Black-Box Optimization Benchmarking for Noiseless Function Testbed using PSO_Bounds

Mohammed El-Abd
University of Waterloo
Waterloo, Ontario, Canada
mhelabd@pami.uwaterloo.ca

Mohamed S. Kamel
University of Waterloo
Waterloo, Ontario, Canada
mkamel@pami.uwaterloo.ca

ABSTRACT

This paper benchmarks the particle swarm optimizer with adaptive bounds algorithm (PSO_Bounds) on the noisefree BBOB 2009 testbed. The algorithm is further augmented with a simple re-initialization mechanism that is invoked if the bounds tend to overlap.

Categories and Subject Descriptors

G.1.6 [**Numerical Analysis**]: OptimizationGlobal Optimization, Unconstrained Optimization; F.2.1 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems

General Terms

Algorithms

Keywords

Benchmarking, Black-box optimization, Evolutionary computation, Particle Swarm Optimization, Hybrid Algorithms

1. INTRODUCTION

Particle Swarm Optimization (PSO) [2, 7] is an optimization method widely used to solve continuous nonlinear functions. It is a stochastic optimization technique that emerged from simulations of the birds flocking and fish schooling behaviors.

The algorithm used in this work incorporates the principles of population-based incremental learning (PBIL) [1] into PSO.

2. ALGORITHM PRESENTATION

A population-based incremental learning (PBIL) approach for continuous search spaces was proposed in [8]. The algorithm explored the search space by dividing the domain of each gene into two equal intervals referred to as the *low* and *high* intervals. A probability h_d, which is initially set to 0.5, is the probability of dimension number d being in the *high* interval as shown:

$$x_d \in [a, b], h_d = Probability(x_d > \frac{a+b}{2}) \tag{1}$$

After each generation, this distribution was updated according to the dimension values of the best individual using the following formula:

$$p = \begin{cases} 0 & \texttt{if} \quad x_d^{best} < \frac{a+b}{2} \\ 1 & otherwise \end{cases} \tag{2}$$

$$h_d^{t+1} = (1-\alpha) * h_d^t + \alpha * p$$

where α is the *relaxation factor* and t is the iteration number. If h_d gets below h_{dmin} or above h_{dmax}, the population gets re-sampled in the corresponding interval, $[a, \frac{a+b}{2}]$ or $[\frac{a+b}{2}, b]$ respectively.

El-Abd and Kamel [3] introduced PSO_Bounds, in which the concepts of PBIL are integrated into PSO. At the beginning of the algorithm, the particles are initialized in the predefined domain. After every iteration, the probability h_d of each dimension d is adjusted according to the probability of the value associated with this dimension being in the *high* interval of the defined domain. To prevent premature convergence, this probability is calculated using information from all the particles and not only *gbest*. Hence, the original equations of PBIL are changed as follows:

$$p_{id}^t = \begin{cases} 0 & \texttt{if} \quad pbest_{id}^t < \frac{a+b}{2} \\ 1 & otherwise \end{cases}$$

$$p_d^t = \frac{\sum_i^n p_{id}^t}{n}$$

$$h_d^{t+1} = (1-\alpha) * h_d^t + \alpha * p_d^t \tag{3}$$

where $i \in \{1..n\}$ and n is the number of particles, t is the iteration number, and d is the dimension.

In PBIL, the probabilities were updated using the value of the best individual, which is analogous to the current position of the particles in PSO. However, in our implementation, we use the values of *pbest* instead beacause these values reflect the best experience of the swarm and would guide the search towards better solutions. When h_d becomes specific enough, the domain of dimension d is adjusted accordingly, and h_d is re-initialized to 0.5. In this model, different dimensions may end up having different domains and different velocity bounds which do not happen in normal PSO.

In order to overcome the problem of the bounds overlapping, thus preventing further particle movement, the width

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

of the adjusted bounds is taken into consideration if the algorithm needs to adjust these bounds for a certain dimension d. If the width drops below a predetermined percentage of the initial search domain width, controlled by the parameter T, the bounds are reset to the initial bounds of the search space and the velocity component is also re-initialized. This will allow the particles to move in different directions and in large steps in the next iteration while still taking the old *pbest* and *gbest* information into account, hence, not losing any previous information gathered during the search.

Fig. 1 shows the MATLAB code for PSO_Bounds where x_{dmin} and x_{dmax} refer to the minimum and maximum bounds for dimension d while v_{dmin} and v_{dmax} refer to the velocity bounds.

3. RESULTS

The parameters are set as $c1 = c2 = 2$, w linearly decreases from 0.9 to 0.1 with the iterations 40 particles are used, $h_{dmin} = 0.2$, $h_{dmax} = 0.8$, $\alpha = 0.05$ and $T = 0.0001$.

The simulations for 2; 3; 5; 10 and 20 D were done with the MATLAB-code and took 12 hours and 15 minutes. No parameter tuning was done and the crafting effort CrE [6] is computed to zero.

Results from experiments according to [5] on the benchmark functions given in [4, 6] are presented in Figures 2 and 3 and in Table 1.

4. CPU TIMING EXPERIMENT

For the timing experiment PSO_Bounds was run with a maximum of 10^4 function evaluations and restarted until 30 seconds has passed (according to Figure 2 in [6]). The experiments have been conducted with an Intel Core 2 Quad 2.4 GHz under Windows XP using the MATLAB-code provided. The time per function evaluation was 1.2; 1.6; 1.6; 1.8; 2.1 times 10^{-5} seconds in dimensions 2; 3; 5; 10; 20 respectively.

5. REFERENCES

[1] S. Baluja. Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Technical Report CMU-CS-94-163, School of Computer Science, Carnegie Mellon University, 1994.

[2] R. C. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Proc. of the 6th International Symposium on Micro Machine and Human Science*, pages 39–43, 1995.

[3] M. El-Abd and M. S. Kamel. Particle swarm optimization with varying bounds. In *IEEE Congress on Evolutionary Computation*, pages 4757–4761, 2007.

[4] 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.

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

[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.

[7] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *Proc. of IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948, 1995.

[8] I. Servet, L. Trave-Massuyes, and D. Stern. Telephone network traffic overloading diagnosis and evolutionary computation technique. In *Artificial Evolution. Springer-Verlag, LNCS 1363*, pages 137–144, 1997.

Δf	f_1 **in 5-D**, N=15, mFE=65440					f_1 **in 20-D**, N=15, mFE=1.14e6				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	15	4.2e1	3.4e1	4.9e1	4.2e1	15	5.3e3	4.2e3	6.5e3	5.3e3
1	15	5.0e2	3.9e2	6.2e2	5.0e2	15	6.4e4	5.2e4	7.6e4	6.4e4
1e−1	15	2.6e3	2.4e3	2.9e3	2.6e3	15	9.2e4	8.3e4	1.0e5	9.2e4
1e−3	15	8.9e3	8.1e3	9.8e3	8.9e3	15	1.4e5	1.3e5	1.5e5	1.4e5
1e−5	15	1.6e4	1.5e4	1.7e4	1.6e4	15	1.9e5	1.9e5	2.0e5	1.9e5
1e−8	15	3.7e4	3.2e4	4.1e4	3.7e4	15	8.8e5	8.5e5	9.2e5	8.8e5

Δf	f_2 **in 5-D**, N=15, mFE=148400					f_2 **in 20-D**, N=15, mFE=2.00e6				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	15	1.3e4	1.1e4	1.5e4	1.3e4	15	1.4e5	1.2e5	1.6e5	1.4e5
1	15	1.7e4	1.5e4	1.9e4	1.7e4	15	2.0e5	1.9e5	2.2e5	2.0e5
1e−1	15	2.3e4	2.1e4	2.5e4	2.3e4	15	3.2e5	2.9e5	3.5e5	3.2e5
1e−3	15	3.6e4	3.3e4	3.9e4	3.6e4	15	6.8e5	6.4e5	7.3e5	6.8e5
1e−5	15	7.9e4	7.5e4	8.3e4	7.9e4	15	8.9e5	8.2e5	9.5e5	8.9e5
1e−8	15	1.2e5	1.2e5	1.3e5	1.2e5	14	1.2e6	1.1e6	1.4e6	1.2e6

Δf	f_3 **in 5-D**, N=15, mFE=500000					f_3 **in 20-D**, N=15, mFE=2.00e6				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	15	5.4e3	4.7e3	6.2e3	5.4e3	13	6.0e5	3.8e5	8.5e5	5.6e5
1	15	4.3e4	3.8e4	4.8e4	4.3e4	10	1.5e6	1.0e6	2.1e6	9.2e5
1e−1	15	6.2e4	4.9e4	7.7e4	6.2e4	7	2.7e6	2.0e6	4.1e6	1.6e6
1e−3	14	1.1e5	6.7e4	1.5e5	1.0e5	7	2.8e6	2.0e6	4.1e6	1.6e6
1e−5	14	1.1e5	7.7e4	1.6e5	1.1e5	7	3.1e6	2.3e6	4.6e6	1.7e6
1e−8	14	1.8e5	1.4e5	2.1e5	1.7e5	7	3.3e6	2.5e6	5.0e6	1.8e6

Δf	f_4 **in 5-D**, N=15, mFE=500000					f_4 **in 20-D**, N=15, mFE=2.00e6				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	15	6.5e3	5.6e3	7.5e3	6.5e3	12	8.8e5	5.8e5	1.3e6	6.7e5
1	15	4.8e4	3.9e4	5.8e4	4.8e4	9	2.2e6	1.6e6	3.1e6	1.3e6
1e−1	14	1.1e5	7.1e4	1.5e5	1.0e5	9	2.3e6	1.7e6	3.2e6	1.4e6
1e−3	12	2.0e5	1.4e5	2.8e5	1.9e5	8	2.8e6	2.0e6	4.1e6	1.5e6
1e−5	12	2.1e5	1.5e5	2.8e5	2.0e5	8	2.9e6	2.2e6	4.4e6	1.5e6
1e−8	12	2.7e5	2.1e5	3.4e5	2.4e5	7	3.5e6	2.6e6	5.5e6	1.7e6

Δf	f_5 **in 5-D**, N=15, mFE=280					f_5 **in 20-D**, N=15, mFE=20600				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	15	9.2e1	8.4e1	1.0e2	9.2e1	15	6.3e3	4.3e3	8.3e3	6.3e3
1	15	1.5e2	1.3e2	1.6e2	1.5e2	15	6.4e3	4.4e3	8.4e3	6.4e3
1e−1	15	1.6e2	1.4e2	1.7e2	1.6e2	15	6.5e3	4.4e3	8.4e3	6.5e3
1e−3	15	1.6e2	1.4e2	1.8e2	1.6e2	15	6.5e3	4.5e3	8.5e3	6.5e3
1e−5	15	1.6e2	1.4e2	1.8e2	1.6e2	15	6.5e3	4.5e3	8.5e3	6.5e3
1e−8	15	1.6e2	1.4e2	1.8e2	1.6e2	15	6.5e3	4.5e3	8.5e3	6.5e3

Δf	f_6 **in 5-D**, N=15, mFE=171360					f_6 **in 20-D**, N=15, mFE=2.00e6				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	15	1.6e3	9.1e2	2.4e3	1.6e3	15	1.5e5	1.4e5	1.7e5	1.5e5
1	15	1.1e4	7.0e3	1.4e4	1.1e4	15	2.6e5	2.4e5	2.7e5	2.6e5
1e−1	15	2.4e4	1.8e4	3.0e4	2.4e4	15	3.5e5	3.4e5	3.7e5	3.5e5
1e−3	15	5.7e4	4.6e4	6.7e4	5.7e4	15	5.0e5	4.8e5	5.2e5	5.0e5
1e−5	15	8.1e4	6.9e4	9.3e4	8.1e4	15	7.7e5	7.5e5	7.9e5	7.7e5
1e−8	15	1.4e5	1.4e5	1.5e5	1.4e5	10	2.6e6	2.1e6	3.4e6	1.8e6

Δf	f_7 **in 5-D**, N=15, mFE=500000					f_7 **in 20-D**, N=15, mFE=2.00e6				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	15	2.2e2	1.8e2	2.6e2	2.2e2	2	1.3e7	7.0e6	>3e7	2.0e6
1	15	4.1e3	3.2e3	4.9e3	4.1e3	0	*22e+0*	*48e−1*	*49e+0*	7.9e4
1e−1	11	1.9e5	9.6e4	3.1e5	1.4e5	.	.	.	.	.
1e−3	11	2.1e5	1.1e5	3.2e5	1.5e5	.	.	.	.	.
1e−5	11	2.1e5	1.1e5	3.2e5	1.5e5	.	.	.	.	.
1e−8	11	2.1e5	1.1e5	3.2e5	1.5e5	.	.	.	.	.

Δf	f_8 **in 5-D**, N=15, mFE=500000					f_8 **in 20-D**, N=15, mFE=2.00e6				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	15	2.2e3	1.9e3	2.5e3	2.2e3	15	1.1e6	1.0e6	1.1e6	1.1e6
1	15	1.3e5	1.1e5	1.5e5	1.3e5	15	1.7e6	1.6e6	1.8e6	1.7e6
1e−1	15	3.1e5	2.9e5	3.3e5	3.1e5	4	7.3e6	4.7e6	1.5e7	1.8e6
1e−3	12	5.5e5	4.7e5	6.7e5	4.4e5	0	*15e−2*	*54e−3*	*32e−2*	2.0e6
1e−5	0	*12e−5*	*29e−6*	*26e−3*	4.5e5	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	f_9 **in 5-D**, N=15, mFE=500000					f_9 **in 20-D**, N=15, mFE=2.00e6				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	15	7.6e3	5.9e3	9.4e3	7.6e3	15	1.2e6	1.2e6	1.2e6	1.2e6
1	13	1.9e5	1.3e5	2.5e5	1.6e5	1	3.0e7	1.5e7	>3e7	2.0e6
1e−1	13	3.4e5	3.0e5	4.1e5	3.0e5	0	*20e−1*	*14e−1*	*42e−1*	2.0e6
1e−3	6	1.2e6	8.5e5	1.8e6	4.9e5	.	.	.	.	.
1e−5	1	7.4e6	3.6e6	>7e6	5.0e5	.	.	.	.	.
1e−8	1	7.5e6	3.7e6	>7e6	5.0e5	.	.	.	.	.

Δf	f_{10} **in 5-D**, N=15, mFE=500000					f_{10} **in 20-D**, N=15, mFE=2.00e6				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	5	1.3e6	9.0e5	2.3e6	4.6e5	0	*41e+1*	*18e+1*	*86e+1*	2.0e6
1	1	7.5e6	3.7e6	>7e6	5.0e5	.	.	.	.	.
1e−1	0	*20e+0*	*15e−1*	*60e+0*	4.5e5	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	f_{11} **in 5-D**, N=15, mFE=500000					f_{11} **in 20-D**, N=15, mFE=2.00e6				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	15	6.1e4	3.1e4	9.5e4	6.1e4	15	5.8e5	5.0e5	6.4e5	5.8e5
1	12	2.9e5	2.2e5	3.8e5	2.4e5	15	9.8e5	8.8e5	1.1e6	9.8e5
1e−1	7	7.7e5	5.6e5	1.2e6	3.9e5	14	1.4e6	1.2e6	1.6e6	1.3e6
1e−3	5	1.3e6	8.7e5	2.2e6	4.7e5	6	4.7e6	3.4e6	7.2e6	1.9e6
1e−5	3	2.2e6	1.4e6	6.7e6	5.0e5	2	1.5e7	7.5e6	>3e7	2.0e6
1e−8	3	2.3e6	1.4e6	7.0e6	5.0e5	0	*12e−4*	*68e−7*	*34e−3*	2.0e6

Δf	f_{12} **in 5-D**, N=15, mFE=500000					f_{12} **in 20-D**, N=15, mFE=2.00e6				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	11	2.0e5	1.1e5	2.9e5	2.0e5	12	7.3e5	4.2e5	1.1e6	5.4e5
1	7	6.6e5	5.0e5	9.2e5	4.3e5	4	5.8e6	3.9e6	1.2e7	2.0e6
1e−1	3	2.0e6	1.1e6	6.6e6	3.5e5	2	1.4e7	7.2e6	>3e7	2.0e6
1e−3	0	*46e−1*	*13e−3*	*19e+0*	4.5e5	0	*64e−1*	*92e−3*	*30e+0*	2.0e6
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	f_{13} **in 5-D**, N=15, mFE=500000					f_{13} **in 20-D**, N=15, mFE=2.00e6				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	14	4.6e4	9.8e3	8.8e4	4.5e4	9	1.5e6	9.8e5	2.1e6	1.2e6
1	8	4.6e5	2.8e5	7.8e5	2.0e5	3	8.3e6	4.6e6	2.5e7	1.4e6
1e−1	4	1.4e6	7.8e5	3.3e6	1.5e5	0	*50e−1*	*47e−2*	*51e+0*	1.8e6
1e−3	1	7.1e6	3.4e6	>7e6	5.0e5	.	.	.	.	.
1e−5	0	*81e−2*	*95e−4*	*69e−1*	2.2e5	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	f_{14} **in 5-D**, N=15, mFE=500000					f_{14} **in 20-D**, N=15, mFE=2.00e6				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	15	1.9e1	1.1e1	2.7e1	1.9e1	15	1.7e3	1.5e3	1.9e3	1.7e3
1	15	5.0e2	4.2e2	5.9e2	5.0e2	15	2.5e4	2.0e4	2.9e4	2.5e4
1e−1	15	2.6e3	2.4e3	2.8e3	2.6e3	15	5.3e4	4.7e4	5.8e4	5.3e4
1e−3	15	1.9e4	1.6e4	2.2e4	1.9e4	15	3.5e5	3.4e5	3.7e5	3.5e5
1e−5	15	1.0e5	8.6e4	1.2e5	1.0e5	0	*78e−6*	*67e−6*	*94e−6*	2.0e6
1e−8	0	*14e−7*	*45e−8*	*29e−7*	4.5e5	.	.	.	.	.

Δf	f_{15} **in 5-D**, N=15, mFE=500000					f_{15} **in 20-D**, N=15, mFE=2.00e6				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	13	8.7e4	1.4e4	1.5e5	8.6e4	0	*51e+0*	*25e+0*	*83e+0*	1.1e6
1	5	1.1e6	7.7e5	2.1e6	4.5e5	.	.	.	.	.
1e−1	0	*20e−1*	*99e−2*	*11e+0*	1.6e5	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	f_{16} **in 5-D**, N=15, mFE=500000					f_{16} **in 20-D**, N=15, mFE=2.00e6				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	15	2.9e2	2.1e2	3.7e2	2.9e2	15	1.8e5	1.4e5	2.2e5	1.8e5
1	15	2.2e4	1.8e4	2.7e4	2.2e4	1	2.8e7	1.3e7	>3e7	2.0e6
1e−1	9	3.7e5	2.4e5	5.5e5	2.4e5	0	*25e−1*	*16e−1*	*45e−1*	1.8e6
1e−3	2	3.3e6	1.8e6	>7e6	5.0e5	.	.	.	.	.
1e−5	0	*16e−3*	*18e−5*	*54e−2*	2.2e5	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	f_{17} **in 5-D**, N=15, mFE=500000					f_{17} **in 20-D**, N=15, mFE=2.00e6				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	15	1.8e1	1.4e1	2.2e1	1.8e1	15	1.9e2	1.5e2	2.2e2	1.9e2
1	15	1.7e3	1.5e3	1.9e3	1.7e3	11	8.6e5	4.7e5	1.3e6	6.5e5
1e−1	14	4.7e4	1.1e4	8.7e4	4.6e4	0	*85e−2*	*26e−2*	*12e−1*	1.8e6
1e−3	12	1.9e5	1.1e5	2.7e5	1.8e5	.	.	.	.	.
1e−5	10	3.9e5	2.9e5	5.2e5	2.9e5	.	.	.	.	.
1e−8	0	*20e−7*	*43e−9*	*10e−3*	4.0e5	.	.	.	.	.

Δf	f_{18} **in 5-D**, N=15, mFE=500000					f_{18} **in 20-D**, N=15, mFE=2.00e6				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	15	3.9e2	2.7e2	5.1e2	3.9e2	15	4.3e4	1.9e4	7.1e4	4.3e4
1	15	8.1e3	6.9e3	9.4e3	8.1e3	1	2.8e7	1.3e7	>3e7	2.0e6
1e−1	10	2.7e5	1.5e5	4.5e5	1.7e5	0	*22e−1*	*10e−1*	*52e−1*	1.8e6
1e−3	5	1.1e6	7.0e5	2.1e6	3.4e5	.	.	.	.	.
1e−5	0	*10e−3*	*94e−6*	*35e−2*	4.0e5	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	f_{19} **in 5-D**, N=15, mFE=500000					f_{19} **in 20-D**, N=15, mFE=2.00e6				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	15	2.7e1	1.9e1	3.6e1	2.7e1	15	8.2e2	5.5e2	1.1e3	8.2e2
1	15	1.6e4	1.2e4	2.1e4	1.6e4	0	*31e−1*	*25e−1*	*35e−1*	1.0e6
1e−1	8	6.1e5	4.2e5	9.1e5	3.2e5	.	.	.	.	.
1e−3	0	*80e−3*	*27e−3*	*28e−2*	4.5e5	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	f_{20} **in 5-D**, N=15, mFE=500000					f_{20} **in 20-D**, N=15, mFE=2.00e6				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	15	1.3e2	1.1e2	1.6e2	1.3e2	15	7.1e3	5.3e3	8.9e3	7.1e3
1	15	7.3e3	6.1e3	8.6e3	7.3e3	13	5.2e5	2.6e5	8.6e5	3.6e5
1e−1	6	8.1e5	6.2e5	1.2e6	5.0e5	0	*53e−2*	*30e−2*	*11e−1*	8.9e5
1e−3	6	8.2e5	6.3e5	1.2e6	5.0e5	.	.	.	.	.
1e−5	6	8.2e5	6.3e5	1.2e6	5.0e5	.	.	.	.	.
1e−8	6	8.9e5	6.8e5	1.4e6	5.0e5	.	.	.	.	.

Δf	f_{21} **in 5-D**, N=15, mFE=500000					f_{21} **in 20-D**, N=15, mFE=2.00e6				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	15	1.4e2	9.1e1	2.0e2	1.4e2	13	3.2e5	1.4e4	6.2e5	1.6e5
1	8	4.4e5	2.5e5	7.5e5	1.9e5	3	8.1e6	4.8e6	2.4e7	2.0e6
1e−1	7	5.7e5	3.6e5	9.2e5	2.9e5	1	2.8e7	1.3e7	>3e7	2.0e6
1e−3	7	5.8e5	3.7e5	9.2e5	2.9e5	1	2.8e7	1.3e7	>3e7	2.0e6
1e−5	7	5.8e5	3.7e5	9.2e5	2.9e5	1	2.8e7	1.3e7	>3e7	2.0e6
1e−8	7	5.9e5	3.8e5	9.3e5	2.9e5	1	2.8e7	1.3e7	>3e7	2.0e6

Δf	f_{22} **in 5-D**, N=15, mFE=500000					f_{22} **in 20-D**, N=15, mFE=2.00e6				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	14	3.6e4	2.4e2	7.7e4	3.6e4	13	3.2e5	1.3e4	5.8e5	3.1e5
1	9	3.4e5	2.0e5	5.0e5	2.2e5	5	4.1e6	2.5e6	8.0e6	1.2e6
1e−1	6	7.7e5	4.6e5	1.4e6	2.5e5	1	2.8e7	1.3e7	>3e7	2.0e6
1e−3	6	8.2e5	5.1e5	1.5e6	2.6e5	1	2.8e7	1.3e7	>3e7	2.0e6
1e−5	6	8.5e5	5.2e5	1.6e6	2.6e5	1	2.9e7	1.4e7	>3e7	2.0e6
1e−8	6	8.9e5	5.6e5	1.6e6	2.8e5	1	2.9e7	1.4e7	>3e7	2.0e6

Δf	f_{23} **in 5-D**, N=15, mFE=500000					f_{23} **in 20-D**, N=15, mFE=2.00e6				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	15	6.3e0	4.6e0	8.1e0	6.3e0	15	9.5e0	6.1e0	1.3e1	9.5e0
1	15	3.0e4	2.5e4	3.5e4	3.0e4	2	1.4e7	7.2e6	>3e7	2.0e6
1e−1	2	3.5e6	1.8e6	>7e6	5.0e5	0	*12e−1*	*96e−2*	*14e−1*	2.8e5
1e−3	0	*30e−2*	*71e−3*	*38e−2*	3.5e5	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	f_{24} **in 5-D**, N=15, mFE=500000					f_{24} **in 20-D**, N=15, mFE=2.00e6				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	15	1.6e4	1.4e4	1.9e4	1.6e4	0	*66e+0*	*44e+0*	*77e+0*	1.8e6
1	1	7.1e6	3.3e6	>7e6	5.0e5	.	.	.	.	.
1e−1	0	*60e−1*	*11e−1*	*66e−1*	1.8e5	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Table 1: Shown are, for a given target difference to the optimal function value Δf: the number of successful trials (#); the expected running time to surpass $f_{\rm opt}+\Delta f$ (ERT, see Figure 2); the 10%-tile and 90%-tile of the bootstrap distribution of ERT; the average number of function evaluations in successful trials or, if none was successful, as last entry the median number of function evaluations to reach the best function value ($\mathbf{RT_{succ}}$). If $f_{\rm opt}+\Delta f$ was never reached, figures in *italics* denote the best achieved Δf-value of the median trial and the 10% and 90%-tile trial. Furthermore, N denotes the number of trials, and mFE denotes the maximum of number of function evaluations executed in one trial. See Figure 2 for the names of functions.

```
function PSO_Bounds(FUN, DIM, ftarget, maxfunevals)

% Set algorithm parameters
popsize = 40;
c1 = 2;
c2 = 2;
xbound = 5;
vbound = 5;
pmin = 0.2;
pmax = 0.8;
alpha = 0.05;
T = 0.0001;

% Allocate memory and initialize
xmin = -xbound * ones(1,DIM);
xmax = xbound * ones(1,DIM);
vmin = -vbound * ones(1,DIM);
vmax = vbound * ones(1,DIM);

x = 2 * xbound * rand(popsize,DIM) - xbound;
v = 2 * vbound * rand(popsize,DIM) - vbound;
pbest = x;

p = 0.5 * ones(1,DIM);

% update pbest and gbest
cost_p = feval(FUN, pbest');
[cost,index] = min(cost_p);
gbest = pbest(index,:);

maxfunevals = min(1e5 * DIM, maxfunevals);
maxiterations = ceil(maxfunevals/popsize);

for iter = 2 : maxiterations
    % Update inertia weight
    w = 0.9 - 0.8*(iter-2)/(maxiterations-2);

    % Update velocity
    v = w*v + c1*rand(popsize,DIM).*(pbest-x) + c2*rand(popsize,DIM).*
(repmat(gbest,popsize,1)-x);

    % Clamp veloctiy
    s = v < repmat(vmin,popsize,1);
    v = (1-s).*v + s.*repmat(vmin,popsize,1);
    b = v > repmat(vmax,popsize,1);
    v = (1-b).*v + b.*repmat(vmax,popsize,1);

    % Update position
    x = x + v;

    % Clamp position - Absorbing boundaries
    % Set x to the boundary
    s = x < repmat(xmin,popsize,1);
    x = (1-s).*x + s.*repmat(xmin,popsize,1);


    b = x > repmat(xmax,popsize,1);
    x = (1-b).*x + b.*repmat(xmax,popsize,1);

    % Clamp position - Absorbing boundaries
    % Set v to zero
    b = s | b;
    v = (1-b).*v + b.*zeros(popsize,DIM);

    % Update pbest and gbest if necessary
    cost_x = feval(FUN, x');
    s = cost_x<cost_p;
    cost_p = (1-s).*cost_p + s.*cost_x;
    s = repmat(s',1,DIM);
    pbest = (1-s).*pbest + s.*x;
    [cost,index] = min(cost_p);
    gbest = pbest(index,:);

    % Update dimension probability
    probability = sum(pbest>repmat(((xmin+xmax)/2),popsize,1));
    p = (1-alpha)*p + alpha*(probability/popsize);

    % Update bounds if necessary
    % Shift xmax
    pmn = p<pmin;
    xmax = (1-pmn).*xmax + pmn.*(xmax - (xmax-xmin)/2);

    % Shift xmin
    pmx = p>pmax;
    xmin = (1-pmx).*xmin + pmx.*(xmin + (xmax-xmin)/2);

    % In either case, set p to 0.5
    pm = pmn | pmx;
    p = (1-pm).*p + pm*0.5;

    % Re-initialize if necessary
    t = (xmax-xmin)<(2*T*xbound);
    xmin = (1-t).*xmin + t*-xbound;
    xmax = (1-t).*xmax + t*xbound;
    vmax = (1-t).*((xmax-xmin)/2) + t.*vbound;
    vmin = -vmax;
    t = repmat(t,popsize,1);
    v = (1-t).*v + t.*(2 * vbound * rand(popsize,DIM) - vbound);

    % Exit if target is reached
    if feval(FUN, 'fbest') < ftarget
      break;
    end
end
```

Figure 1: PSO_Bounds MATLAB-code.

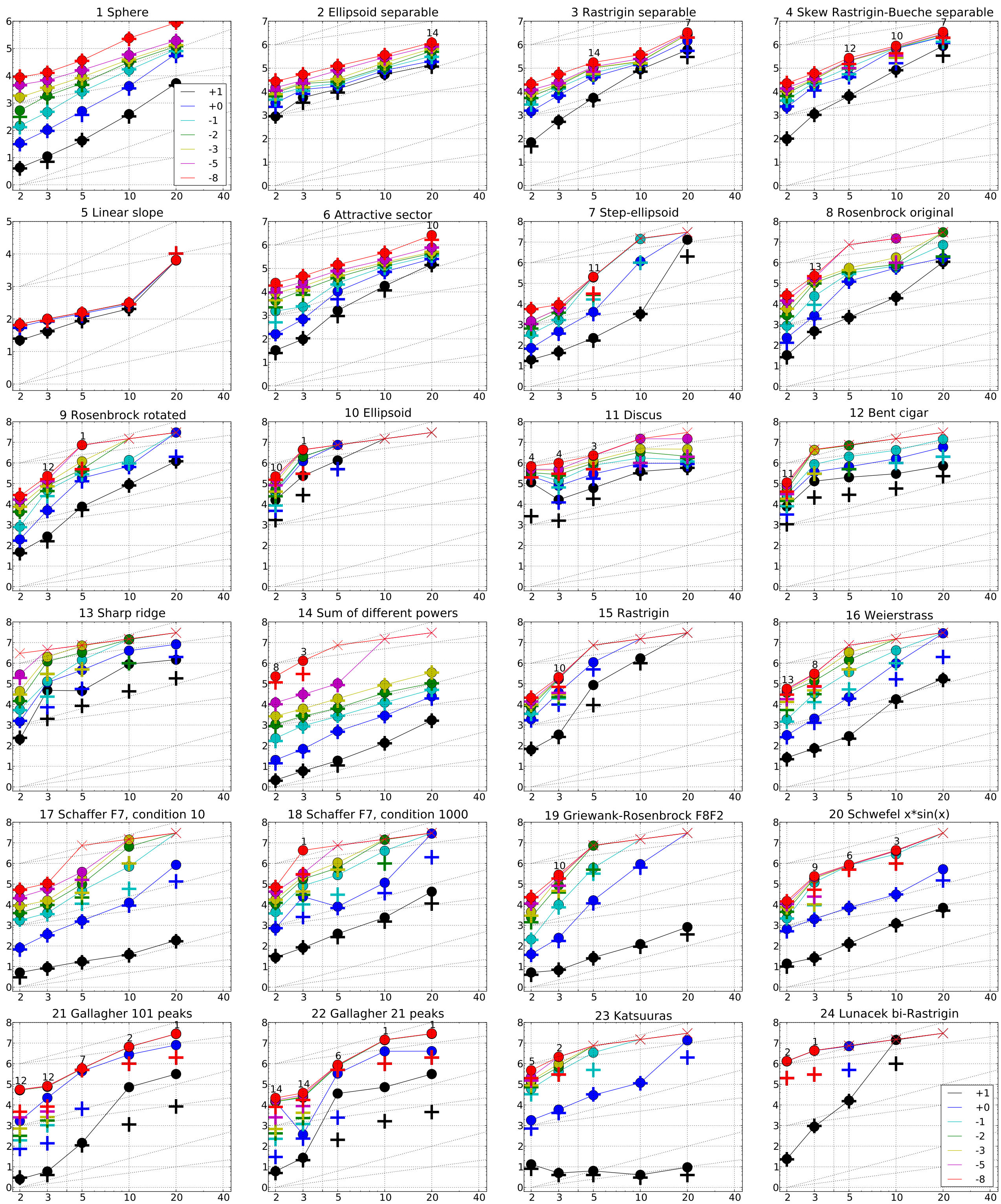

Figure 2: Expected Running Time (ERT, ●) to reach $f_{\rm opt} + \Delta f$ and median number of function evaluations of successful trials (+), shown for $\Delta f = 10, 1, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-5}, 10^{-8}$ (the exponent is given in the legend of f_1 and f_{24}) versus dimension in log-log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

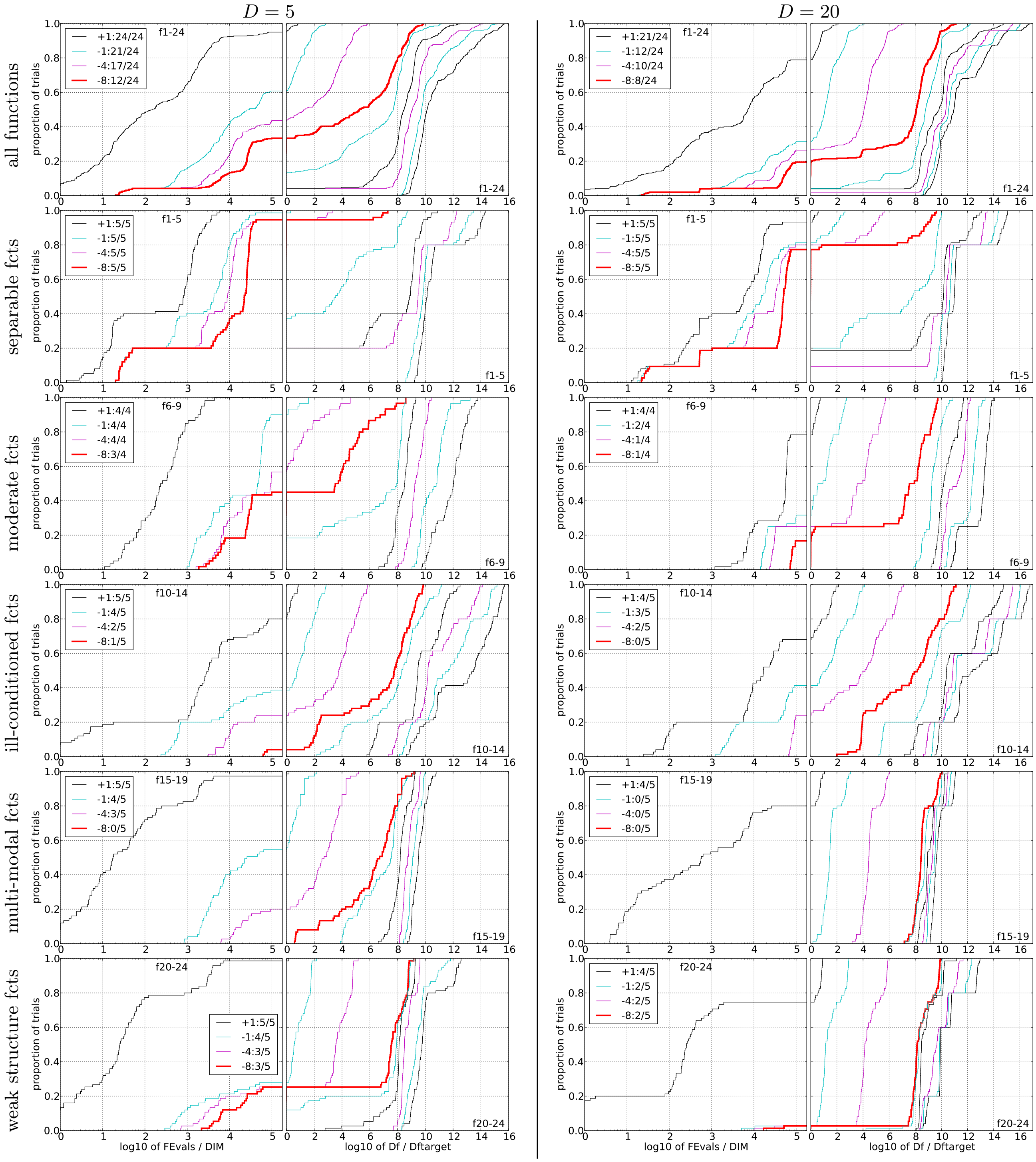

Figure 3: Empirical cumulative distribution functions (ECDFs), plotting the fraction of trials versus running time (left) or Δf. Left subplots: ECDF of the running time (number of function evaluations), divided by search space dimension D, to fall below $f_{\rm opt} + \Delta f$ with $\Delta f = 10^k$, where k is the first value in the legend. Right subplots: ECDF of the best achieved Δf divided by 10^k (upper left lines in continuation of the left subplot), and best achieved Δf divided by 10^{-8} for running times of $D, 10\,D, 100\,D\ldots$ function evaluations (from right to left cycling black-cyan-magenta). Top row: all results from all functions; second row: separable functions; third row: misc. moderate functions; fourth row: ill-conditioned functions; fifth row: multi-modal functions with adequate structure; last row: multi-modal functions with weak structure. The legends indicate the number of functions that were solved in at least one trial. FEvals denotes number of function evaluations, D and DIM denote search space dimension, and Δf and Df denote the difference to the optimal function value.