Benchmarking Natural Evolution Strategies with Adaptation Sampling 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. Exponential NES (xNES) are the most common instantiation of NES, and particularly appropriate for the BBOB 2012 benchmarks, given that many are non-separable, and their relatively small problem dimensions. The technique of *adaptation sampling*, which adapts learning rates online further improves the algorithm's performance. This report provides the the most extensive empirical results on that combination (xNES-as) 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 [10]) innovated the field by introducing a parameterization that includes the full covariance matrix, allowing them to solve highly non-separable problems.

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

A more recent variant, *natural evolution strategies* (NES [17, 6, 15, 16]) 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 [12, 11] 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.

Exponential NES (xNES), the most common instantiation of NES, used a search distribution parameterized by a mean vector and a full covariance matrix, and is thus most similar to CMA-ES (in fact, the precise relation is described in [4] and [5]). Given the relatively small problem dimensions of the BBOB benchmarks, and the fact that many are non-separable, it is also among the most appropriate NES variants for the task.

In this report, we retain the original formulation of xNES (including all parameter settings, except for an added stopping criterion), augmented with a technique for the online adaptation of its learning rate called *adaptation sampling*, which is designed to speed up convergence. We 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 [10], 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 [17]) 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 updated using natural gradient ascent

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

with learning rate parameter η.

2.1 Exponential NES

While the NES formulation is applicable to arbitrary parameterizable search distributions [17, 11], the most common variant employs multinormal search distributions. For that case, two helpful techniques were introduced in [6], 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*, and the pseudocode is given in Algorithm 1.

Algorithm 1: Exponential NES (xNES)

input: f, $\boldsymbol{\mu}_{\text{init}}$, η_σ, $\eta_{\mathbf{B}}$, u_k

initialize $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu}_{\text{init}}$, $\sigma \leftarrow 1$, $\mathbf{B} \leftarrow \mathbb{I}$

repeat

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

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

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

    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

  $\nabla_{\boldsymbol{\delta}} J \leftarrow \sum_{k=1}^n u_k \cdot \mathbf{s}_k$

  $\nabla_{\mathbf{M}} J \leftarrow \sum_{k=1}^n u_k \cdot (\mathbf{s}_k \mathbf{s}_k^\top - \mathbb{I})$

  $\nabla_\sigma J \leftarrow \operatorname{tr}(\nabla_{\mathbf{M}} J)/d$

  $\nabla_{\mathbf{B}} J \leftarrow \nabla_{\mathbf{M}} J - \nabla_\sigma J \cdot \mathbb{I}$

  update parameters

  $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} + \sigma \mathbf{B} \cdot \nabla_{\boldsymbol{\delta}} J$

  $\sigma \leftarrow \sigma \cdot \exp(\eta_\sigma/2 \cdot \nabla_\sigma J)$

  $\mathbf{B} \leftarrow \mathbf{B} \cdot \exp(\eta_{\mathbf{B}}/2 \cdot \nabla_{\mathbf{B}} J)$

until *stopping criterion is met*

2.2 Adaptation Sampling

First introduced in [11] (chapter 2, section 4.4), *adaptation sampling* is a new meta-learning technique [14] that can adapt hyper-parameters online, in an economical way that is grounded on a measure statistical improvement.

Here, we apply it to the learning rate of the global step-size η_σ. The idea is to consider whether a larger learning-rate $\eta'_\sigma = \frac{3}{2}\eta_\sigma$ would have been more likely to generate the good samples in the current batch. For this we determine the (hypothetical) search distribution that would have resulted from such a larger update $\pi(\cdot|\theta')$. Then we compute importance weights

$$w'_k = \frac{\pi(\mathbf{z}_k|\theta')}{\pi(\mathbf{z}_k|\theta)}$$

for each of the n samples $\mathbf{z}_k$ in our current population, generated from the actual search distribution $\pi(\cdot|\theta)$. We then conduct a *weighted* Mann-Whitney test [11] (appendix A) to determine if the set $\{\text{rank}(\mathbf{z}_k)\}$ is inferior to its reweighted counterpart $\{w'_k \cdot \text{rank}(\mathbf{z}_k)\}$ (corresponding to the larger learning rate), with statistical significance ρ. If so, we increase the learning rate by a factor of $1 + c'$, up to at most $\eta_\sigma = 1$ (where $c' = 0.1$). Otherwise it decays to its initial value:

$$\eta_\sigma \leftarrow (1 - c') \cdot \eta_\sigma + c' \cdot \eta_{\sigma,\text{init}}$$

The procedure is summarized in algorithm 2 (for details and derivations, see [11]). The combination of xNES with adaptation sampling is dubbed *xNES-as*.

One interpretation of why adaptation sampling is helpful is that half-way into the search, (after a local attractor has been found, e.g., towards the end of the valley on the Rosenbrock benchmarks f_8 or f_9), the convergence speed can be boosted by an increased learning rate. For such situations, an online adaptation of hyper-parameters is inherently well-suited.

Algorithm 2: Adaptation sampling

input : $\eta_{\sigma,t}$,$\eta_{\sigma,\text{init}}$, θ_t, θ_{t-1}, $\{(\mathbf{z}_k, f(\mathbf{z}_k))\}$, c', ρ

output: $\eta_{\sigma,t+1}$

compute hypothetical θ', given θ_{t-1} and using $3/2\eta_{\sigma,t}$

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

  $w'_k = \frac{\pi(\mathbf{z}_k|\theta')}{\pi(\mathbf{z}_k|\theta)}$

end

$S \leftarrow \{\text{rank}(\mathbf{z}_k)\}$

$S' \leftarrow \{w'_k \cdot \text{rank}(\mathbf{z}_k)\}$

if *weighted-Mann-Whitney*$(S, S') < \rho$ **then**

  return $(1 - c') \cdot \eta_\sigma + c' \cdot \eta_{\sigma,\text{init}}$

else

  return $\min((1 + c') \cdot \eta_\sigma, 1)$

end

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	$4 + \lfloor 3\log(d) \rfloor$
$\eta_\sigma = \eta_{\mathbf{B}}$	$\frac{3(3 + \log(d))}{5d\sqrt{d}}$
u_k	$\frac{\max\left(0, \log(\frac{n}{2} + 1) - \log(k)\right)}{\sum_{j=1}^n \max\left(0, \log(\frac{n}{2} + 1) - \log(j)\right)} - \frac{1}{n}$
ρ	$\frac{1}{2} - \frac{1}{3(d+1)}$
c'	$\frac{1}{10}$

3. EXPERIMENTAL SETTINGS

We use identical default hyper-parameter values for all benchmarks (both noisy and noise-free functions), which are taken from [6, 11]. 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 [13], 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 xNES-as implementation (in Python, based on the PyBrain [13] library), running on an Intel Xeon with 2.67GHz, required an average time of 1.1, 0.9, 0.7, 0.7, 0.9, 2.7 milliseconds per function evaluation for dimensions 2, 5, 10, 20, 40, 80 respectively (the function evaluations themselves take about 0.1ms).

5. RESULTS

Results of xNES-as on the noiseless testbed (from experiments according to [7] on the benchmark functions given in [2, 8]) are presented in Figures 1, 3 and 5, and in Table 2.

Similarly, results of xNES-as on the testbed of noisy functions (from experiments according to [7] on the benchmark functions given in [3, 9]) are presented in Figures 2, 4 and 5, and in Table 3.

6. DISCUSSION

The top rows in Figures 3 and 4 give a good overview picture, showing that across all benchmarks taken together, xNES-as performs almost as well as the best and better than most of the BBOB 2009 contestants. Beyond this high-level perspective, the results speak for themselves, of course, we will just highlight a few observations.

According to Tables 2 and 3, the only conditions where xNES-as significantly outperforms *all* algorithms from the BBOB2009 competition are on function f_{115} and in the early phase on f_{18}, f_{118} and f_{119}. We observe the worst performance on multimodal functions like f_3, f_4 and f_{15} that other algorithms tackle very easily. Comparing different types of noise, xNES-as appears to be least sensitive to Cauchy noise and most sensitive to uniform noise (see Figure 2).

From Figure 5 we observe a good loss ratio across the board on all benchmarks, with the best ones on moderate functions, ill-conditioned functions, and for all levels of noise. These results hold equally in dimensions 5 and 20. On the other hand, the algorithm is less competitive on (noisy or noise-free) multimodal benchmarks, which we expect to be directly related to its small default population size.

[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 [11].

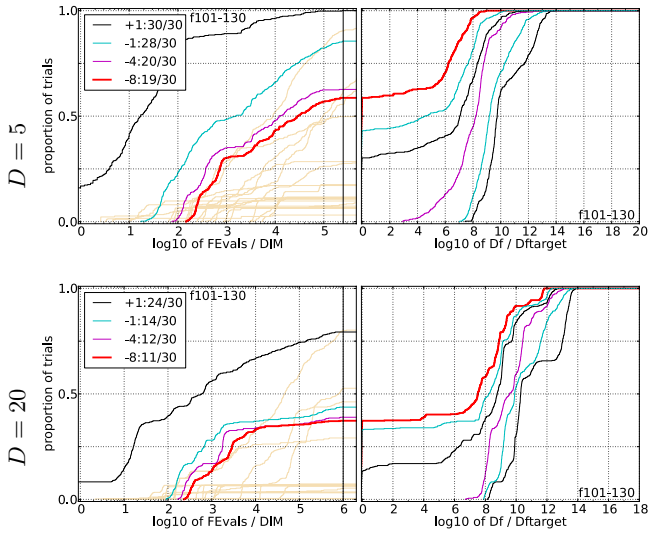

Figure 4: Empirical cumulative distribution functions (ECDFs) of the 30 noisy benchmark functions. Plotted is the fraction of trials versus running time (left subplots) or versus Δf (right subplots) (see Figure 3 for details).

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] N. Fukushima, Y. Nagata, S. Kobayashi, and I. Ono. Proposal of distance-weighted exponential natural evolution strategies. In *2011 IEEE Congress of Evolutionary Computation*, pages 164–171. IEEE, 2011.

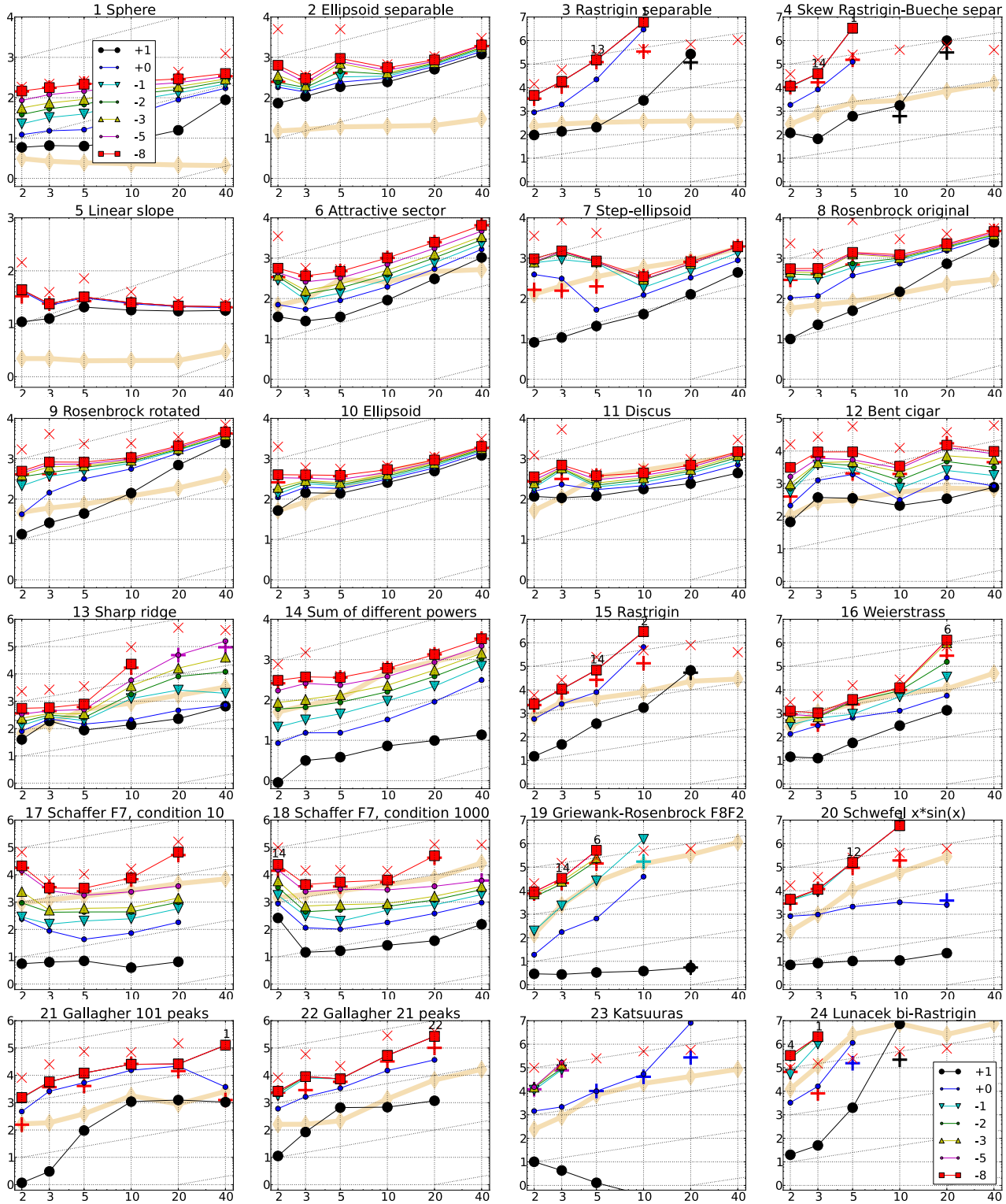

Figure 1: Expected number of f-evaluations (ERT, with lines, see legend) to reach $f_{\mathrm{opt}} + \Delta f$, median number of f-evaluations to reach the most difficult target that was reached at least once ($+$) and maximum number of f-evaluations in any trial ($\times$), all divided by dimension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

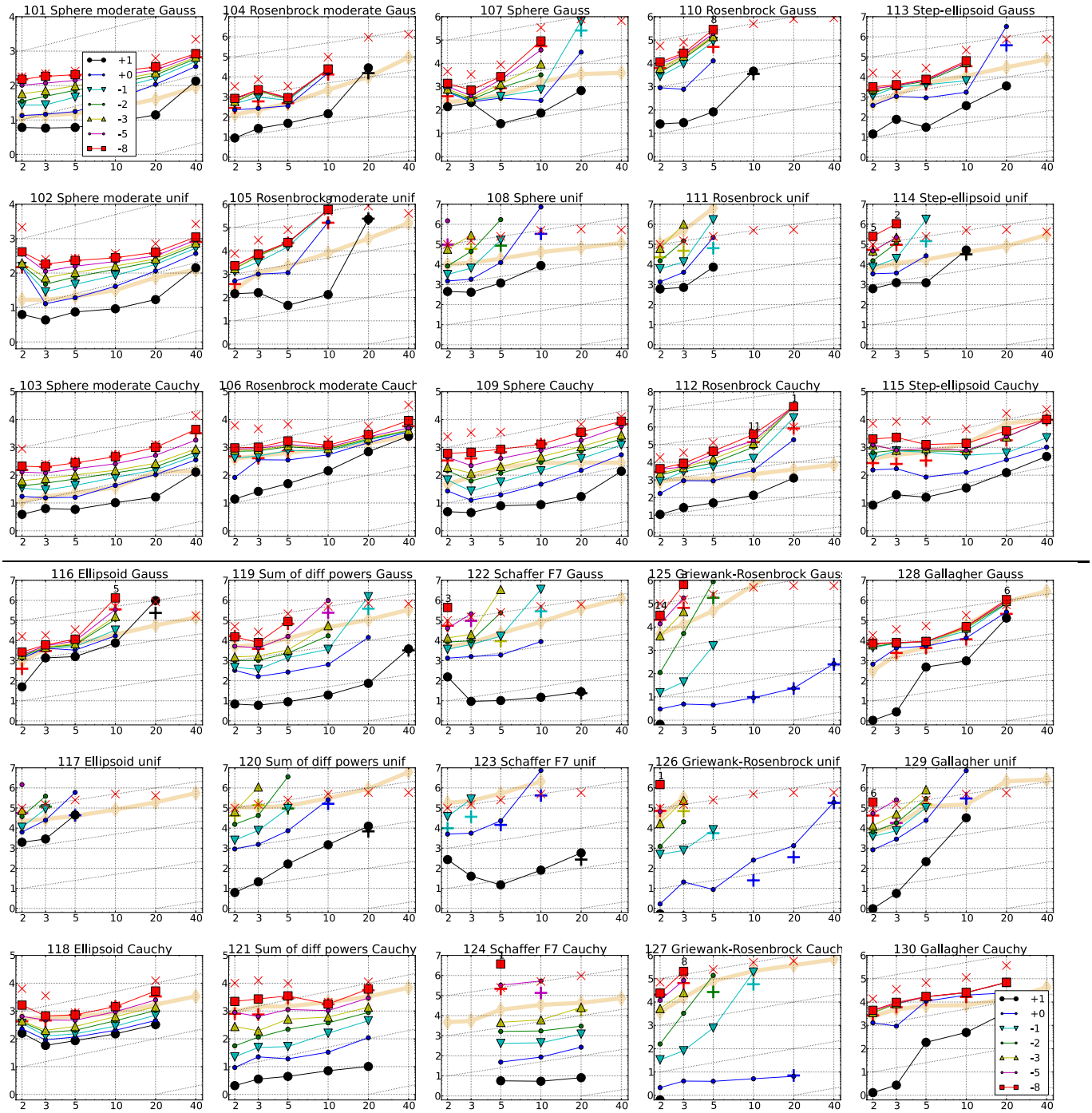

Figure 2: Expected number of f-evaluations (ERT, with lines, see legend) to reach $f_{\mathrm{opt}} + \Delta f$, median number of f-evaluations to reach the most difficult target that was reached at least once (+) and maximum number of f-evaluations in any trial (×), all divided by dimension and plotted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

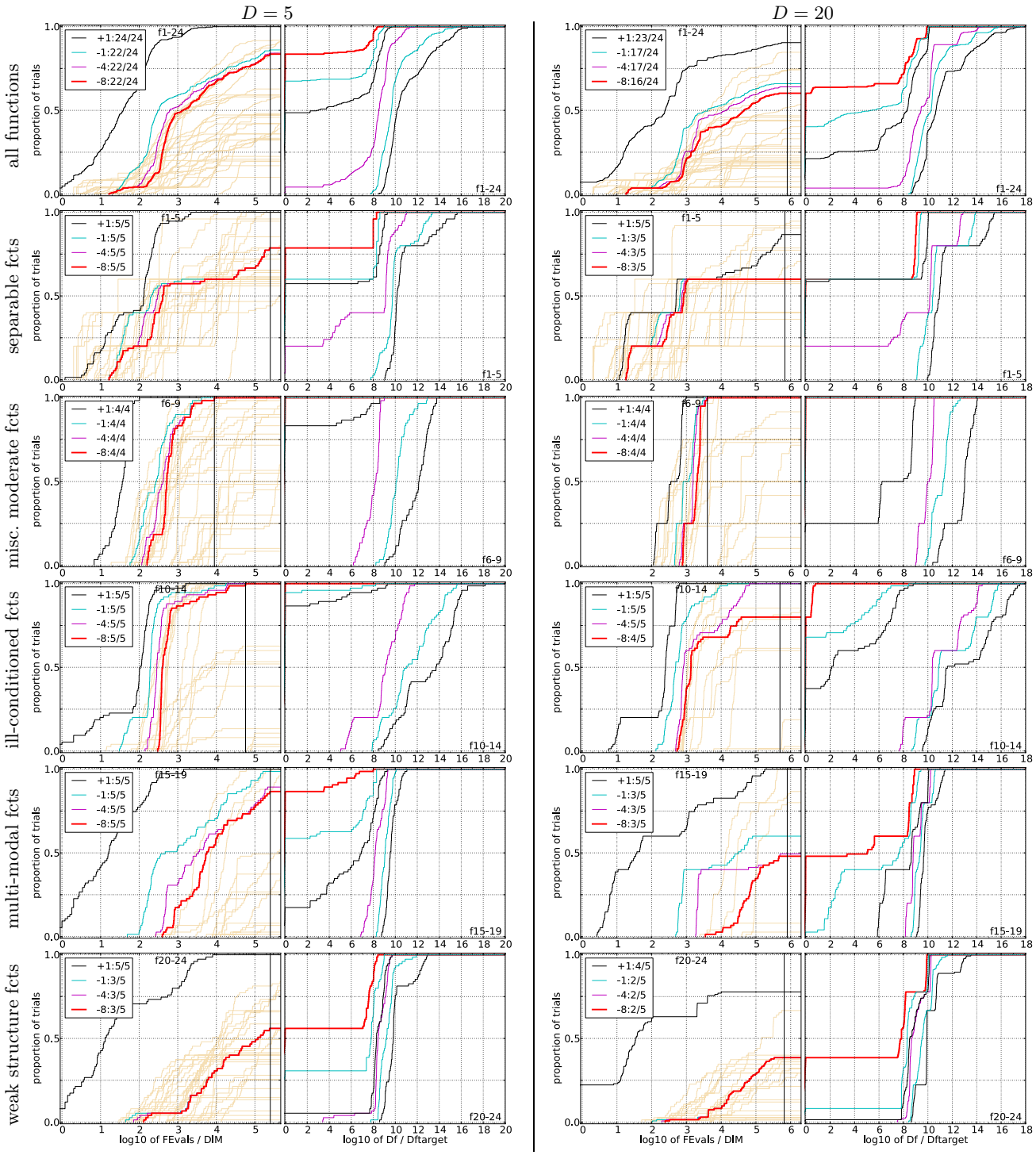

Figure 3: Empirical cumulative distribution functions (ECDFs), plotting the fraction of trials with an outcome not larger than the respective value on the x-axis. Left subplots: ECDF of number of function evaluations (FEvals) divided by search space dimension D, to fall below $f_{\mathrm{opt}} + \Delta f$ 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 $f_{\mathrm{opt}} + 10^{-8}$. 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
	2.9(2)	6.6(3)	16(5)	37(8)	60(12)	78(17)	15/15
f_2	83	87	88	90	92	94	15/15
	11(5)	14(9)	19(18)	39(62)	43(63)	49(92)	15/15
f_3	716	1622	1637	1646	1650	1654	15/15
	1.5(0.7)	69(80)	454(375)	452(384)	451(377)	450(382)	13/15
f_4	809	1633	1688	1817	1886	1903	15/15
	3.8(5)	381(434)	9998(10786)	9287(9745)	8949(9769)	8868(8768)	1/15
f_5	10	10	10	10	10	10	15/15
	10(4)	15(9)	16(9)	16(8)	16(8)	16(8)	15/15
f_6	114	214	281	580	1038	1332	15/15
	1.5(1)	2.1(0.6)	2.4(0.6)	2.0(0.2)	1.5(0.2)	1.6(0.2)	15/15
f_7	24	324	1171	1572	1572	1597	15/15
	4.4(3)	0.81(0.3)	3.2(4)	2.6(3)	2.6(3)	2.6(3)	15/15
f_8	73	273	336	391	410	422	15/15
	3.4(2)	6.8(3)	8.7(4)	16(13)	16(13)	16(12)	15/15
f_9	35	127	214	300	335	369	15/15
	6.4(2)	13(4)	12(3)	11(6)	11(6)	11(6)	15/15
f_{10}	349	500	574	626	829	880	15/15
	2.0(0.8)	1.8(0.7)	1.8(0.6)	2.0(0.5)	1.9(0.4)	2.0(0.3)	15/15
f_{11}	143	202	763	1177	1467	1673	15/15
	4.2(3)	4.3(1)	1.3(0.3)	1.1(0.2)	1.0(0.1)	1.1(0.1)	15/15
f_{12}	108	268	371	461	1303	1494	15/15
	16(28)	36(66)	36(58)	51(97)	21(35)	31(34)	15/15
f_{13}	132	195	250	1310	1752	2255	15/15
	3.3(0.6)	3.7(0.5)	4.0(0.5)	1.3(0.2)	1.4(0.2)	1.5(0.2)	15/15
f_{14}	10	41	58	139	251	476	15/15
	2.0(2)	1.9(0.9)	3.9(1)	4.9(1)	4.6(0.5)	3.3(0.3)	15/15
f_{15}	511	9310	19369	20073	20769	21359	14/15
	3.6(6)	4.3(4)	18(20)	18(19)	17(19)	16(18)	14/15
f_{16}	120	612	2662	10449	11644	12095	15/15
	2.3(2)	5.3(8)	1.7(3)	1.8(2)	1.6(2)	1.6(2)	15/15
f_{17}	5.2	215	899	3669	6351	7934	15/15
	6.8(7)	1.0(0.7)	1.1(0.7)	0.81(0.7)	1.4(1)	2.0(3)	15/15
f_{18}	103	378	3968	9280	10905	12469	15/15
	0.80(0.5)	1.4(0.3)	0.25(0.1)	0.43(0.5)	1.4(0.9)	2.0(3)	15/15
f_{19}	1	1	242	1.2e5	1.2e5	1.2e5	15/15
	17(18)	3280(5087)	542(792)	11(11)	20(21)	21(23)	6/15
f_{20}	16	851	38111	54470	54861	55313	14/15
	3.2(3)	12(22)	21(24)	15(16)	15(17)	14(16)	12/15
f_{21}	41	1157	1674	1705	1729	1757	14/15
	12(1)	23(25)	36(57)	35(56)	35(55)	35(54)	15/15
f_{22}	71	386	938	1008	1040	1068	14/15
	46(61)	44(57)	39(42)	37(39)	36(37)	35(37)	15/15
f_{23}	3.0	518	14249	31654	33030	34256	15/15
	2.2(2)	97(97)	∞	∞	∞	∞ *1.3e6*	0/15
f_{24}	1622	2.2e5	6.4e6	9.6e6	1.3e7	1.3e7	3/15
	6.2(8)	26(27)	∞	∞	∞	∞ *1.3e6*	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
	7.3(2)	41(9)	61(16)	88(23)	110(25)	128(32)	15/15
f_2	385	386	387	390	391	393	15/15
	26(1)	31(2)	34(3)	38(4)	41(6)	43(6)	15/15
f_3	5066	7626	7635	7643	7646	7651	15/15
	1055(1500)	∞	∞	∞	∞	∞ *1.3e7*	0/15
f_4	4722	7628	7666	7700	7758	1.4e5	9/15
	4193(4430)	∞	∞	∞	∞	∞ *1.2e7*	0/15
f_5	41	41	41	41	41	41	15/15
	8.6(1)	10(1)	11(2)	11(2)	11(2)	11(2)	15/15
f_6	1296	2343	3413	5220	6728	8409	15/15
	4.8(0.2)	4.6(0.2)	4.5(0.1)	4.8(0.1)	5.2(0.1)	5.3(0.1)	15/15
f_7	1351	4274	9503	16524	16524	16969	15/15
	1.9(0.2)	1.5(0.1)	1.0(0.1)	0.89(0.1)	0.89(0.1)	0.91(0.1)	15/15
f_8	2039	3871	4040	4219	4371	4484	15/15
	7.2(0.6)	7.9(2)	9.1(3)	9.4(4)	10(4)	10(4)	15/15
f_9	1716	3102	3277	3455	3594	3727	15/15
	8.1(1)	8.9(2)	10(2)	10(2)	11(2)	11(2)	15/15
f_{10}	7413	8661	10735	14920	17073	17476	15/15
	1.3(0.1)	1.4(0.1)	1.3(0.1)	1.0(0.1)	0.99(0.1)	1.0(0.1)	15/15
f_{11}	1002	2228	6278	9762	12285	14831	15/15
	4.8(0.3)	3.1(0.2)	1.4(0.1)	1.1(0.2)	1.00(0.2)	0.91(0.2)	15/15
f_{12}	1042	1938	2740	4140	12407	13827	15/15
	6.6(4)	16(8)	18(18)	35(38)	21(21)	22(21)	15/15
f_{13}	652	2021	2751	18749	24455	30201	15/15
	7.0(3)	4.6(3)	19(28)	17(21)	40(33)	81(83)	0/15
f_{14}	75	239	304	932	1648	15661	15/15
	2.6(0.8)	7.6(1)	14(3)	12(1)	10(1)	1.5(0.1)	15/15
f_{15}	30378	1.5e5	3.1e5	3.2e5	4.5e5	4.6e5	15/15
	44(52)	∞	∞	∞	∞	∞ *1.6e7*	0/15
f_{16}	1384	27265	77015	1.9e5	2.0e5	2.2e5	15/15
	20(10)	4.2(5)	9.2(7)	108(131)	133(153)	119(128)	6/15
f_{17}	63	1030	4005	30677	56288	80472	15/15
	2.1(1.0)	3.5(0.4)	2.9(0.2)	0.92(0.0)	1.4(0.7)	12(9)	15/15
f_{18}	621	3972	19561	67569	1.3e5	1.5e5	15/15
	1.2(0.5)	1.9(0.1)	**0.81**(0.0)$^{\downarrow 3}$	**0.48**(0.0)$^{\downarrow 4}$	0.58(0.3)	5.6(6)	15/15
f_{19}	1	1	3.4e5	6.2e6	6.7e6	6.7e6	15/15
	108(35)	∞	∞	∞	∞	∞ *1.2e7*	0/15
f_{20}	82	46150	3.1e6	5.5e6	5.6e6	5.6e6	14/15
	5.4(3)	1.1(0.7)	∞	∞	∞	∞ *1.2e7*	0/15
f_{21}	561	6541	14103	14643	15567	17589	15/15
	45(71)	66(98)	37(44)	35(55)	33(51)	30(35)	30/30
f_{22}	467	5580	23491	24948	26847	1.3e5	12/15
	50(86)	132(251)	226(268)	213(241)	198(206)	39(41)	22/30
f_{23}	3.2	1614	67457	4.9e5	8.1e5	8.4e5	15/15
	1.5(2)	99427(1e5)	∞	∞	∞	∞ *1.1e7*	0/30
f_{24}	1.3e6	7.5e6	5.2e7	5.2e7	5.2e7	5.2e7	3/15
	∞	∞	∞	∞	∞	∞ *1.2e7*	0/30

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

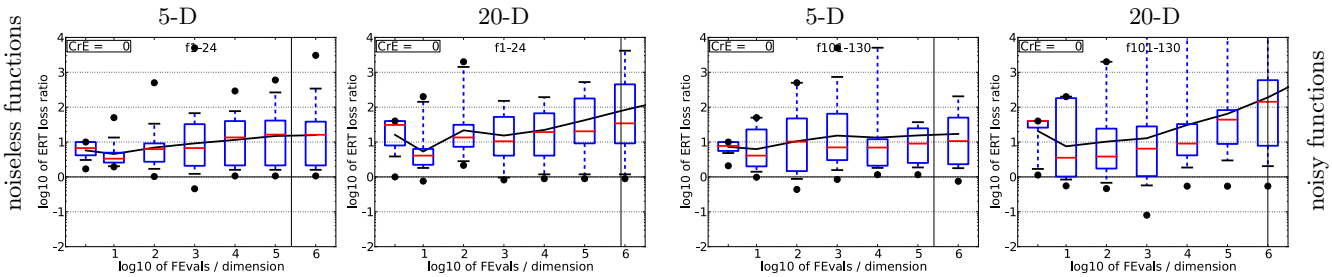

Figure 5: ERT **loss ratio vs. a given budget** FEvals**. The target value f_t used for a given** FEvals **is the smallest (best) recorded function value such that** $\mathrm{ERT}(f_t) \leq$ FEvals **for the presented algorithm. Shown is** FEvals **divided by the respective best** $\mathrm{ERT}(f_t)$ **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] T. Glasmachers, T. Schaul, and J. Schmidhuber. A Natural Evolution Strategy for Multi-Objective Optimization. In *Parallel Problem Solving from Nature (PPSN)*, 2010.

[6] 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.

[7] N. Hansen, A. Auger, S. Finck, and R. Ros. Real-parameter black-box optimization benchmarking 2012: Experimental setup. Technical report, INRIA, 2012.

[8] 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.

[9] N. Hansen, S. Finck, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking

5-D

Δf	1e+1	1e+0	1e-1	1e-3	1e-5	1e-7	#succ
f_{101}	11	37	44	62	69	75	15/15
	2.8(2)	2.4(1)	5.2(2)	7.9(2)	10(2)	13(3)	15/15
f_{102}	11	35	50	72	86	99	15/15
	3.3(3)	2.7(2)	4.7(1)	7.3(1)	9.2(1)	10(2)	15/15
f_{103}	11	28	30	31	35	115	15/15
	2.7(2)	2.9(2)	7.2(2)	16(3)	24(5)	11(2)	15/15
f_{104}	173	773	1287	1768	2040	2284	15/15
	1.4(0.4)	2.4(0.6)	2.7(6)	2.3(4)	2.1(4)	2.0(3)	15/15
f_{105}	167	1436	5174	10388	10824	11202	15/15
	1.4(0.5)	4.0(4)	14(13)	11(7)	11(7)	10(7)	15/15
f_{106}	92	529	1050	2666	2887	3087	15/15
	2.7(0.8)	3.3(0.9)	3.7(5)	2.2(3)	2.2(3)	2.7(3)	15/15
f_{107}	40	228	453	940	1376	1850	15/15
	3.2(5)	7.1(1)	4.2(0.5)	6.9(11)	7.2(8)	7.0(10)	15/15
f_{108}	87	5144	14469	30935	58628	80667	15/15
	69(120)	12(17)	56(60)	∞	∞	∞*1.3e6*	0/15
f_{109}	11	57	216	572	873	946	15/15
	3.6(2)	1.7(0.7)	1.3(0.5)	1.8(0.5)	2.3(0.6)	4.1(0.8)	15/15
f_{110}	949	33625	1.2e5	5.9e5	6.0e5	6.1e5	15/15
	0.44(0.2)	1.9(2)	3.6(5)	1.2(1)	1.6(2)	2.3(3)	8/15
f_{111}	6856	6.1e5	8.8e6	2.3e7	3.1e7	3.1e7	3/15
	5.4(8)	1.8(2)	0.96(1)	∞	∞	∞*1.3e6*	0/15
f_{112}	107	1684	3421	4502	5132	5596	15/15
	2.3(1)	2.6(5)	7.9(10)	17(18)	28(23)	37(38)	15/15
f_{113}	133	1883	8081	24128	24128	24402	15/15
	1.2(1)	2.4(4)	2.5(4)	1.4(2)	1.4(2)	1.5(2)	15/15
f_{114}	767	14720	56311	83272	83272	84949	15/15
	8.0(13)	9.1(13)	157(169)	∞	∞	∞*1.3e6*	0/15
f_{115}	64	485	1829	2550	2550	2970	15/15
	1.3(0.9)	0.88(0.4)	1.8(4)	1.8(4)	1.8(4)	1.6(3)	15/15
f_{116}	5730	14472	22311	26868	30329	31661	15/15
	1.4(2)	1.2(2)	1.1(2)	1.3(1)	1.6(2)	1.8(2)	15/15
f_{117}	26686	76052	1.1e5	1.4e5	1.7e5	1.9e5	15/15
	8.4(6)	39(49)	∞	∞	∞	∞*1.3e6*	0/15
f_{118}	429	1217	1555	1998	2430	2913	15/15
	1.00(0.4)	**0.48**(0.1)$^{\downarrow 4}$	**0.48**(0.1)$^{\downarrow 4}$	**0.69**(0.2)$^{\downarrow 2}$	0.96(0.2)	1.1(0.2)	15/15
f_{119}	12	657	1136	10372	35296	49747	15/15
	3.8(4)	2.1(1)	6.3(13)	1.6(2)	2.3(3)	5.9(6)	15/15
f_{120}	16	2900	18698	72438	3.3e5	5.5e5	15/15
	51(18)	13(21)	25(38)	∞	∞	∞*1.3e6*	0/15
f_{121}	8.6	111	273	1583	3870	6195	15/15
	2.6(2)	0.85(0.6)	0.96(0.5)	1.6(0.8)	1.5(2)	2.3(3)	15/15
f_{122}	10	1727	9190	30087	53743	1.1e5	15/15
	5.0(4)	5.5(11)	8.9(8)	583(727)	∞	∞*1.3e6*	0/15
f_{123}	11	16066	81505	3.4e5	6.7e5	2.2e6	15/15
	6.6(9)	7.2(7)	∞	∞	∞	∞*1.3e6*	0/15
f_{124}	10	202	1040	20478	45337	95200	15/15
	2.9(3)	1.2(0.5)	2.0(0.7)	1.1(1)	36(41)	60(66)	1/15
f_{125}	1	1	1	2.4e5	2.4e5	2.5e5	15/15
	1.3(0.5)	23(22)	7786(7916)	∞	∞	∞*1.3e6*	0/15
f_{126}	1	1	1	∞	∞	∞	0
	1.1(0.5)	44(88)	41598(46392)	∞	∞	∞	0/15
f_{127}	1	1	1	3.4e5	3.9e5	4.0e5	15/15
	1.1(0.5)	20(16)	3858(5345)	∞	∞	∞*1.3e6*	0/15
f_{128}	111	4248	7808	12447	17217	21162	15/15
	22(67)	6.2(9)	5.7(7)	3.6(4)	2.6(3)	2.1(2)	15/15
f_{129}	64	10710	59443	2.8e5	5.1e5	5.8e5	15/15
	17(16)	11(11)	8.6(11)	14(16)	∞	∞*1.3e6*	0/15
f_{130}	55	812	3034	32823	33889	34528	10/15
	17(0.9)	67(79)	28(48)	2.6(4)	2.5(4)	2.5(4)	15/15

20-D

Δf	1e+1	1e+0	1e-1	1e-3	1e-5	1e-7	#succ
f_{101}	59	425	571	700	739	783	15/15
	4.8(2)	5.1(0.8)	5.7(2)	6.7(3)	7.7(2)	8.4(2)	30/30
f_{102}	231	399	579	921	1157	1407	15/15
	1.5(0.4)	5.8(2)	6.2(2)	5.6(2)	5.4(2)	5.4(2)	30/30
f_{103}	65	417	629	1313	1893	2464	14/15
	5.0(2)	5.0(1)	4.9(2)	4.1(2)	5.5(3)	6.8(2)	30/30
f_{104}	23690	85656	1.7e5	1.8e5	1.9e5	2.0e5	15/15
	24(29)	∞	∞	∞	∞	∞*1.8e7*	0/30
f_{105}	1.9e5	6.1e5	6.3e5	6.5e5	6.6e5	6.7e5	15/15
	24(23)	∞	∞	∞	∞	∞*1.6e7*	0/21
f_{106}	11480	21668	23746	25470	26492	27360	15/15
	1.2(0.2)	1.4(0.3)	1.5(0.3)	1.7(0.3)	1.8(0.8)	2.0(0.7)	30/30
f_{107}	8571	13582	16226	27357	52486	65052	15/15
	1.6(0.9)	44(41)	786(980)	∞	∞	∞*1.5e7*	0/15
f_{108}	58063	97228	2.0e5	4.5e5	6.3e5	9.0e5	15/15
	∞	∞	∞	∞	∞	∞*1.1e7*	0/15
f_{109}	333	632	1138	2287	3583	4952	15/15
	1.0(0.2)	4.7(0.6)	6.9(0.8)	9.4(0.7)	10(0.5)	13(8)	15/15
f_{110}	∞	∞	∞	∞	∞	∞	0
	∞	∞	∞	∞	∞	∞	0/15
f_{111}	∞	∞	∞	∞	∞	∞	0
	∞	∞	∞	∞	∞	∞	0/15
f_{112}	25552	64124	69621	73557	76137	78238	15/15
	0.99(0.2)	59(67)	952(972)	3878(3942)	3747(3913)	3646(3966)	1/15
f_{113}	50123	3.6e5	5.6e5	5.9e5	5.9e5	5.9e5	15/15
	1.4(1)	179(181)	∞	∞	∞	∞*1.5e7*	0/15
f_{114}	2.1e5	1.1e6	1.4e6	1.6e6	1.6e6	1.6e6	15/15
	∞	∞	∞	∞	∞	∞*1.0e7*	0/15
f_{115}	2405	30268	91749	1.3e5	1.3e5	1.3e5	15/15
	1.0(0.1)	**0.24**(0.0)$^{\downarrow 4}$	**0.14**(0.0)$^{\downarrow}$	0.37(0.3)	0.37(0.3)	0.46(0.5)	15/15
f_{116}	5.0e5	6.9e5	8.9e5	1.0e6	1.1e6	1.1e6	15/15
	39(48)	∞	∞	∞	∞	∞*1.6e7*	0/15
f_{117}	1.8e6	2.5e6	2.6e6	2.9e6	3.2e6	3.6e6	15/15
	∞	∞	∞	∞	∞	∞*8.1e6*	0/30
f_{118}	6908	11786	17514	26342	30062	32659	15/15
	0.94(0.1)	0.77(0.1)	0.80(0.1)	1.1(0.1)	1.7(0.7)	2.1(0.7)	15/15
f_{119}	2771	29365	35930	4.1e5	1.4e6	1.9e6	15/15
	0.53(0.4)$^{\downarrow}$	10(10)	851(775)	∞	∞	∞*1.4e7*	0/13
f_{120}	36040	1.8e5	2.8e5	1.6e6	6.7e6	1.4e7	13/15
	6.9(7)	∞	∞	∞	∞	∞*1.1e7*	0/15
f_{121}	249	769	1426	9304	34434	57404	15/15
	0.81(0.3)	2.8(0.6)	6.3(0.5)	2.9(0.2)	1.7(0.6)	1.7(1)	15/15
f_{122}	692	52008	1.4e5	7.9e5	2.0e6	5.8e6	15/15
	0.81(0.9)	∞	∞	∞	∞	∞*1.2e7*	0/13
f_{123}	1063	5.3e5	1.5e6	5.3e6	2.7e7	1.6e8	0
	11(14)	∞	∞	∞	∞	∞*1.1e7*	0/15
f_{124}	192	1959	40840	1.3e5	3.9e5	8.0e5	15/15
	0.84(0.3)	2.7(0.5)	0.59(0.1)	4.0(3)	∞	∞*1.8e7*	0/15
f_{125}	1	1	1	2.5e7	8.0e7	8.1e7	4/15
	1.7(1)	470(238)	∞	∞	∞	∞*1.1e7*	0/15
f_{126}	1	1	1	∞	∞	∞	0
	1.4(1)	26641(51744)	∞	∞	∞	∞	0/15
f_{127}	1	1	1	4.4e6	7.3e6	7.4e6	15/15
	1.5(0.5)	131(87)	∞	∞	∞	∞*1.1e7*	0/15
f_{128}	1.4e5	1.3e7	1.7e7	1.7e7	1.7e7	1.7e7	9/15
	19(23)	0.40(0.4)	0.78(0.7)	1.0(1)	1.0(1)	1.3(1)	6/15
f_{129}	7.8e6	4.1e7	4.2e7	4.2e7	4.2e7	4.2e7	5/15
	∞	∞	∞	∞	∞	∞*1.1e7*	0/15
f_{130}	4904	93149	2.5e5	2.5e5	2.6e5	2.6e5	7/15
	15(23)	5.3(4)	5.4(9)	5.4(9)	5.4(9)	5.4(9)	15/15

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

2009: Noisy functions definitions. Technical Report RR-6869, INRIA, 2009. Updated February 2010.

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

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

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

[13] 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.

[14] T. Schaul and J. Schmidhuber. Metalearning. *Scholarpedia*, 5(6):4650, 2010.

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

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

[17] 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.