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 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 $\mathbf{z}_i \sim \pi(\mathbf{z}|\theta)$, $i \in \{1, \ldots, n\}$, 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

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

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 $\mathbf{F} = \mathbb{E}\left[\nabla_\theta \log \pi\left(\mathbf{z}|\theta\right) \nabla_\theta \log \pi\left(\mathbf{z}|\theta\right)^\top\right]$ is the Fisher information matrix. The search distribution is iteratively

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

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, $\boldsymbol{\mu}_{\text{init}}$

initialize $\begin{array}{lcl} \boldsymbol{\mu} & \leftarrow & \boldsymbol{\mu}_{\text{init}} \\ \boldsymbol{\sigma} & \leftarrow & \mathbf{1} \end{array}$

repeat

  for $k = 1 \ldots n$ **do**

    draw sample $\mathbf{s}_k \sim \mathcal{N}(0, \mathbb{I})$

    $\mathbf{z}_k \leftarrow \boldsymbol{\mu} + \boldsymbol{\sigma}\mathbf{s}_k$

    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 $\begin{array}{l} \nabla_{\boldsymbol{\mu}} J \leftarrow \sum_{k=1}^{n} u_k \cdot \mathbf{s}_k \\ \nabla_{\boldsymbol{\sigma}} J \leftarrow \sum_{k=1}^{n} u_k \cdot (\mathbf{s}_k^2 - 1) \end{array}$

  update parameters $\begin{array}{l} \boldsymbol{\mu} \leftarrow \boldsymbol{\mu} + \eta_{\boldsymbol{\mu}} \cdot \boldsymbol{\sigma} \cdot \nabla_{\boldsymbol{\mu}} J \\ \boldsymbol{\sigma} \leftarrow \boldsymbol{\sigma} \cdot \exp(\eta_{\boldsymbol{\sigma}}/2 \cdot \nabla_{\boldsymbol{\sigma}} J) \end{array}$

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	$4 + \lfloor 3 \log(d) \rfloor$
$\eta_\sigma = \eta_{\mathbf{B}}$	$\dfrac{3 + \log(d)}{5\sqrt{d}}$
u_k	$\dfrac{\max\left(0, \log(\frac{n}{2} + 1) - \log(k)\right)}{\sum_{j=1}^{n} \max\left(0, \log(\frac{n}{2} + 1) - \log(j)\right)} - \dfrac{1}{n}$

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 restarts[1], 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/nes.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.

[1]It 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].

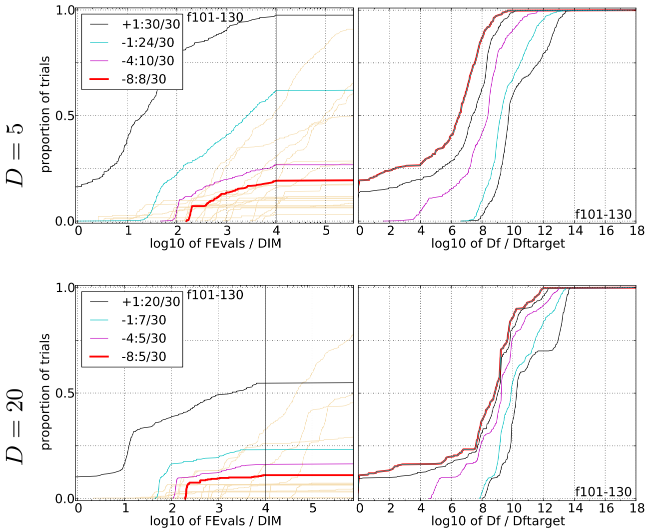

Figure 4: Empirical cumulative distribution functions (ECDFs) of the 30 noisy benchmark functions. Plotted is the fraction of trials versus running time (left subplots) or versus Δf (right subplots) (see Figure 3 for details).

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.

Acknowlegements

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 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.

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 $\mathrm{RL_{US}}/\mathrm{D}$ 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.

	$f1$–$f24$ in 5-D, maxFE/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
$\mathrm{RL_{US}}$/D	2e5	2e5	2e5	2e5	2e5	2e5
	$f1$–$f24$ in 20-D, maxFE/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
$\mathrm{RL_{US}}$/D	3e5	4e5	4e5	4e5	4e5	4e5
	$f101$–$f130$ in 5-D, maxFE/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
$\mathrm{RL_{US}}$/D	1e4	1e4	1e4	1e4	1e4	1e4
	$f101$–$f130$ in 20-D, maxFE/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
$\mathrm{RL_{US}}$/D	1e4	1e4	1e4	1e4	1e4	1e4

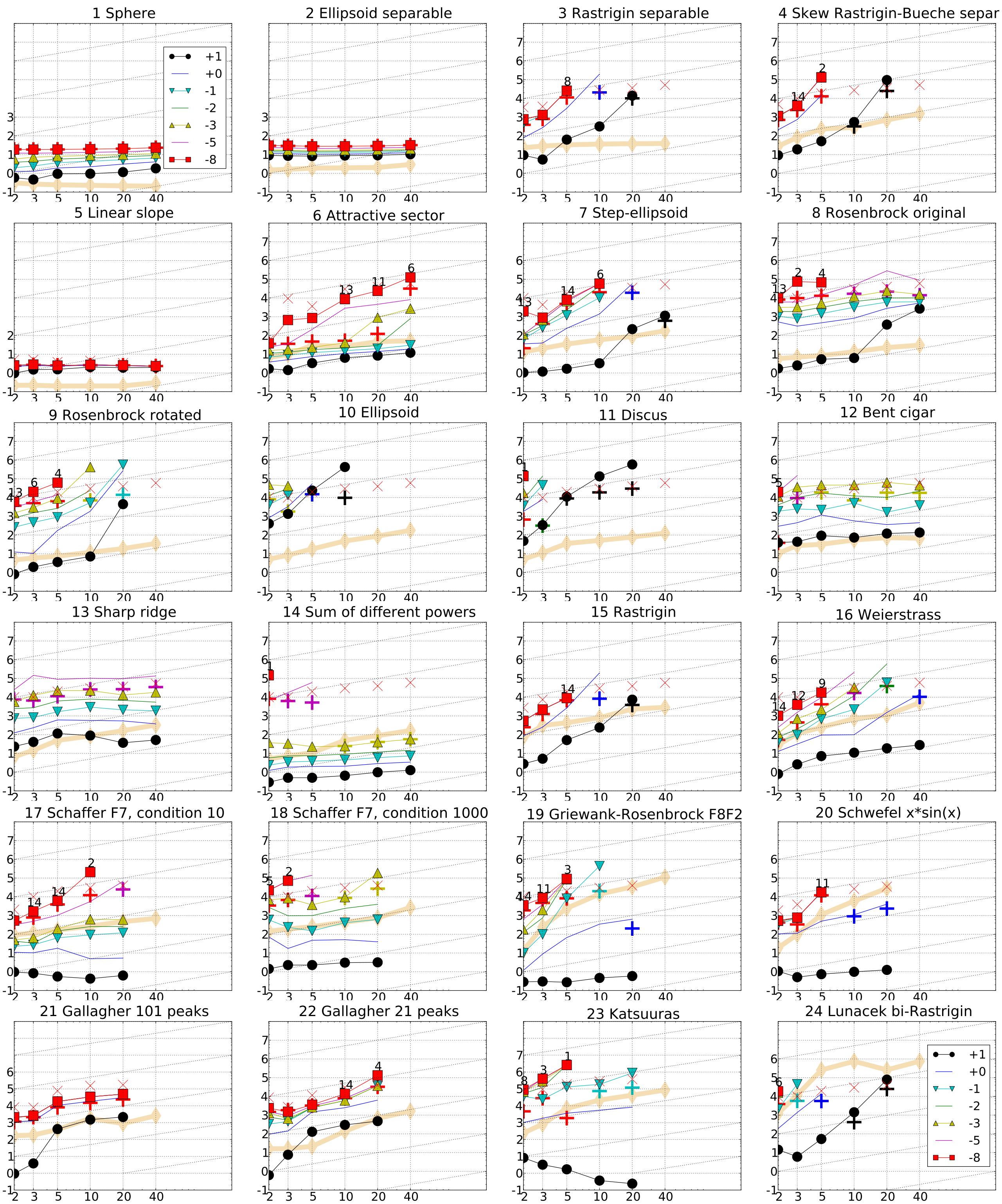

Figure 1: Expected number of f-evaluations (ERT, with lines, see legend) to reach $f_{\mathrm{opt}} + \Delta f$, median number of f-evaluations to reach the most difficult target that was reached at least once (+) and maximum number of f-evaluations in any trial (×), all divided by dimension and plotted 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.

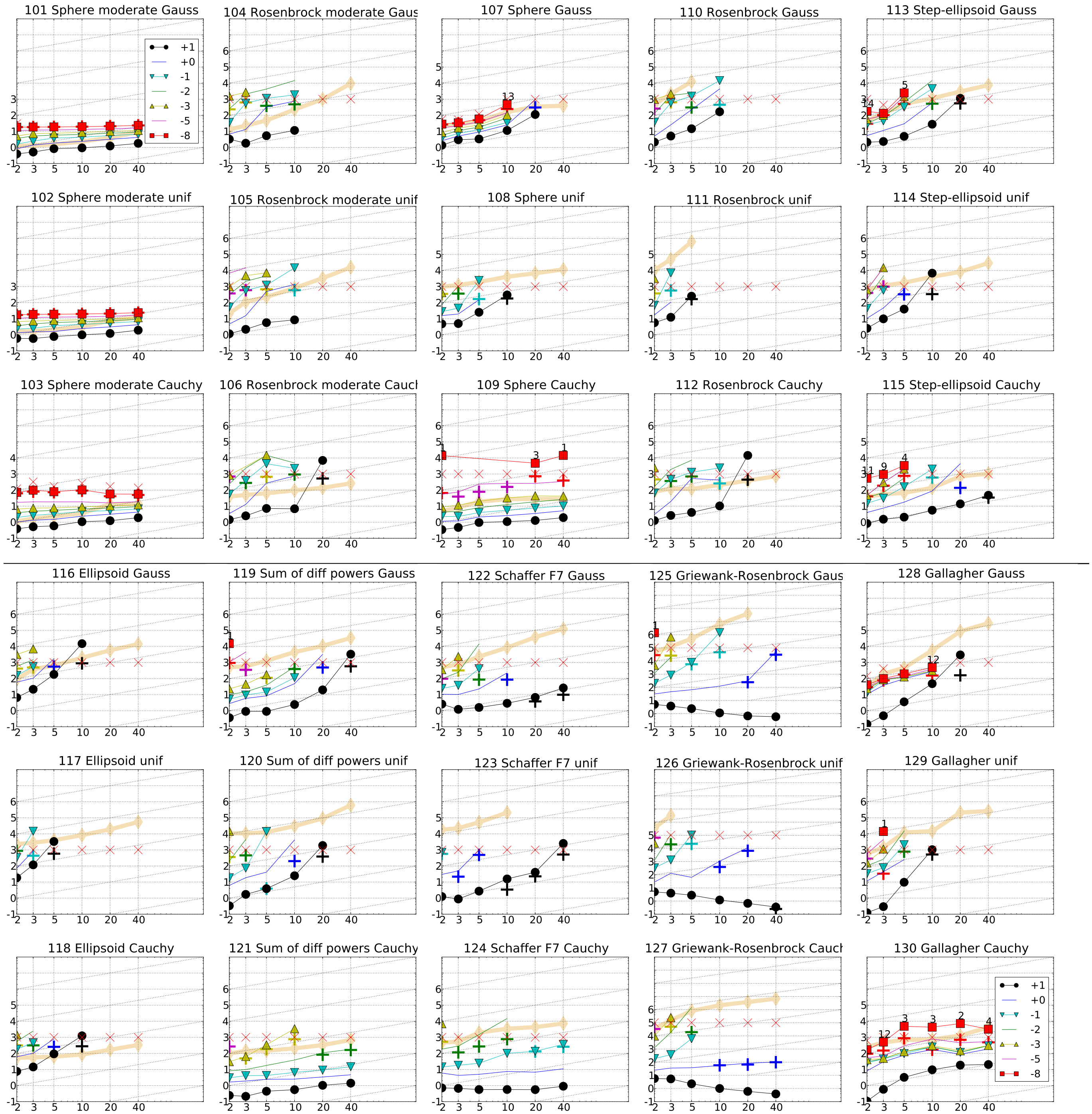

Figure 2: Expected number of f-evaluations (ERT, with lines, see legend) to reach $f_{\mathrm{opt}} + \Delta f$, median number of f-evaluations to reach the most difficult target that was reached at least once (+) and maximum number of f-evaluations in any trial ($\times$), all divided by dimension and 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.

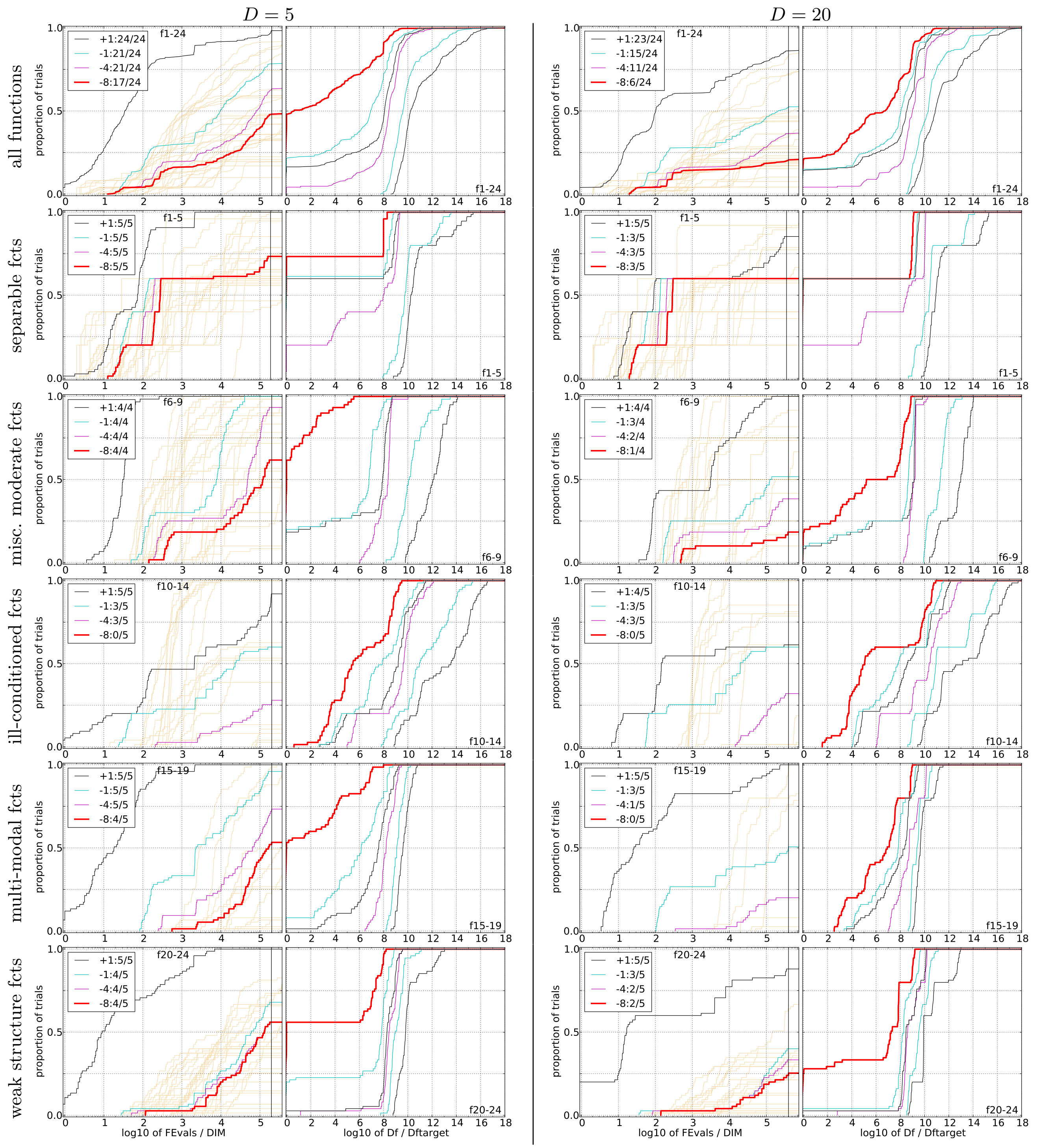

Figure 3: Empirical cumulative distribution functions (ECDFs), plotting the fraction of trials with an outcome not larger than the respective value on the x-axis. Left subplots: ECDF of number of function evaluations (FEvals) divided by search space dimension D, to fall below $f_{\rm opt} + \Delta f$ 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^{-8} for running times of $D, 10\,D, 100\,D, \ldots$ function evaluations (from right to left cycling black-cyan-magenta). The thick red line represents the most difficult target value $f_{\rm opt} + 10^{-8}$. 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.

5-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
	4.3(3)	7.9(3)	15(5)	33(5)	51(4)	68(5)	15/15
f_2	83	87	88	90	92	94	15/15
	5.0(1)	5.9(0.9)	7.0(0.9)	9.3(0.6)	11(0.6)	13(0.6)	15/15
f_3	716	1622	1637	1646	1650	1654	15/15
	4.4(7)	91(146)	785(788)	781(884)	779(724)	778(688)	8/15
f_4	809	1633	1688	1817	1886	1903	15/15
	3.2(6)	443(389)	3934(4283)	3654(3975)	3521(3907)	3489(3798)	2/15
f_5	10	10	10	10	10	10	15/15
	7.8(3)	12(4)	12(4)	12(4)	12(4)	12(4)	15/15
f_6	114	214	281	580	1038	1332	15/15
	1.5(1)	1.7(0.6)	2.2(0.5)	2.0(0.6)	10(15)	23(51)	15/15
f_7	24	324	1171	1572	1572	1597	15/15
	3.5(3)	37(76)	51(56)	212(256)	212(256)	237(253)	14/15
f_8	73	273	336	391	410	422	15/15
	3.7(1)	87(92)	219(164)	667(328)	1770(1489)	4064(3811)	4/15
f_9	35	127	214	300	335	369	15/15
	5.3(2)	69(48)	211(80)	1480(1872)	2065(1866)	5488(5520)	4/15
f_{10}	349	500	574	626	829	880	15/15
	3419(3283)	5375(5002)	∞	∞	∞	∞ *1.0e6*	0/15
f_{11}	143	202	763	1177	1467	1673	15/15
	3907(2964)	∞	∞	∞	∞	∞ *1.0e6*	0/15
f_{12}	108	268	371	461	1303	1494	15/15
	43(92)	220(274)	296(610)	5016(4650)	∞	∞ *1.0e6*	0/15
f_{13}	132	195	250	1310	1752	2255	15/15
	44(76)	157(175)	335(295)	860(642)	2575(2559)	6385(6733)	0/15
f_{14}	10	41	58	139	251	476	15/15
	2.6(3)	2.4(1)	3.3(1)	8.3(10)	12105(14446)	∞ *1.0e6*	0/15
f_{15}	511	9310	19369	20073	20769	21359	14/15
	5.0(10)	14(14)	23(20)	22(19)	22(18)	21(18)	14/15
f_{16}	120	612	2662	10449	11644	12095	15/15
	3.0(3)	7.8(16)	13(24)	11(10)	41(46)	61(61)	9/15
f_{17}	5.2	215	899	3669	6351	7934	15/15
	5.3(4)	4.2(0.9)	3.6(6)	2.7(3)	8.1(9)	19(20)	14/15
f_{18}	103	378	3968	9280	10905	12469	15/15
	1.1(0.9)	6.3(13)	1.9(3)	20(22)	647(648)	∞ *1.0e6*	0/15
f_{19}	1	1	242	1.2e5	1.2e5	1.2e5	15/15
	14(12)	3327(5308)	1668(2179)	36(45)	36(41)	36(37)	3/15
f_{20}	16	851	38111	54470	54861	55313	14/15
	2.3(2)	29(29)	23(20)	16(13)	16(12)	16(12)	11/15
f_{21}	41	1157	1674	1705	1729	1757	14/15
	51(123)	46(74)	53(65)	52(63)	52(62)	51(62)	15/15
f_{22}	71	386	938	1008	1040	1068	14/15
	91(152)	191(260)	134(155)	129(144)	149(188)	161(189)	15/15
f_{23}	3.0	518	14249	31654	33030	34256	15/15
	2.8(2)	35(38)	46(66)	416(468)	399(470)	384(412)	1/15
f_{24}	1622	2.2e5	6.4e6	9.6e6	1.3e7	1.3e7	3/15
	1.6(2)	3.2(4)	∞	∞	∞	∞ *9.8e5*	0/15

20-D

Δf	1e+1	1e+0	1e-1	1e-3	1e-5	1e-7	#succ
f_1	43	43	43	43	43	43	15/15
	5.4(0.8)	14(1)	25(3)	45(2)	66(2)	86(2)	15/15
f_2	385	386	387	390	391	393	15/15
	4.8(0.3)	5.9(0.2)	7.0(0.3)	9.2(0.3)	11(0.3)	14(0.4)	15/15
f_3	5066	7626	7635	7643	7646	7651	15/15
	550(526)	∞	∞	∞	∞	∞ *7.1e6*	0/15
f_4	4722	7628	7666	7700	7758	1.4e5	9/15
	4050(3891)	∞	∞	∞	∞	∞ *7.0e6*	0/15
f_5	41	41	41	41	41	41	15/15
	9.4(2)	12(3)	12(3)	12(3)	12(3)	12(3)	15/15
f_6	1296	2343	3413	5220	6728	8409	15/15
	1.3(0.2)	1.2(0.2)	1.2(0.2)	35(57)	137(396)	445(523)	11/15
f_7	1351	4274	9503	16524	16524	16969	15/15
	32(59)	2715(3070)	∞	∞	∞	∞ *7.0e6*	0/15
f_8	2039	3871	4040	4219	4371	4484	15/15
	37(10)	145(138)	299(251)	1107(1149)	12891(14612)	25981(28550)	0/15
f_9	1716	3102	3277	3455	3594	3727	15/15
	520(442)	17624(19345)	35043(37845)	∞	∞	∞ *8.0e6*	0/15
f_{10}	7413	8661	10735	14920	17073	17476	15/15
	∞	∞	∞	∞	∞	∞ *8.0e6*	0/15
f_{11}	1002	2228	6278	9762	12285	14831	15/15
	1.2e5(*1e5*)	∞	∞	∞	∞	∞ *8.0e6*	0/15
f_{12}	1042	1938	2740	4140	12407	13827	15/15
	23(38)	37(41)	122(57)	3134(3400)	∞	∞ *8.0e6*	0/15
f_{13}	652	2021	2751	18749	24455	30201	15/15
	11(0.5)	54(59)	153(139)	140(126)	796(800)	∞ *7.3e6*	0/15
f_{14}	75	239	304	932	1648	15661	15/15
	2.6(0.9)	2.4(0.4)	3.9(0.4)	8.5(4)	∞	∞ *8.0e6*	0/15
f_{15}	30378	1.5e5	3.1e5	3.2e5	4.5e5	4.6e5	15/15
	49(76)	∞	∞	∞	∞	∞ *7.8e6*	0/15
f_{16}	1384	27265	77015	1.9e5	2.0e5	2.2e5	15/15
	2.7(1)	11(15)	157(157)	∞	∞	∞ *8.0e6*	0/15
f_{17}	63	1030	4005	30677	56288	80472	15/15
	2.0(1)	1.0(0.3)	5.9(10)	3.9(5)	239(245)	∞ *8.0e6*	0/15
f_{18}	621	3972	19561	67569	1.3e5	1.5e5	15/15
	1.0(0.4)	2.0(0.4)	6.3(4)	547(593)	∞	∞ *8.0e6*	0/15
f_{19}	1	1	3.4e5	6.2e6	6.7e6	6.7e6	15/15
	118(40)	1.3e5 (*1e5*)	∞	∞	∞	∞ *8.0e6*	0/15
f_{20}	82	46150	3.1e6	5.5e6	5.6e6	5.6e6	14/15
	3.1(0.9)	18(19)	∞	∞	∞	∞ *7.0e6*	0/15
f_{21}	561	6541	14103	14643	15567	17589	15/15
	76(71)	93(116)	70(84)	67(81)	63(74)	56(67)	15/15
f_{22}	467	5580	23491	24948	26847	1.3e5	12/15
	205(255)	227(281)	312(318)	294(334)	317(325)	111(116)	4/15
f_{23}	3.2	1614	67457	4.9e5	8.1e5	8.4e5	15/15
	1.5(1)	102(102)	261(283)	∞	∞	∞ *7.5e6*	0/15
f_{24}	1.3e6	7.5e6	5.2e7	5.2e7	5.2e7	5.2e7	3/15
	12(14)	∞	∞	∞	∞	∞ *7.7e6*	0/15

Table 2: 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 $f_{\mathrm{opt}} + 10^{-8}$.

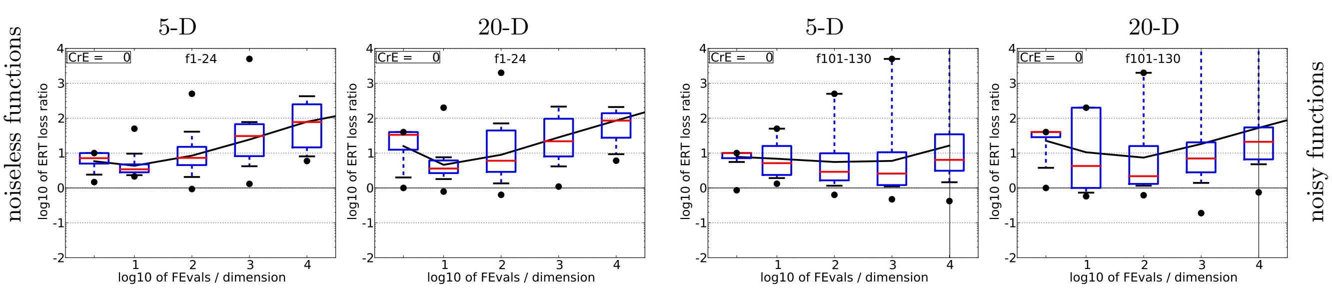

Figure 5: ERT loss ratio vs. a given budget FEvals. The target value f_{t} used for a given FEvals is the smallest (best) recorded function value such that $\mathrm{ERT}(f_{\mathrm{t}}) \leq \mathrm{FEvals}$ for the presented algorithm. Shown is FEvals divided by the respective best $\mathrm{ERT}(f_{\mathrm{t}})$ from BBOB-2009 for all functions (noiseless f_1–f_{24}, left columns, and noisy f_{101}–f_{130}, right columns) in 5-D and 20-D. Line: geometric mean. Box-Whisker error bar: 25-75%-ile with median (box), 10-90%-ile (caps), and minimum and maximum ERT loss ratio (points). The vertical line gives the maximal number of function evaluations in a single trial in this function subset.

5-D

Δf	1e+1	1e+0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathbf{f_{101}}$	11	37	44	62	69	75	15/15
	3.8(2)	2.5(0.9)	4.3(0.8)	6.1(0.7)	8.8(1.0)	11(1)	15/15
$\mathbf{f_{102}}$	11	35	50	72	86	99	15/15
	3.5(2)	2.3(1)	3.7(1)	5.4(0.7)	7.3(0.7)	8.6(0.4)	15/15
$\mathbf{f_{103}}$	11	28	30	31	35	115	15/15
	2.7(2)	2.7(1)	5.5(1)	13(2)	26(15)	21(11)	15/15
$\mathbf{f_{104}}$	173	773	1287	1768	2040	2284	15/15
	1.5(0.6)	21(26)	44(42)	∞	∞	∞ *5.0e4*	0/15
$\mathbf{f_{105}}$	167	1436	5174	10388	10824	11202	15/15
	1.7(0.4)	17(21)	12(12)	35(40)	∞	∞ *5.0e4*	0/15
$\mathbf{f_{106}}$	92	529	1050	2666	2887	3087	15/15
	3.9(1)	26(32)	209(228)	276(291)	∞	∞ *5.0e4*	0/15
$\mathbf{f_{107}}$	40	228	453	940	1376	1850	15/15
	4.2(6)	2.0(1)	1.4(0.8)	1.3(0.9)	1.4(0.7)	1.3(0.5)	15/15
$\mathbf{f_{108}}$	87	5144	14469	30935	58628	80667	15/15
	15(20)	1.5(3)	8.4(9)	∞	∞	∞ *5.0e4*	0/15
$\mathbf{f_{109}}$	11	57	216	572	873	946	15/15
	4.4(2)	1.9(0.7)	0.93(0.3)	1.8(1)	13(21)	371(411)	0/15
$\mathbf{f_{110}}$	949	33625	1.2e5	5.9e5	6.0e5	6.1e5	15/15
	0.76(1)	**0.33**(0.2)↓	0.65(0.7)	∞	∞	∞ *5.0e4*	0/15
$\mathbf{f_{111}}$	6856	6.1e5	8.8e6	2.3e7	3.1e7	3.1e7	3/15
	1.9(2)	∞	∞	∞	∞	∞ *5.0e4*	0/15
$\mathbf{f_{112}}$	107	1684	3421	4502	5132	5596	15/15
	1.9(0.4)	16(19)	19(20)	∞	∞	∞ *5.0e4*	0/15
$\mathbf{f_{113}}$	133	1883	8081	24128	24128	24402	15/15
	1.8(2)	0.78(0.9)	2.1(3)	3.1(3)	3.1(3)	4.1(5)	5/15
$\mathbf{f_{114}}$	767	14720	56311	83272	83272	84949	15/15
	2.6(3)	2.4(3)	∞	∞	∞	∞ *5.0e4*	0/15
$\mathbf{f_{115}}$	64	485	1829	2550	2550	2970	15/15
	1.6(0.8)	1.8(2)	4.4(4)	40(45)	40(40)	45(44)	4/15
$\mathbf{f_{116}}$	5730	14472	22311	26868	30329	31661	15/15
	1.6(1)	5.9(6)	∞	∞	∞	∞ *5.0e4*	0/15
$\mathbf{f_{117}}$	26686	76052	1.1e5	1.4e5	1.7e5	1.9e5	15/15
	6.3(7)	∞	∞	∞	∞	∞ *5.0e4*	0/15
$\mathbf{f_{118}}$	429	1217	1555	1998	2430	2913	15/15
	11(9)	33(36)	∞	∞	∞	∞ *5.0e4*	0/15
$\mathbf{f_{119}}$	12	657	1136	10372	35296	49747	15/15
	3.9(6)	0.64(0.9)	0.64(0.5)	0.84(0.9)	∞	∞ *5.0e4*	0/15
$\mathbf{f_{120}}$	16	2900	18698	72438	3.3e5	5.5e5	15/15
	12(32)	0.72(0.9)	38(44)	∞	∞	∞ *5.0e4*	0/15
$\mathbf{f_{121}}$	8.6	111	273	1583	3870	6195	15/15
	2.6(3)	1.1(0.8)	0.77(0.4)	11(14)	∞	∞ *5.0e4*	0/15
$\mathbf{f_{122}}$	10	1727	9190	30087	53743	1.1e5	15/15
	7.8(10)	0.65(0.6)	2.3(3)	∞	∞	∞ *5.0e4*	0/15
$\mathbf{f_{123}}$	11	16066	81505	3.4e5	6.7e5	2.2e6	15/15
	12(16)	3.1(3)	∞	∞	∞	∞ *5.0e4*	0/15
$\mathbf{f_{124}}$	10	202	1040	20478	45337	95200	15/15
	2.9(4)	1.2(1.0)	1.2(0.9)	∞	∞	∞ *5.0e4*	0/15
$\mathbf{f_{125}}$	1	1	1	2.4e5	2.4e5	2.5e5	15/15
	1.2(0.5)	33(34)	3958(3495)	∞	∞	∞ *5.0e4*	0/15
$\mathbf{f_{126}}$	1	1	1	∞	∞	∞	0
	1.4(1)	32(50)	51876(53162)	∞	∞	∞	0/15
$\mathbf{f_{127}}$	1	1	1	3.4e5	3.9e5	4.0e5	15/15
	1.1(0.5)	19(16)	3060(2752)	∞	∞	∞ *5.0e4*	0/15
$\mathbf{f_{128}}$	111	4248	7808	12447	17217	21162	15/15
	1.6(2)	1.1(1)	0.78(0.8)	0.51(0.5)	**0.40**(0.3)↓2	**0.45**(0.4)↓2	15/15
$\mathbf{f_{129}}$	64	10710	59443	2.8e5	5.1e5	5.8e5	15/15
	7.6(14)	1.2(1)	1.8(2)	∞	∞	∞ *5.0e4*	0/15
$\mathbf{f_{130}}$	55	812	3034	32823	33889	34528	10/15
	2.9(7)	4.8(5)	1.6(2)	0.19(0.2)	0.44(0.4)	2.7(2)	3/15

20-D

Δf	1e+1	1e+0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathbf{f_{101}}$	59	425	571	700	739	783	15/15
	4.1(0.7)	1.5(0.2)	1.9(0.1)	2.8(0.2)	3.9(0.1)	4.7(0.2)	15/15
$\mathbf{f_{102}}$	231	399	579	921	1157	1407	15/15
	1.1(0.2)	1.6(0.3)	1.9(0.1)	2.1(0.1)	2.5(0.1)	2.7(0.1)	15/15
$\mathbf{f_{103}}$	65	417	629	1313	1893	2464	14/15
	3.8(0.9)	1.5(0.2)	1.7(0.1)	1.6(0.1)	1.7(0.1)	3.4(2)	15/15
$\mathbf{f_{104}}$	23690	85656	1.7e5	1.8e5	1.9e5	2.0e5	15/15
	∞	∞	∞	∞	∞	∞ *2.0e5*	0/15
$\mathbf{f_{105}}$	1.9e5	6.1e5	6.3e5	6.5e5	6.6e5	6.7e5	15/15
	∞	∞	∞	∞	∞	∞ *2.0e5*	0/15
$\mathbf{f_{106}}$	11480	21668	23746	25470	26492	27360	15/15
	123(140)	∞	∞	∞	∞	∞ *2.0e5*	0/15
$\mathbf{f_{107}}$	8571	13582	16226	27357	52486	65052	15/15
	2.6(3)	14(15)	∞	∞	∞	∞ *2.0e5*	0/15
$\mathbf{f_{108}}$	58063	97228	2.0e5	4.5e5	6.3e5	9.0e5	15/15
	∞	∞	∞	∞	∞	∞ *2.0e5*	0/15
$\mathbf{f_{109}}$	333	632	1138	2287	3583	4952	15/15
	0.77(0.1)↓2	1.1(0.1)	1.3(0.2)	3.8(3)	15(13)	103(104)	3/15
$\mathbf{f_{110}}$	∞	∞	∞	∞	∞	∞	0
	∞	∞	∞	∞	∞	∞	0/15
$\mathbf{f_{111}}$	∞	∞	∞	∞	∞	∞	0
	∞	∞	∞	∞	∞	∞	0/15
$\mathbf{f_{112}}$	25552	64124	69621	73557	76137	78238	15/15
	113(120)	∞	∞	∞	∞	∞ *2.0e5*	0/15
$\mathbf{f_{113}}$	50123	3.6e5	5.6e5	5.9e5	5.9e5	5.9e5	15/15
	4.5(5)	∞	∞	∞	∞	∞ *2.0e5*	0/15
$\mathbf{f_{114}}$	2.1e5	1.1e6	1.4e6	1.6e6	1.6e6	1.6e6	15/15
	∞	∞	∞	∞	∞	∞ *2.0e5*	0/15
$\mathbf{f_{115}}$	2405	30268	91749	1.3e5	1.3e5	1.3e5	15/15
	1.2(1)	29(32)	∞	∞	∞	∞ *2.0e5*	0/15
$\mathbf{f_{116}}$	5.0e5	6.9e5	8.9e5	1.0e6	1.1e6	1.1e6	15/15
	∞	∞	∞	∞	∞	∞ *2.0e5*	0/15
$\mathbf{f_{117}}$	1.8e6	2.5e6	2.6e6	2.9e6	3.2e6	3.6e6	15/15
	∞	∞	∞	∞	∞	∞ *2.0e5*	0/15
$\mathbf{f_{118}}$	6908	11786	17514	26342	30062	32659	15/15
	∞	∞	∞	∞	∞	∞ *2.0e5*	0/15
$\mathbf{f_{119}}$	2771	29365	35930	4.1e5	1.4e6	1.9e6	15/15
	1.4(2)	22(22)	∞	∞	∞	∞ *2.0e5*	0/15
$\mathbf{f_{120}}$	36040	1.8e5	2.8e5	1.6e6	6.7e6	1.4e7	13/15
	10(11)	∞	∞	∞	∞	∞ *2.0e5*	0/15
$\mathbf{f_{121}}$	249	769	1426	9304	34434	57404	15/15
	0.83(0.2)	0.89(0.2)	1.3(0.2)	∞	∞	∞ *2.0e5*	0/15
$\mathbf{f_{122}}$	692	52008	1.4e5	7.9e5	2.0e6	5.8e6	15/15
	1.9(2)	∞	∞	∞	∞	∞ *2.0e5*	0/15
$\mathbf{f_{123}}$	1063	5.3e5	1.5e6	5.3e6	2.7e7	1.6e8	0
	7.7(9)	∞	∞	∞	∞	∞ *2.0e5*	0/15
$\mathbf{f_{124}}$	192	1959	40840	1.3e5	3.9e5	8.0e5	15/15
	0.58(0.4)↓	**0.69**(0.2)↓	0.66(0.5)	∞	∞	∞ *2.0e5*	0/15
$\mathbf{f_{125}}$	1	1	1	2.5e7	8.0e7	8.1e7	4/15
	1.3(0.5)	625(509)	∞	∞	∞	∞ *2.0e5*	0/15
$\mathbf{f_{126}}$	1	1	1	∞	∞	∞	0
	1.3(0.5)	22572(23656)	∞	∞	∞	∞	0/15
$\mathbf{f_{127}}$	1	1	1	4.4e6	7.3e6	7.4e6	15/15
	1.2(0.5)	167(80)	∞	∞	∞	∞ *2.0e5*	0/15
$\mathbf{f_{128}}$	1.4e5	1.3e7	1.7e7	1.7e7	1.7e7	1.7e7	9/15
	4.2(5)	∞	∞	∞	∞	∞ *2.0e5*	0/15
$\mathbf{f_{129}}$	7.8e6	4.1e7	4.2e7	4.2e7	4.2e7	4.2e7	5/15
	∞	∞	∞	∞	∞	∞ *2.0e5*	0/15
$\mathbf{f_{130}}$	4904	93149	2.5e5	2.5e5	2.6e5	2.6e5	7/15
	0.76(1)	0.19(0.3)	0.09(0.1)	0.11(0.1)	0.36(0.3)	2.1(2)	2/15

Table 3: ERT ratios, as in table 2, for functions f_{101}–f_{130}.

[8] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *IEEE Transactions on Evolutionary Computation*, 9:159–195, 2001.

[9] R. Ros and N. Hansen. A Simple Modification in CMA-ES Achieving Linear Time and Space Complexity. Technical Report April, 2008.

[10] T. Schaul. *Studies in Continuous Black-box Optimization*. Ph.D. thesis, Technische Universität München, 2011.

[11] T. Schaul. Natural Evolution Strategies Converge on Sphere Functions. In *Genetic and Evolutionary Computation Conference (GECCO)*, Philadelphia, PA, 2012.

[12] T. Schaul, J. Bayer, D. Wierstra, Y. Sun, M. Felder, F. Sehnke, T. Rückstieß, and J. Schmidhuber. PyBrain. *Journal of Machine Learning Research*, 11:743–746, 2010.

[13] T. Schaul, T. Glasmachers, and J. Schmidhuber. High Dimensions and Heavy Tails for Natural Evolution Strategies. In *Genetic and Evolutionary Computation Conference (GECCO)*, Dublin, Ireland, 2011.

[14] Y. Sun, D. Wierstra, T. Schaul, and J. Schmidhuber. Stochastic search using the natural gradient. In *International Conference on Machine Learning (ICML)*, 2009.

[15] D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, and J. Schmidhuber. Natural Evolution Strategies. Technical report, 2011.

[16] D. Wierstra, T. Schaul, J. Peters, and J. Schmidhuber. Natural Evolution Strategies. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, Hong Kong, China, 2008.