log_{10} values versus dimension. Shown are $\Delta f = 10^{\{1,0,-1,-2,-3,-5,-8\}}$. Numbers above ERT-symbols indicate the number of successful trials. The light thick line with diamonds indicates the respective best result from BBOB-2009 for $\Delta f = 10^{-8}$. Horizontal lines mean linear scaling, slanted grid lines depict quadratic scaling.

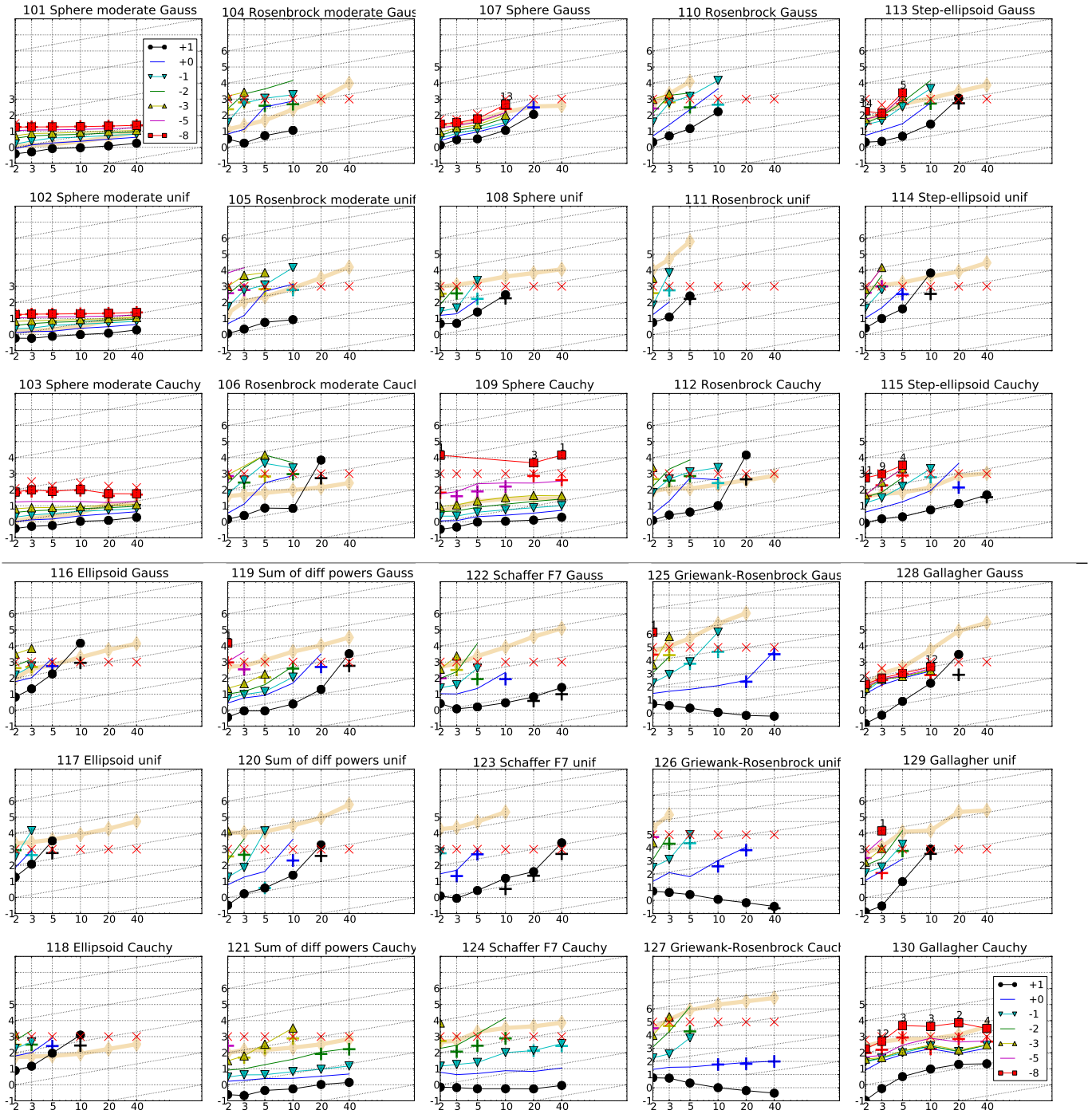


Figure 2: Expected number of f -evaluations (ERT, with lines, see legend) to reach $f_{\text{opt}} + \Delta f$, median number of f -evaluations to reach the most difficult target that was reached at least once (+) and maximum number of f -evaluations in any trial (\times), all divided by dimension and plotted as \log_{10} values versus dimension. Shown are $\Delta f = 10^{\{1,0,-1,-2,-3,-5,-8\}}$. Numbers above ERT-symbols indicate the number of successful trials. The light thick line with diamonds indicates the respective best result from BBOB-2009 for $\Delta f = 10^{-8}$. Horizontal lines mean linear scaling, slanted grid lines depict quadratic scaling.

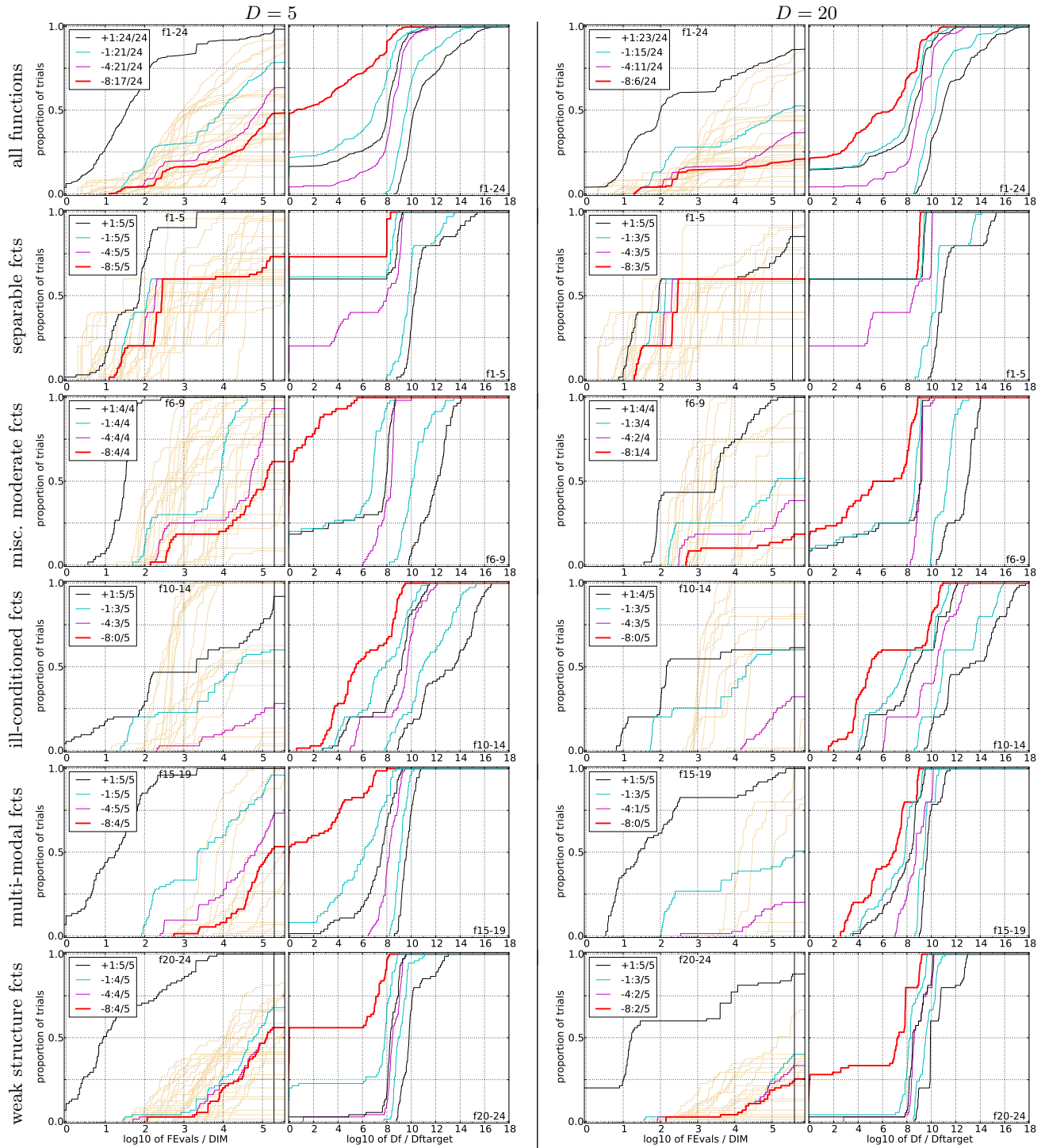


Figure 3: Empirical cumulative distribution functions (ECDFs), plotting the fraction of trials with an outcome not larger than the respective value on the x -axis. Left subplots: ECDF of number of function evaluations (FEvals) divided by search space dimension D , to fall below $f_{\text{opt}} + \Delta f$ 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^{-8} for running times of $D, 10D, 100D, \dots$ function evaluations (from right to left cycling black-cyan-magenta). The thick red line represents the most difficult target value $f_{\text{opt}} + 10^{-8}$. Legends indicate the number of functions that were solved in at least one trial. Light brown lines in the background show ECDFs for $\Delta f = 10^{-8}$ of all algorithms benchmarked during BBOB-2009.

