BBO-Benchmarking of Pure Random Search for Noiseless Function Testbed

An example BBOB 2009 Workshop Paper [*]

Anne Auger[†]
TAO Team - INRIA Saclay
LRI, Bat 490, Univ. Paris-Sud
91405 Orsay Cedex, France
anne.auger@inria.fr

Raymond Ros
TAO Team - INRIA Saclay
LRI, Bat 490, Univ. Paris-Sud
91405 Orsay Cedex, France
raymond.ros@lri.fr

The BBOBies

ABSTRACT

As an example, we benchmark the Pure-Random-Search algorithm on the noisefree BBOB 2009 testbed. Each candidate solution is sampled uniformly in $[-5,5]^D$, where D denotes the search space dimension. The maximum number of function evaluations is chosen as 10^5 times the search space dimension.

Keywords

Benchmarking, Pure Random Search, Monte-Carlo, Black-box optimization, Evolutionary computation

1. INTRODUCTION

The pure random search, first proposed by Brooks in 1958 [1] is the most simple stochastic search algorithm that consists in sampling each search point independently in the search domain and keeping the best solution found.

2. METHODS

We have used a uniform sampling in $[-5,5]^D$, where D denotes the dimension of the search space. The experiments according to [3] on the benchmark functions given in [2, 4] have been conducted using both a C-code and Matlab code. The algorithm implementation in Matlab is given in Figure 1. A maximum of $10^5 \times D$ function evaluations has been used. The simulations for 2; 5; 10 and 20 D were done with the C-code and took 2 hours and a half. The 40 D experiments were done at the same time using the Matlab code and took 17 hours.

No parameter tuning was done and the crafting effort CrE [3] is computed to zero.

[*]Camera-ready paper due April 17th.

[†]Dr. Auger insisted his name be first.

3. RESULTS AND DISCUSSION

Results from experiments according to [3] on the benchmark functions given in [2, 4] are presented in Figures 3 and 4 and in Table 1.

Since we use a uniform sampling in the search domain, we obtain as a by-product of the results an estimate of the volume of the sublevel sets: the sublevel sets of a function $f : \mathbb{R}^D \to \mathbb{R}$ are defined as $S_c = \{x \in \mathbb{R}^D | f(x) \leq c\}$ for c spanning $\mathbb{R}$. If S_c is a subset of $[-5,5]^D$, the hitting time T_c (assuming infinite horizon) of the sublevel set S_c is distributed according to a geometric random variable of parameter $p_c = Vol(S_c)/Vol([-5,5]^D)$. The expected running time ERT(Δf) estimates the expected value of $T_{\Delta f}$ (see Figure 2), that equals $1/p_c$ since $T_{\Delta f}$ is a geometric random variable. And thus ERT(Δf) gives the ratio between $Vol([-5,5]^D)$ and $Vol(S_c)$.

4. CPU TIMING EXPERIMENT

For the timing experiment the Pure Random Search was run with a maximum of $10^5 \times D$ function evaluations and restarted until 30 seconds has passed (according to Figure 2 in [3]). The experiments have been conducted with an Intel Core 2 Duo 2.53 GHz under Mac OS X Version 10.5.6 using the C-code provided. The time per function evaluation was 2.0; 2.3; 2.8; 4.2; 6.9 times 10^{-7} seconds in dimensions 2; 3; 5; 10; 20; 40 respectively.

5. CONCLUSION

We have presented the results of the Pure Random Search, a non-adaptive algorithm, that does not use information gathered during search for guiding its next steps. Those results provide a baseline comparison that every adaptive algorithm should outperform.

Acknowledgments

The first two authors would like to acknowledge the great and hard work of the BBOB team with special kudos to Steffen Finck and (of course!) to the natural leader of the team Nikolaus Hansen.

6. REFERENCES

[1] S. H. Brooks. A discussion of random methods for seeking maxima. *Operations Research*, 6:244– 251, 1958.

	f_1 in 5-D, N=15, mFE=500000					**f_1 in 20-D**, N=15, mFE=2000000				
Δf	#	ERT	10%	90%	RT$_\text{S}$	#	ERT	10%	90%	RT$_\text{S}$
10	15	8.7e1	6.2e1	1.0e2	8.7e1	0	*33e+0*	*29e+0*	*40e+0*	1.1e6
1	15	2.5e4	1.8e4	3.3e4	2.5e4	.	.	.	.	.
1e−1	2	3.5e6	3.3e6	3.7e6	4.2e5	.	.	.	.	.
1e−3	0	*19e−2*	*86e−3*	*42e−2*	1.8e5	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	f_2 in 5-D, N=15, mFE=500000					**f_2 in 20-D**, N=15, mFE=2000000				
Δf	#	ERT	10%	90%	RT$_\text{S}$	#	ERT	10%	90%	RT$_\text{S}$
10	0	*26e+1*	*62e+0*	*55e+1*	2.5e5	0	*21e+4*	*16e+4*	*28e+4*	1.0e6
1	.	.	.	.	.	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	f_3 in 5-D, N=15, mFE=500000					**f_3 in 20-D**, N=15, mFE=2000000				
Δf	#	ERT	10%	90%	RT$_\text{S}$	#	ERT	10%	90%	RT$_\text{S}$
10	4	1.6e6	1.4e6	1.8e6	4.4e5	0	*30e+1*	*26e+1*	*34e+1*	1.3e6
1	0	*13e+0*	*63e−1*	*16e+0*	2.8e5	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	f_4 in 5-D, N=15, mFE=500000					**f_4 in 20-D**, N=15, mFE=2000000				
Δf	#	ERT	10%	90%	RT$_\text{S}$	#	ERT	10%	90%	RT$_\text{S}$
10	1	7.4e6	7.4e6	7.5e6	5.0e5	0	*39e+1*	*34e+1*	*43e+1*	1.0e6
1	0	*17e+0*	*10e+0*	*21e+0*	4.0e5	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	f_5 in 5-D, N=15, mFE=500000					**f_5 in 20-D**, N=15, mFE=2000000				
Δf	#	ERT	10%	90%	RT$_\text{S}$	#	ERT	10%	90%	RT$_\text{S}$
10	15	4.9e4	3.5e4	6.1e4	4.9e4	0	*13e+1*	*13e+1*	*14e+1*	7.9e5
1	0	*54e−1*	*42e−1*	*71e−1*	3.2e5	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	f_6 in 5-D, N=15, mFE=500000					**f_6 in 20-D**, N=15, mFE=2000000				
Δf	#	ERT	10%	90%	RT$_\text{S}$	#	ERT	10%	90%	RT$_\text{S}$
10	15	2.5e4	1.8e4	3.4e4	2.5e4	0	*69e+2*	*37e+1*	*93e+3*	7.1e5
1	1	7.3e6	7.0e6	7.5e6	5.0e5	.	.	.	.	.
1e−1	0	*28e−1*	*14e−1*	*38e−1*	2.2e5	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	f_7 in 5-D, N=15, mFE=500000					**f_7 in 20-D**, N=15, mFE=2000000				
Δf	#	ERT	10%	90%	RT$_\text{S}$	#	ERT	10%	90%	RT$_\text{S}$
10	15	7.9e2	3.4e2	1.1e3	7.9e2	0	*14e+1*	*11e+1*	*18e+1*	7.1e5
1	9	5.0e5	3.8e5	5.7e5	3.0e5	.	.	.	.	.
1e−1	0	*90e−2*	*45e−2*	*14e−1*	2.2e5	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	f_8 in 5-D, N=15, mFE=500000					**f_8 in 20-D**, N=15, mFE=2000000				
Δf	#	ERT	10%	90%	RT$_\text{S}$	#	ERT	10%	90%	RT$_\text{S}$
10	3	2.2e6	2.0e6	2.4e6	5.0e5	0	*14e+3*	*91e+2*	*17e+3*	5.0e5
1	0	*16e+0*	*86e−1*	*26e+0*	1.8e5	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	f_9 in 5-D, N=15, mFE=500000					**f_9 in 20-D**, N=15, mFE=2000000				
Δf	#	ERT	10%	90%	RT$_\text{S}$	#	ERT	10%	90%	RT$_\text{S}$
10	3	2.1e6	1.9e6	2.3e6	3.3e5	0	*10e+3*	*79e+2*	*14e+3*	1.0e6
1	0	*17e+0*	*74e−1*	*23e+0*	2.5e5	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	f_{10} in 5-D, N=15, mFE=500000					**f_{10} in 20-D**, N=15, mFE=2000000				
Δf	#	ERT	10%	90%	RT$_\text{S}$	#	ERT	10%	90%	RT$_\text{S}$
10	0	*27e+1*	*13e+1*	*54e+1*	1.8e5	0	*19e+4*	*13e+4*	*25e+4*	7.1e5
1	.	.	.	.	.	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	f_{11} in 5-D, N=15, mFE=500000					**f_{11} in 20-D**, N=15, mFE=2000000				
Δf	#	ERT	10%	90%	RT$_\text{S}$	#	ERT	10%	90%	RT$_\text{S}$
10	15	7.2e4	6.3e4	9.0e4	7.2e4	0	*81e+0*	*63e+0*	*90e+0*	1.1e6
1	0	*43e−1*	*23e−1*	*64e−1*	1.8e5	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	f_{12} in 5-D, N=15, mFE=500000					**f_{12} in 20-D**, N=15, mFE=2000000				
Δf	#	ERT	10%	90%	RT$_\text{S}$	#	ERT	10%	90%	RT$_\text{S}$
10	0	*68e+3*	*24e+3*	*98e+3*	2.5e5	0	*38e+6*	*31e+6*	*47e+6*	1.3e6
1	.	.	.	.	.	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	f_{13} in 5-D, N=15, mFE=500000					**f_{13} in 20-D**, N=15, mFE=2000000				
Δf	#	ERT	10%	90%	RT$_\text{S}$	#	ERT	10%	90%	RT$_\text{S}$
10	0	*52e+0*	*26e+0*	*70e+0*	2.8e5	0	*11e+2*	*10e+2*	*12e+2*	1.3e6
1	.	.	.	.	.	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	f_{14} in 5-D, N=15, mFE=500000					**f_{14} in 20-D**, N=15, mFE=2000000				
Δf	#	ERT	10%	90%	RT$_\text{S}$	#	ERT	10%	90%	RT$_\text{S}$
10	15	1.7e1	1.5e1	2.1e1	1.7e1	5	5.2e6	4.7e6	5.6e6	1.8e6
1	15	9.1e3	7.3e3	1.3e4	9.1e3	0	*11e+0*	*89e−1*	*12e+0*	1.0e6
1e−1	1	7.3e6	7.2e6	7.5e6	3.4e5	.	.	.	.	.
1e−3	0	*22e−2*	*11e−2*	*31e−2*	2.5e5	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	f_{15} in 5-D, N=15, mFE=500000					**f_{15} in 20-D**, N=15, mFE=2000000				
Δf	#	ERT	10%	90%	RT$_\text{S}$	#	ERT	10%	90%	RT$_\text{S}$
10	4	1.6e6	1.3e6	1.8e6	4.4e5	0	*31e+1*	*28e+1*	*35e+1*	5.6e5
1	0	*12e+0*	*88e−1*	*15e+0*	2.5e5	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	f_{16} in 5-D, N=15, mFE=500000					**f_{16} in 20-D**, N=15, mFE=2000000				
Δf	#	ERT	10%	90%	RT$_\text{S}$	#	ERT	10%	90%	RT$_\text{S}$
10	15	5.4e2	4.7e2	6.4e2	5.4e2	0	*13e+0*	*11e+0*	*16e+0*	8.9e5
1	12	3.4e5	3.1e5	4.1e5	3.1e5	.	.	.	.	.
1e−1	0	*67e−2*	*31e−2*	*11e−1*	3.2e5	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	f_{17} in 5-D, N=15, mFE=500000					**f_{17} in 20-D**, N=15, mFE=2000000				
Δf	#	ERT	10%	90%	RT$_\text{S}$	#	ERT	10%	90%	RT$_\text{S}$
10	15	1.8e1	1.4e1	2.2e1	1.8e1	15	4.1e3	2.3e3	5.6e3	4.1e3
1	15	1.8e5	1.3e5	2.1e5	1.8e5	0	*60e−1*	*47e−1*	*67e−1*	7.9e5
1e−1	0	*80e−2*	*57e−2*	*94e−2*	2.5e5	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	f_{18} in 5-D, N=15, mFE=500000					**f_{18} in 20-D**, N=15, mFE=2000000				
Δf	#	ERT	10%	90%	RT$_\text{S}$	#	ERT	10%	90%	RT$_\text{S}$
10	15	1.4e3	1.2e3	2.3e3	1.4e3	0	*21e+0*	*17e+0*	*23e+0*	1.0e6
1	0	*26e−1*	*19e−1*	*32e−1*	2.8e5	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	f_{19} in 5-D, N=15, mFE=500000					**f_{19} in 20-D**, N=15, mFE=2000000				
Δf	#	ERT	10%	90%	RT$_\text{S}$	#	ERT	10%	90%	RT$_\text{S}$
10	15	3.3e1	1.6e1	5.6e1	3.3e1	15	4.6e5	3.2e5	5.9e5	4.6e5
1	15	1.7e5	1.3e5	2.0e5	1.7e5	0	*91e−1*	*81e−1*	*97e−1*	1.3e6
1e−1	0	*60e−2*	*39e−2*	*75e−2*	3.2e5	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	f_{20} in 5-D, N=15, mFE=500000					**f_{20} in 20-D**, N=15, mFE=2000000				
Δf	#	ERT	10%	90%	RT$_\text{S}$	#	ERT	10%	90%	RT$_\text{S}$
10	15	4.8e2	3.5e2	5.5e2	4.8e2	0	*30e+2*	*24e+2*	*42e+2*	8.9e5
1	1	7.2e6	6.5e6	7.2e6	5.0e5	.	.	.	.	.
1e−1	0	*15e−1*	*11e−1*	*17e−1*	1.6e5	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	f_{21} in 5-D, N=15, mFE=500000					**f_{21} in 20-D**, N=15, mFE=2000000				
Δf	#	ERT	10%	90%	RT$_\text{S}$	#	ERT	10%	90%	RT$_\text{S}$
10	15	1.4e2	9.0e1	2.1e2	1.4e2	0	*33e+0*	*24e+0*	*41e+0*	1.4e6
1	15	1.3e4	7.9e3	1.7e4	1.3e4	.	.	.	.	.
1e−1	11	4.4e5	3.5e5	5.1e5	3.5e5	.	.	.	.	.
1e−3	0	*53e−3*	*13e−3*	*15e−2*	2.8e5	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	f_{22} in 5-D, N=15, mFE=500000					**f_{22} in 20-D**, N=15, mFE=2000000				
Δf	#	ERT	10%	90%	RT$_\text{S}$	#	ERT	10%	90%	RT$_\text{S}$
10	15	2.9e2	1.3e2	4.2e2	2.9e2	0	*52e+0*	*36e+0*	*58e+0*	8.9e5
1	15	1.5e4	1.2e4	2.7e4	1.5e4	.	.	.	.	.
1e−1	8	6.0e5	5.8e5	7.3e5	3.9e5	.	.	.	.	.
1e−3	0	*94e−3*	*13e−3*	*14e−2*	2.2e5	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	f_{23} in 5-D, N=15, mFE=500000					**f_{23} in 20-D**, N=15, mFE=2000000				
Δf	#	ERT	10%	90%	RT$_\text{S}$	#	ERT	10%	90%	RT$_\text{S}$
10	15	6.5e0	4.9e0	8.5e0	6.5e0	15	5.4e0	4.1e0	7.1e0	5.4e0
1	15	1.9e4	1.2e4	2.5e4	1.9e4	0	*15e−1*	*10e−1*	*16e−1*	5.6e5
1e−1	0	*50e−2*	*39e−2*	*60e−2*	2.5e5	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	f_{24} in 5-D, N=15, mFE=500000					**f_{24} in 20-D**, N=15, mFE=2000000				
Δf	#	ERT	10%	90%	RT$_\text{S}$	#	ERT	10%	90%	RT$_\text{S}$
10	1	7.2e6	6.8e6	7.5e6	5.0e5	0	*28e+1*	*27e+1*	*30e+1*	1.1e6
1	0	*14e+0*	*12e+0*	*16e+0*	2.2e5	.	.	.	.	.
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 $f_{\text{opt}} + \Delta f$ (ERT, see Figure 3); the 10%-tile and 90%-tile of the bootstrap distribution of ERT; the total number of function evaluations in unsuccessful trials divided either by the number of successful trials or by 1, if none was successful (RT$_{\text{US}}$). If $f_{\text{opt}} + \Delta f$ 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 1: Pure Random Search in Matlab. At each iteration (iter), 200 points are sampled and stored in a matrix of size $D \times 200$ so as to reduce loops and function calls within Matlab.

```
function MY_OPTIMIZER(FUN, DIM, ftarget, maxfunevals)
% MY_OPTIMIZER(FUN, DIM, ftarget, maxfunevals)
% samples new points uniformly randomly in [-5,5]^DIM
% and evaluates them on FUN until ftarget of maxfunevals
% is reached, or until 1e8 * DIM fevals are conducted.
% Relies on FUN to keep track of the best point.

  maxfunevals = min(1e8 * DIM, maxfunevals);
  popsize = min(maxfunevals, 200);
  for iter = 1:ceil(maxfunevals/popsize)
    feval(FUN, 10 * rand(DIM, popsize) - 5);
    if feval(FUN, 'fbest') < ftarget  % task achieved
      break;
    end
    % if useful, modify more options here for next start
  end
```

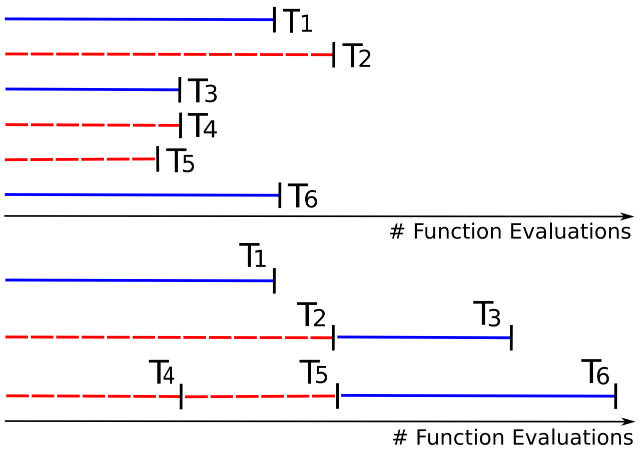

Figure 2: Illustration that $ERT(\Delta f)$ estimates the expected hitting time of an algorithm restarted until success (assuming infinite horizon): among 6 runs of the same algorithm A, the 1st, 3rd and 6th are successful while the 2nd, 4th and 5th are unsuccessful and thus T_1, $T_2 + T_3$ and $T_4 + T_5 + T_6$ are 3 instances of the algorithm restart-A (i.e., algorithm A restarted until success). Thus an estimate of the expected hitting time of restart-A is $(T_1 + (T_2 + T_3 + T_4) + (T_4 + T_5 + T_6))/3$, i.e., total number of function evaluations divided by number of successes of algorithm A, i.e., $ERT(\Delta f)$. In the case where algorithm A is the pure random search, the picture is simpler because unsuccessful runs always reach the maximum number of evaluations and thus the 2nd, 4th and 5th runs have the same length. T_1, $T_2 + T_3$ and $T_4 + T_5 + T_6$ represent then 3 instances of the pure random search that would be run with infinite horizon until a success is reached and $ERT(\Delta f)$ estimates thus the expected hitting time of the pure random search with infinite horizon.

[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.

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

[4] 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.

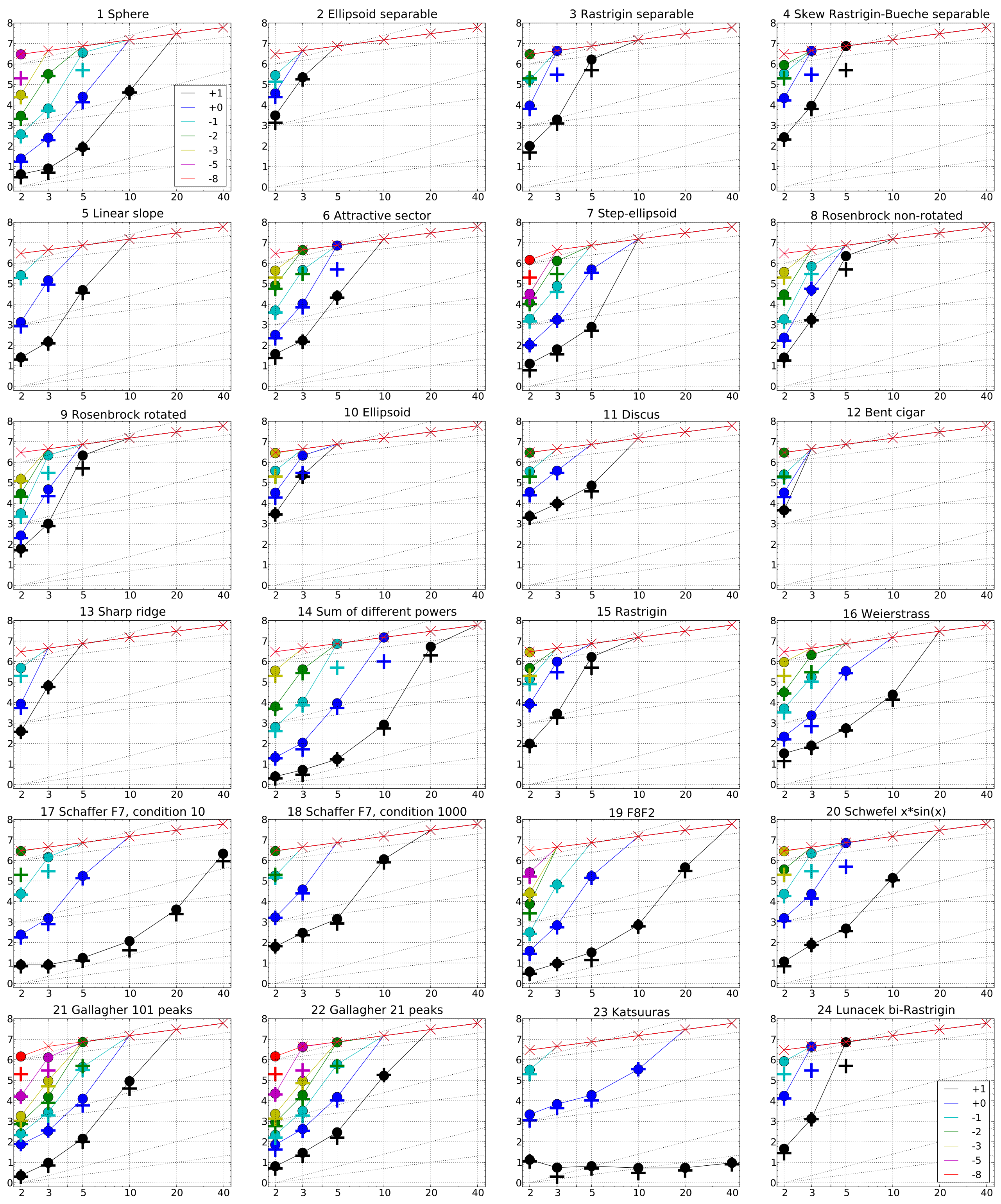

Figure 3: Expected Running Time (ERT, ●) and number of function evaluations of the median trial (+) to reach $f_{\mathrm{opt}} + \Delta f$, shown for $\Delta f = 10, 1, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-5}, 10^{-8}$ (the exponent is given in the legend of f_1 and f_{24}) versus dimension in log-log presentation. 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 $\#\mathrm{FEs}(-\infty)$. Annotated numbers on the ordinate are decimal logarithms. Additional grid lines show linear and quadratic scaling.

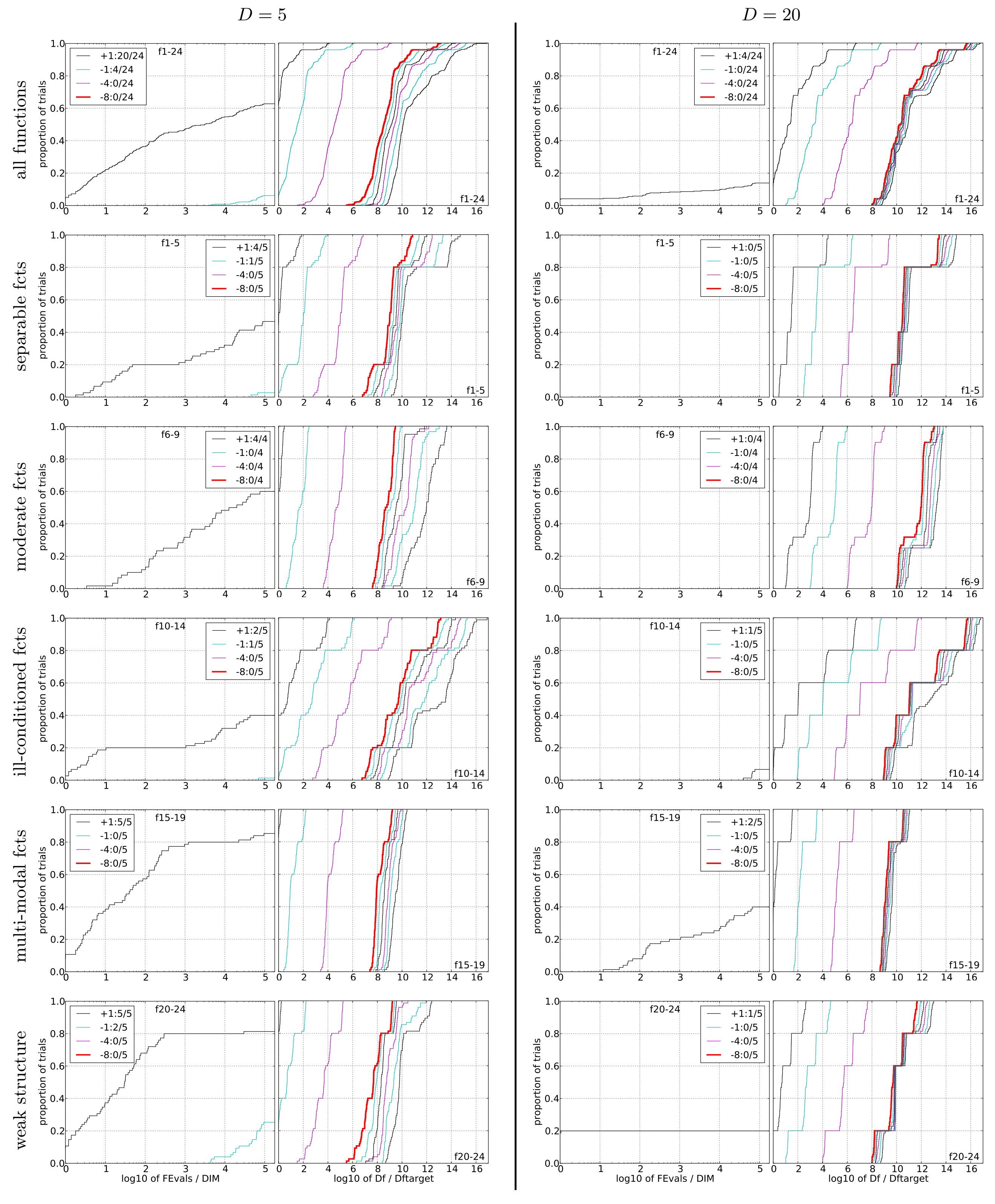

Figure 4: Empirical cumulative distribution functions (ECDFs), plotting the fraction of trials versus running time (left) or Δf. Left subplots: ECDF of the running time (number of function evaluations), divided by search space dimension D, to fall below $f_{\text{opt}} + \Delta f$ with $\Delta f = 10^k$, where k is the first value in the legend. Right subplots: ECDF of the best achieved Δf divided by 10^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.