Particle Swarm Hybridized with Differential Evolution: Black-Box Optimization Benchmarking for Noisy Functions

José García-Nieto
Dept. de Lenguajes y Ciencias de la Computación
University of Málaga,
ETSI Informática,
Málaga - 29071, Spain
jnieto@lcc.uma.es

Enrique Alba
Dept. de Lenguajes y Ciencias de la Computación
University of Málaga,
ETSI Informática,
Málaga - 29071, Spain
eat@lcc.uma.es

Javier Apolloni
LIDIC - Departamento de Informática
University of San Luis,
Ejército de los Andes 950,
5700, Argentina
javierma@unsl.edu.ar

ABSTRACT

In this work we evaluate a Particle Swarm Optimizer hybridized with Differential Evolution and apply it to the Black-Box Optimization Benchmarking for noisy functions (BBOB 2009). We have performed the complete procedure established in this special session dealing with noisy functions with dimension: 2, 3, 5, 10, 20, and 40 variables. Our pro-posal obtained an accurate level of coverage rate, despite the simplicity of the model and the relatively small number of function evaluations used.

Keywords

Benchmarking, Black-box optimization, Noisy Functions, Hy-brid Algorithms, Particle Swarm, Differential Evolution

1. INTRODUCTION

Particle Swarm Optimization (PSO) [8] and Differential Evolution (DE) [9] have been successfully used on real parameter function optimization since they are two well adapted algorithms for continuous solution encoding. Real parameter optimization problems consist basically in this: *find* $\mathbf{x}^*$ *such that* $\forall\ \mathbf{x}\ f(\mathbf{x}^*) \leq f(\mathbf{x})$ (minimization). Here, $f(.)$ is a function in a real space domain that models an opti-mization problem, $\mathbf{x} = \{x_1, x_2, \ldots, x_{DIM}\}$ is a solution for such problem, and DIM is the number of variables with $x_i \in [x_i^{low}, x_i^{upp}]$ $(1 \leq i \leq DIM)$. Finally, x_i^{low}, $x_i^{upp} \in \mathbb{R}$ correspond to lower (low) and upper (upp) limits of the variable domain, respectively.

Swagatam Das *et al.* [3] proposed an initial hybridization of PSO and DE for continuous optimization. Based on this first idea, in this work we propose a PSO algorithm which basically uses the differential variation scheme employed in DE for adjusting the velocity of particles. In this way, by combining search strategies, parameter adaptation, and differential operators present in PSO and DE we expect to improve the performance of the resulting technique. Our proposal, called DEPSO, is evaluated by means of the Black-Box Optimization Benchmarking for noisy functions (BBOB 2009) [4, 7] according to the experimental design from [6]. We have performed the complete procedure established in this GECCO'09 workshop dealing with noisy functions with dimension: 2, 3, 5, 10, 20, and 40 variables. Our proposal will be shown to obtain an accurate level of coverage rate, even for the higher dimensions, and despite the relatively small number of function evaluations used ($1000 \times DIM$).

The remaining of the paper is organized as follows. In Section 2 the DEPSO algorithm is briefly described. Sections 3 and 4 presents the experimentation procedure and the results obtained, respectively. In Section 5, a brief analysis of the CPU Timing experiment is reported. Finally, conclusions and remarks are given in Section 6.

2. THE ALGORITHM: DEPSO

Our proposal, DEPSO, is basically running a PSO algorithm in which we incorporate ideas of DE. For each particle $\mathbf{p}$ (with velocity and position vectors $\mathbf{v}$ and $\mathbf{x}$, respectively) the velocity is updated according to two main influences: social and differential variation factors. The social factor concerns the local or global best position $\mathbf{g}$ of a given neighborhood of particles (being global when this neighborhood is the entire swarm). The differential variation factor is composed by using two randomly selected particle positions as made in DE. This way, for each particle $\mathbf{x}_i$ of the swarm, a differential vector $\mathbf{w} = \mathbf{x}_{r1} - \mathbf{x}_{r2}$ is generated where particles $\mathbf{x}_{r1}$ and $\mathbf{x}_{r2}$ are randomly (uniformly) selected (with $\mathbf{x}_i \neq \mathbf{x}_{r1} \neq \mathbf{x}_{r2}$). Then, the new velocity $\mathbf{v}'$ of a particle i is calculated by means of the following equation:

$$\mathbf{v}'_i \leftarrow \omega \cdot \mathbf{v}_i + \mu \cdot \mathbf{w} + \varphi \cdot (\mathbf{g}_i - \mathbf{x}_i), \quad (1)$$

where ω is the inertia weight of the particle that con-

trols the trade-off between global and local experience and μ is a scaling factor applied to the differential vector ($\mu = UN(0,1)$). The third component corresponds to the social factor which is influenced by the global best $\mathbf{g}$ of the swarm and directly proportional to the social coefficient φ (in this case $\varphi = UN(0,1)$). Therefore, in the velocity calculation, the standard historical influence used in PSO is replaced in our proposal by the differential vector $\mathbf{w}$.

Similarly to DE, the update of each j component of the velocity vector of a given particle i is carried out by means of the Equation 2 as follows:

$$v_i'(j) = \begin{cases} v_i'(j) & \text{if } r \leq Cr, \\ v_i(j) & \text{otherwise.} \end{cases} \quad (2)$$

Here, $r \in [0,1]$ is a uniformly distributed value which determines if the j^{th} component is selected from the new velocity or is selected from the current velocity, based on the crossover probability $Cr \in [0,1]$. This mechanism is used to increase the exploitation ability of the algorithm through the search space [9]. Finally, a particle i changes its position (moves) only if the new one $\mathbf{x}_i'$ is lower or equal than the current position $\mathbf{x}_i$ (minimizing in this work). In other case, the particle keeps its current position (equations 3 and 4).

$$\mathbf{x}_i'' = \begin{cases} \mathbf{x}_i' & \text{if } f(\mathbf{x'}_i) \leq f(\mathbf{x}_i) \\ \mathbf{x}_i & \text{otherwise,} \end{cases} \quad (3)$$

being

$$\mathbf{x}_i' \leftarrow \mathbf{x}_i + \mathbf{v'}_i \quad (4)$$

Additionally, with certain probability p_{mut}, a mutation operation is made on each particle in order to avoid an early convergence to a local optimum. The new mutated position $\mathbf{x}'$ is generated by means of the Equation 5.

$$\mathbf{x}'' \leftarrow \mathbf{x}^{low} + UN(0,1) \cdot (\mathbf{x}^{upp} - \mathbf{x}^{low}) \quad (5)$$

Vectors $\mathbf{x}^{low}$ and $\mathbf{x}^{upp}$ correspond to lower and upper limits of each dimension of the function to optimize, respectively.

Algorithm 1 shows pseudocode of the hybrid DEPSO algorithm developed for this work. First, an initialization process of all particles in the swarm S (as stated in [6]), and their initial evaluation (line 1) are carried out. After this, each evolution step the particle's positions are updated following the differential variation model of the equations previously explained (lines 4 to 18). In addition, the global best position reached at the moment is updated in order to guide the rest of the swarm. Eventually, a mutation operation is made (lines 20 to 24). Finally, the algorithm returns the best solution found during the whole process.

3. EXPERIMENTAL PROCEDURE

The experimentation procedure has been carried out according to [6] on the benchmark of 30 noisy functions given in [4, 7]. The DEPSO algorithm was implemented in C++ using the MALLBA library [1] of metaheuristics. The noisy functions were tackled connecting the C-code of the Black-Box Optimization Benchmarking to our implementation of DEPSO. Each candidate solution was sampled uniformly in $[-5,5]^{DIM}$, where DIM represents the search space dimension. The maximal number of function evaluation was set to $1000 \times DIM$.

Algorithm 1 Pseudocode of DEPSO

```
1: initialize(S)
2: while not stop condition is met do
3:   for each particle position x_i of the swarm S do
4:     /* Differential Variation */
5:     for each dimension j of the particle position x_i do
6:       w(j) ← x_r1(j) − x_r2(j)
7:       if r ≤ Cr then
8:         v'_i(j) ← ω · v_i(j) + μ · w(j) + φ · (g(j) − x_i(j))
9:       end if
10:    end for
11:    for each dimension j of the particle position x_i do
12:      x'_i(j) ← x_i(j) + v'_i(j)
13:    end for
14:    if f(x'_i) ≤ f(x_i) then
15:      x''_i ← x'_i
16:    else
17:      x''_i ← x_i
18:    end if
19:    /* Mutation */
20:    if UN(0,1) < p_mut then
21:      for each dimension j of the particle position x_i do
22:        x''_i(j) ← x^inf(j) + UN(0,1) · (x^sup(j) − x^inf(j))
23:      end for
24:    end if
25:  end for
26: end while
27: Output: Best solution found
```

Our proposal was tested performing 15 independent runs for each noisy function and each dimension. Table 1 shows the parameter setting used to configure DEPSO. These parameters were tuned in the context of the special session of CEC'05 for real parameter optimization [11, 5] reaching results statistically similar to the best participant algorithms (G-CMA-ES [2] and K-PCX [10]) in that session. This parameterization was kept the same for all the experiments, and therefore the *crafting effort* [6] is zero.

Table 1: Parameter setting used in DEPSO

Description	Parameter	Value
Swarm size	Ss	20
Crossover probability	Cr	0.9
Inertia weight	ω	0.1
Differential mutation	μ	$UN(0,1)$
Social coefficient	φ	$1 \cdot UN(0,1)$
Mutation probability	p_{mut}	$\frac{1}{DIM}$

4. RESULTS AND DISCUSSION

In this section, the results are presented and some discussions are made in terms of the number of successful trials ($f_{\text{opt}} + \Delta f \leq 10^{-8}$) reached for each kind of noisy function: moderate (f101-f106), severe (f107-f121), and severe multimodal (f122-f139).

Figure 1 shows the Expected Running Time (ERT, ●) to reach $f_{\text{opt}} + \Delta f$ and median number of function evaluations of successful trials (+). As we can observe, our proposal obtains the highest number of successful trials in moderate noise functions, specifically in dimensions 2, 3, 5 and 10, in f101 and f102. With regards to severe noise functions, the target function is reached in 9 out of 15 trials for the lower dimensions. In severe noise multimodal functions, DEPSO

obtains successful trials in f125 and f130 for dimension 2, and in f128 for dimensions 2, 3 and 5. As a summary, in Table 2 the functions are ranked by means of the number of successful trials that DEPSO has obtained with them (with the best ranked functions in the top).

For higher dimensions (20 and 40), the best behavior of our proposal can be observed when facing moderate noise functions as shown in Table 3, concretely in functions f101 and f102 where error precisions slightly higher than 10^{-5} were reached. Secondly, for severe noise and severe noise multimodal functions (tables 3 and 4 DIM=20), the best achieved Δf precisions (of the median trials) are always close to $e+0$, although being some of them close to $e-2$ as in f109, f125, and f127. We suspect that the relative low convergence rate in severe noise functions (for the higher dimensions) can be due to the small number of maximal function evaluations employed ($1000 \times DIM$).

Concerning this last issue, the empirical cumulative distribution function (ECDF) of the running time (in terms of function evaluations) is shown in Figure 2 left. We can easily observe that the ECDF tends to get reduced with the number of functions evaluation for all noisy functions. Specifically for severe noise and severe noise multimodal functions, the ECDF of the best achieved Δf (Figure 2 right) tends to reduce the difference to the optimal function value noticeably (with the proportion of trials).

Table 2: Noisy Functions ranked by the number of successful trials obtained by DEPSO. Columns indicate dimensions

2	3	5	10	20	40
f101	f101	f101	f101	-	-
f102	f102	f102	f102	-	-
f113	f107	f107	-	-	-
f128	f113	f113	-	-	-
f107	f128	f128	-	-	-
f103	f103	-	-	-	-
f115	f115	-	-	-	-
f125	-	-	-	-	-
f130	-	-	-	-	-
f116	-	-	-	-	-
f105	-	-	-	-	-

5. CPU TIMING EXPERIMENT

For the timing experiment, the same DEPSO algorithm was run on f8 until at least 30 seconds had passed (according to the `exampletiming` procedure available in BBOB 2009 [6]). These experiments have been conducted with an Intel(R) Corel(TM)2 CPU processor with 1.66 GHz and 1GB RAM; O.S Linux Ubuntu version 8.10 using the C-code provided. The results were 1.1, 0.9, 1.3, 2.3, 3.9, and 7.1 × e-06 seconds per function evaluation in dimensions 2, 3, 5, 10, 20, and 40, respectively.

6. CONCLUSION

In this work we have experimentally studied DEPSO, a simple and easy to implement optimization algorithm constructed by hybridizing the Particle Swarm Optimizer with Differential Evolution operations. The experiments have been made in the context of the special session of real parameter Black-Box Optimization Benchmarking (GECCO BBOB 2009), performing the complete procedure previously established, and dealing with noisy functions with dimension: 2, 3, 5, 10, 20, and 40 variables. Our proposal obtained an accurate level of coverage rate for dimensions 2, 3, 5, and 10, specifically with moderate noise and severe noise multimodal functions. Successful trials were found for 11 out of 30 functions. The fact of using the same parameter setting for all functions (and dimensions), together with the relatively small number of function evaluations used ($1000 \times DIM$), leads us to think that DEPSO can be easily improved for better covering noise functions with higher dimensions. Future experiments with different parameter settings depending of each family of noisy functions, and performing larger number of evaluations can be made in this sense.

7. ACKNOWLEDGMENTS

Authors acknowledge funds from the Spanish Ministry of Sciences and Innovation European FEDER under contract TIN2008-06491-C04-01 (M* project, available in URL `http://mstar.lcc.uma.es`) and CICE, Junta de Andalucía under contract P07-TIC-03044 (DIRICOM project, available in URL `http://diricom.lcc.uma.es`).

8. REFERENCES

[1] E. Alba, G. Luque, J. García-Nieto, G. Ordonez, and G. Leguizamón. Mallba: a software library to design efficient optimisation algorithms. *Int. J. of Innovative Computing and Applications 2007 (IJICA)*, 1(1):74–85, 2007.

[2] A. Auger and N. Hansen. A restart cma evolution strategy with increasing population size. *IEEE Congress on Evolutionary Computation*, 2:1769–1776, 2005.

[3] S. Das, A. Abraham, and A. Konar. Particle swarm optimization and differential evolution algorithms: Technical analysis, applications and hybridization perspectives. *In Advances of Computational Intelligence in Industrial Systems*, 2008.

[4] S. Finck, N. Hansen, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2009: Presentation of the noisy functions. Technical Report 2009/21, Research Center PPE, 2009.

[5] J. Garcia-Nieto, E. J. Apolloni, Alba, and G. Leguizamón. Algoritmo Basado en Cúmulos de Partículas y Evolución Diferencial para la Resolución de Problemas de Optimización Continua. In *VI Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB'09)*, page 433.440, Málaga, 11 a 13 de Febrero, 2009.

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

[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.

[8] J. Kennedy and R. Eberhart. Particle swarm optimization. *Neural Networks, Piscataway, NJ.,*

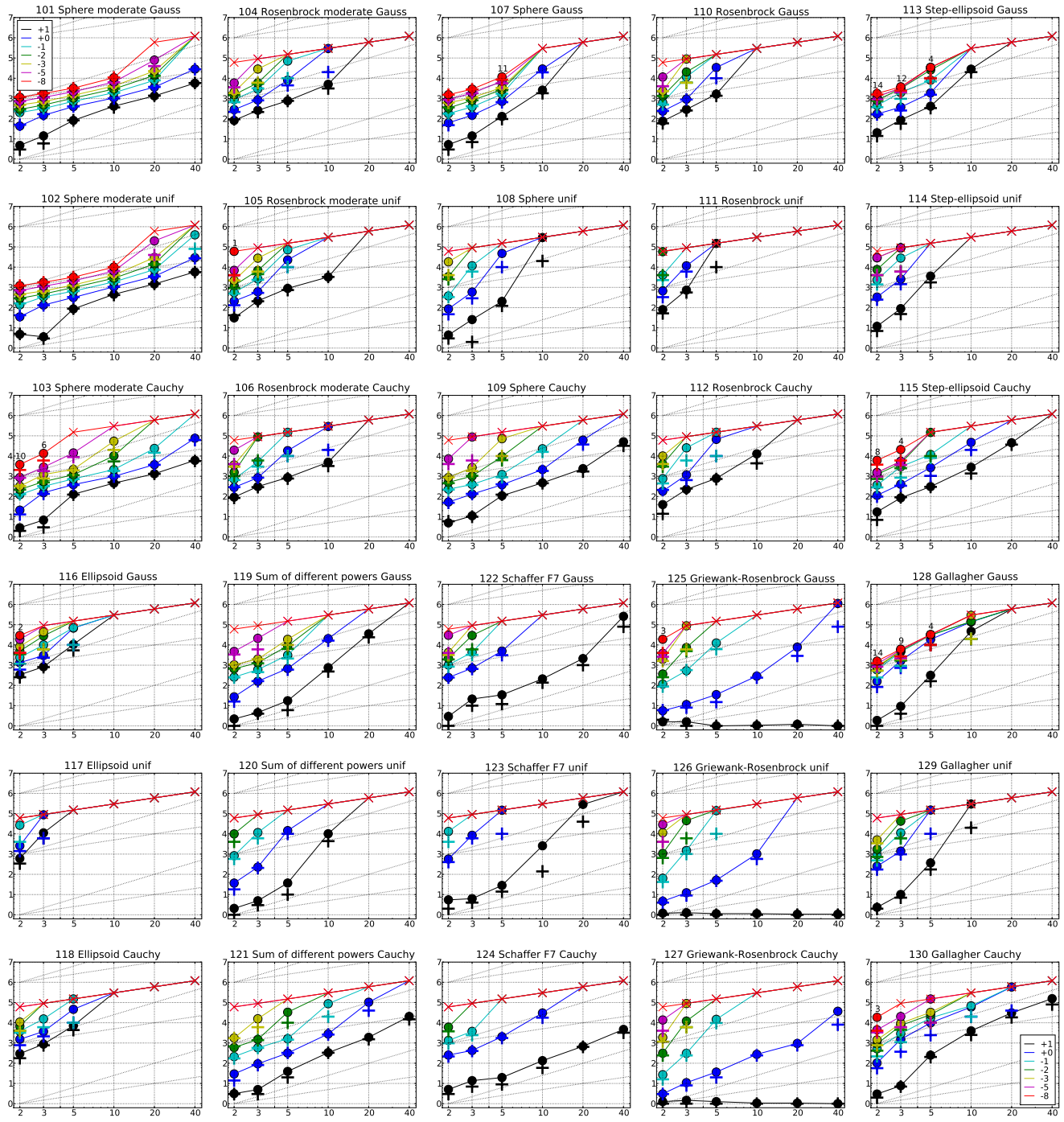

Figure 1: Expected Running Time (ERT, ●) **to reach** $f_{\mathrm{opt}} + \Delta f$ **and median number of function evaluations of successful trials** (+), **shown for** $\Delta f = 10, 1, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-5}, 10^{-8}$ **(the exponent is given in the legend of** f_{101} **and** f_{130}**) 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 (×) 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.**

Δf	#	ERT	10%	90%	$\mathrm{RT_{succ}}$	#	ERT	10%	90%	$\mathrm{RT_{succ}}$
	f_{101} **in 5-D**, N=15, mFE=3600					f_{101} **in 20-D**, N=15, mFE=40040				
10	15	8.2e1	6.6e1	9.9e1	8.2e1	15	1.3e3	1.2e3	1.4e3	1.3e3
1	15	4.0e2	3.7e2	4.2e2	4.0e2	15	3.6e3	3.4e3	3.8e3	3.6e3
1e−1	15	7.3e2	7.0e2	7.5e2	7.3e2	15	7.9e3	7.6e3	8.2e3	7.9e3
1e−3	15	1.4e3	1.4e3	1.5e3	1.4e3	15	2.3e4	2.1e4	2.4e4	2.3e4
1e−5	15	2.1e3	2.1e3	2.2e3	2.1e3	7	7.9e4	5.9e4	1.2e5	3.7e4
1e−8	15	3.2e3	3.2e3	3.3e3	3.2e3	0	*27e−6*	*29e−8*	*12e−5*	4.0e4
	f_{103} **in 5-D**, N=15, mFE=10040					f_{103} **in 20-D**, N=15, mFE=40040				
10	15	1.3e2	1.0e2	1.6e2	1.3e2	15	1.3e3	1.3e3	1.4e3	1.3e3
1	15	3.9e2	3.5e2	4.3e2	3.9e2	15	3.8e3	3.6e3	3.9e3	3.8e3
1e−1	15	7.4e2	7.0e2	7.9e2	7.4e2	12	2.4e4	1.8e4	3.1e4	2.0e4
1e−3	15	2.2e3	1.9e3	2.5e3	2.2e3	0	*34e−3*	*16e−3*	*12e−2*	3.2e4
1e−5	8	1.4e4	1.1e4	2.0e4	8.4e3	.	.	.	.	.
1e−8	0	*33e−7*	*79e−8*	*16e−5*	7.9e3	.	.	.	.	.
	f_{105} **in 5-D**, N=15, mFE=10040					f_{105} **in 20-D**, N=15, mFE=40040				
10	15	8.9e2	8.0e2	9.8e2	8.9e2	0	*20e+0*	*15e+0*	*84e+0*	3.5e4
1	5	2.3e4	1.6e4	4.1e4	8.8e3	.	.	.	.	.
1e−1	2	7.2e4	3.7e4	>1e5	1.0e4	.	.	.	.	.
1e−3	0	*17e−1*	*87e−3*	*30e−1*	7.9e3	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.
	f_{107} **in 5-D**, N=15, mFE=10040					f_{107} **in 20-D**, N=15, mFE=40040				
10	15	1.3e2	9.4e1	1.6e2	1.3e2	0	*38e+0*	*18e+0*	*71e+0*	1.4e4
1	15	7.2e2	6.2e2	8.1e2	7.2e2	.	.	.	.	.
1e−1	15	1.6e3	1.5e3	1.8e3	1.6e3	.	.	.	.	.
1e−3	15	3.5e3	3.2e3	3.8e3	3.5e3	.	.	.	.	.
1e−5	13	7.1e3	5.8e3	9.0e3	5.6e3	.	.	.	.	.
1e−8	11	1.2e4	9.6e3	1.5e4	8.3e3	.	.	.	.	.
	f_{109} **in 5-D**, N=15, mFE=10040					f_{109} **in 20-D**, N=15, mFE=40040				
10	15	1.1e2	8.9e1	1.3e2	1.1e2	15	2.4e3	2.0e3	2.8e3	2.4e3
1	15	3.8e2	3.4e2	4.2e2	3.8e2	8	6.1e4	4.6e4	9.0e4	3.3e4
1e−1	15	1.2e3	1.0e3	1.5e3	1.2e3	0	*99e−2*	*67e−2*	*28e−1*	3.2e4
1e−3	2	7.2e4	3.6e4	>1e5	9.0e3	.	.	.	.	.
1e−5	0	*51e−4*	*97e−5*	*44e−3*	7.1e3	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.
	f_{111} **in 5-D**, N=15, mFE=10040					f_{111} **in 20-D**, N=15, mFE=40040				
10	1	1.5e5	7.4e4	>1e5	1.0e4	0	*77e+3*	*42e+3*	*14e+4*	3.2e3
1	0	*31e+0*	*12e+0*	*94e+0*	5.0e3	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.
	f_{113} **in 5-D**, N=15, mFE=10040					f_{113} **in 20-D**, N=15, mFE=40040				
10	15	4.1e2	3.3e2	5.0e2	4.1e2	0	*14e+1*	*91e+0*	*29e+1*	1.6e4
1	15	1.9e3	1.5e3	2.2e3	1.9e3	.	.	.	.	.
1e−1	12	8.1e3	6.5e3	1.0e4	6.4e3	.	.	.	.	.
1e−3	4	3.4e4	2.1e4	7.0e4	8.8e3	.	.	.	.	.
1e−5	4	3.4e4	2.2e4	7.0e4	8.8e3	.	.	.	.	.
1e−8	4	3.4e4	2.2e4	7.0e4	9.0e3	.	.	.	.	.
	f_{115} **in 5-D**, N=15, mFE=10040					f_{115} **in 20-D**, N=15, mFE=40040				
10	15	3.1e2	2.7e2	3.6e2	3.1e2	9	4.5e4	3.6e4	6.1e4	3.1e4
1	13	2.7e3	1.6e3	4.0e3	2.5e3	0	*93e−1*	*64e−1*	*24e+0*	3.2e4
1e−1	8	1.2e4	8.6e3	1.7e4	7.0e3	.	.	.	.	.
1e−3	0	*58e−3*	*11e−3*	*10e−1*	6.3e3	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.
	f_{117} **in 5-D**, N=15, mFE=10040					f_{117} **in 20-D**, N=15, mFE=40040				
10	0	*15e+1*	*61e+0*	*29e+1*	3.2e3	0	*26e+3*	*18e+3*	*43e+3*	7.1e3
1	.	.	.	.	.	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.
	f_{119} **in 5-D**, N=15, mFE=10040					f_{119} **in 20-D**, N=15, mFE=40040				
10	15	1.8e1	1.0e1	2.6e1	1.8e1	10	3.5e4	2.6e4	4.9e4	2.4e4
1	15	6.8e2	5.8e2	8.0e2	6.8e2	0	*83e−1*	*56e−1*	*15e+0*	1.8e4
1e−1	14	3.1e3	2.4e3	4.0e3	2.9e3	.	.	.	.	.
1e−3	7	1.9e4	1.4e4	2.8e4	9.7e3	.	.	.	.	.
1e−5	0	*19e−4*	*17e−5*	*29e−3*	7.9e3	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	#	ERT	10%	90%	$\mathrm{RT_{succ}}$	#	ERT	10%	90%	$\mathrm{RT_{succ}}$
	f_{102} **in 5-D**, N=15, mFE=3480					f_{102} **in 20-D**, N=15, mFE=40040				
10	15	8.6e1	6.5e1	1.1e2	8.6e1	15	1.5e3	1.4e3	1.5e3	1.5e3
1	15	3.2e2	2.7e2	3.6e2	3.2e2	15	3.5e3	3.4e3	3.6e3	3.5e3
1e−1	15	6.5e2	6.0e2	7.0e2	6.5e2	15	7.5e3	7.3e3	7.8e3	7.5e3
1e−3	15	1.3e3	1.2e3	1.4e3	1.3e3	13	3.0e4	2.5e4	3.7e4	2.5e4
1e−5	15	2.1e3	2.0e3	2.1e3	2.1e3	3	2.0e5	1.2e5	6.0e5	4.0e4
1e−8	15	3.2e3	3.1e3	3.3e3	3.2e3	0	*46e−6*	*61e−7*	*14e−4*	4.0e4
	f_{104} **in 5-D**, N=15, mFE=10040					f_{104} **in 20-D**, N=15, mFE=40040				
10	15	7.7e2	7.0e2	8.4e2	7.7e2	0	*19e+0*	*18e+0*	*82e+0*	3.5e4
1	10	8.1e3	6.5e3	1.0e4	6.8e3	.	.	.	.	.
1e−1	2	7.1e4	3.7e4	>1e5	1.0e4	.	.	.	.	.
1e−3	0	*53e−2*	*60e−3*	*19e−1*	8.9e3	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.
	f_{106} **in 5-D**, N=15, mFE=10040					f_{106} **in 20-D**, N=15, mFE=40040				
10	15	8.7e2	7.9e2	9.5e2	8.7e2	0	*25e+0*	*18e+0*	*83e+0*	3.5e4
1	6	1.8e4	1.3e4	3.1e4	7.5e3	.	.	.	.	.
1e−1	1	1.5e5	7.3e4	>1e5	1.0e4	.	.	.	.	.
1e−3	0	*14e−1*	*18e−2*	*31e−1*	8.9e3	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.
	f_{108} **in 5-D**, N=15, mFE=10040					f_{108} **in 20-D**, N=15, mFE=40040				
10	15	2.0e2	9.4e1	3.2e2	2.0e2	0	*81e+0*	*59e+0*	*14e+1*	1.1e4
1	3	4.8e4	2.8e4	1.5e5	8.3e3	.	.	.	.	.
1e−1	0	*19e−1*	*74e−2*	*47e−1*	2.8e3	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.
	f_{110} **in 5-D**, N=15, mFE=10040					f_{110} **in 20-D**, N=15, mFE=40040				
10	15	1.6e3	1.4e3	1.9e3	1.6e3	0	*72e+2*	*14e+2*	*22e+3*	2.5e4
1	4	3.4e4	2.2e4	7.0e4	9.1e3	.	.	.	.	.
1e−1	0	*20e−1*	*38e−2*	*31e−1*	7.1e3	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.
	f_{112} **in 5-D**, N=15, mFE=10040					f_{112} **in 20-D**, N=15, mFE=40040				
10	15	7.9e2	7.4e2	8.4e2	7.9e2	0	*37e+0*	*24e+0*	*13e+1*	3.2e4
1	2	6.7e4	3.6e4	>1e5	1.0e4	.	.	.	.	.
1e−1	1	1.5e5	7.4e4	>1e5	1.0e4	.	.	.	.	.
1e−3	0	*25e−1*	*31e−2*	*30e−1*	7.1e3	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.
	f_{114} **in 5-D**, N=15, mFE=10040					f_{114} **in 20-D**, N=15, mFE=40040				
10	13	3.6e3	2.1e3	5.4e3	2.7e3	0	*49e+1*	*44e+1*	*83e+1*	2.5e4
1	0	*42e−1*	*15e−1*	*15e+0*	5.0e3	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.
	f_{116} **in 5-D**, N=15, mFE=10040					f_{116} **in 20-D**, N=15, mFE=40040				
10	9	1.1e4	7.8e3	1.5e4	6.5e3	0	*16e+3*	*94e+2*	*38e+3*	1.4e4
1	2	6.9e4	3.6e4	>1e5	1.0e4	.	.	.	.	.
1e−1	2	7.1e4	3.6e4	>1e5	1.0e4	.	.	.	.	.
1e−3	0	*73e−1*	*45e−3*	*26e+0*	7.9e3	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.
	f_{118} **in 5-D**, N=15, mFE=10040					f_{118} **in 20-D**, N=15, mFE=40040				
10	11	7.1e3	5.3e3	9.5e3	5.2e3	0	*55e+1*	*38e+1*	*73e+1*	3.5e4
1	3	4.6e4	2.6e4	1.4e5	7.7e3	.	.	.	.	.
1e−1	1	1.5e5	7.5e4	>2e5	1.0e4	.	.	.	.	.
1e−3	0	*37e−1*	*21e−2*	*19e+0*	7.9e3	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.
	f_{120} **in 5-D**, N=15, mFE=10040					f_{120} **in 20-D**, N=15, mFE=40040				
10	15	3.7e1	1.6e1	6.0e1	3.7e1	0	*31e+0*	*21e+0*	*42e+0*	3.5e3
1	7	1.4e4	9.6e3	2.4e4	5.9e3	.	.	.	.	.
1e−1	0	*10e−1*	*60e−2*	*23e−1*	2.8e3	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Table 3: Shown are, for functions f_{101}-f_{120} and for a given target difference to the optimal function value Δf: the number of successful trials (#); the expected running time to surpass $f_{\mathrm{opt}} + \Delta f$ (ERT, see Figure 1); 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 ($\mathrm{RT_{succ}}$). If $f_{\mathrm{opt}} + \Delta f$ 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 1 for the names of functions.

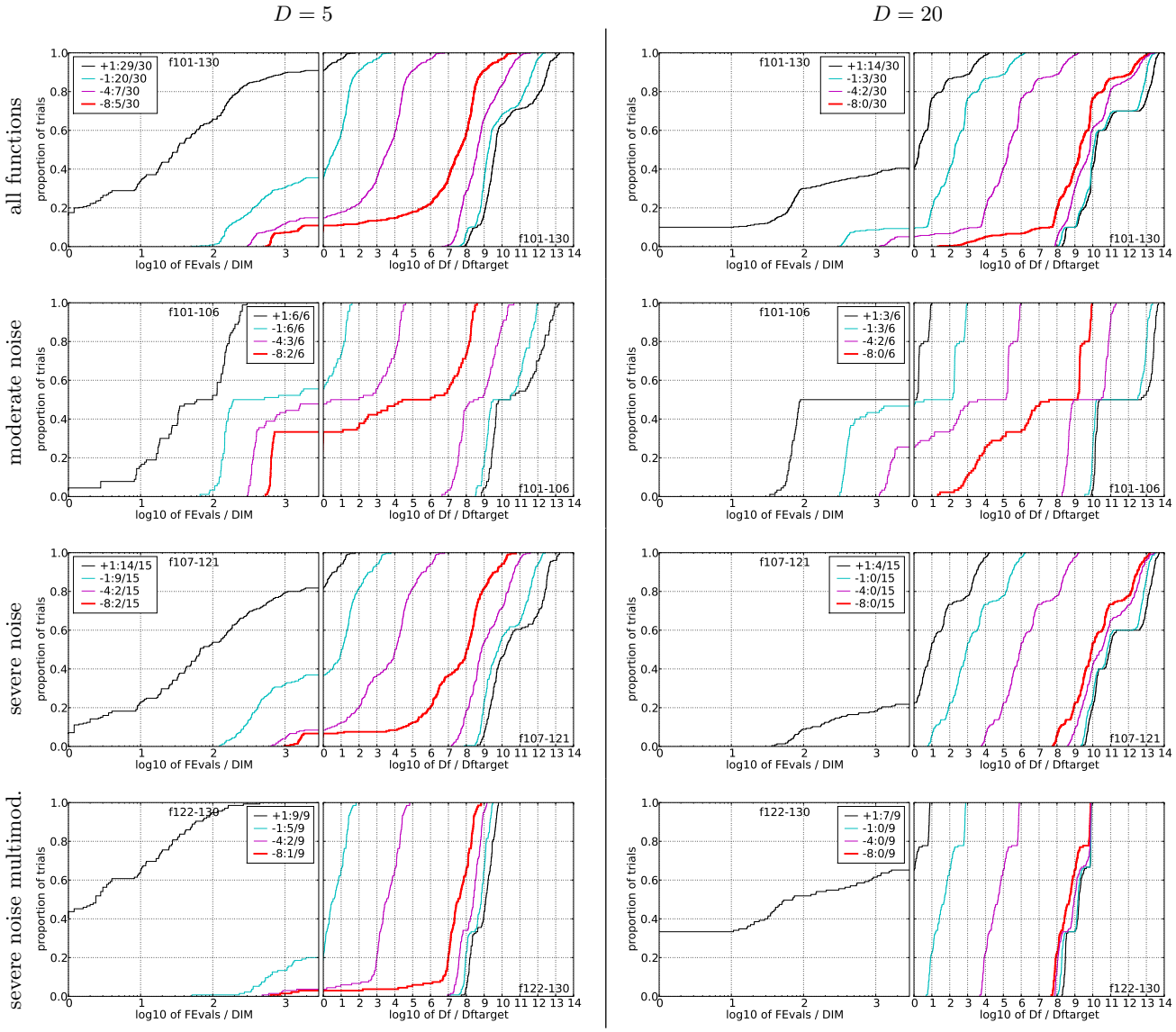

Figure 2: Empirical cumulative distribution functions (ECDFs), plotting the fraction of trials versus running time (left) or Δf. Left subplots: ECDF of the running time (number of function evaluations), divided by search space dimension D, to fall below $f_{\mathrm{opt}} + \Delta f$ with $\Delta f = 10^k$, where k is the first value in the legend. Right subplots: ECDF of the best achieved Δf divided by 10^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: moderate noise functions; third row: severe noise functions; fourth row: severe noise and highly-multimodal functions. 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.

Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
	f_{121} **in 5-D**, N=15, mFE=10040					f_{121} **in 20-D**, N=15, mFE=40040				
10	15	3.9e1	2.8e1	5.1e1	3.9e1	15	1.9e3	1.5e3	2.3e3	1.9e3
1	15	3.2e2	2.7e2	3.7e2	3.2e2	5	1.0e5	6.8e4	1.9e5	3.3e4
1e−1	15	1.6e3	1.4e3	1.8e3	1.6e3	0	*11e−1*	*71e−2*	*28e−1*	2.8e4
1e−3	0	*14e−3*	*68e−4*	*32e−3*	5.6e3	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.
	f_{123} **in 5-D**, N=15, mFE=10040					f_{123} **in 20-D**, N=15, mFE=40040				
10	15	2.8e1	1.7e1	4.0e1	2.8e1	2	2.9e5	1.5e5	>6e5	4.0e4
1	1	1.5e5	7.2e4	>1e5	1.0e4	0	*12e+0*	*95e−1*	*17e+0*	2.0e4
1e−1	0	*26e−1*	*13e−1*	*39e−1*	2.8e3	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.
	f_{125} **in 5-D**, N=15, mFE=10040					f_{125} **in 20-D**, N=15, mFE=40040				
10	15	1.0e0	1.0e0	1.0e0	1.0e0	15	1.2e0	1.1e0	1.3e0	1.2e0
1	15	3.5e1	2.2e1	5.0e1	3.5e1	14	7.8e3	4.5e3	1.2e4	7.3e3
1e−1	8	1.2e4	8.7e3	1.9e4	6.1e3	0	*81e−2*	*70e−2*	*96e−2*	1.8e4
1e−3	0	*98e−3*	*74e−3*	*17e−2*	2.5e3	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.
	f_{127} **in 5-D**, N=15, mFE=10040					f_{127} **in 20-D**, N=15, mFE=40040				
10	15	1.3e0	1.1e0	1.4e0	1.3e0	15	1.1e0	1.0e0	1.1e0	1.1e0
1	15	3.6e1	2.6e1	4.8e1	3.6e1	15	9.5e2	7.9e2	1.1e3	9.5e2
1e−1	8	1.5e4	1.2e4	2.1e4	9.1e3	0	*65e−2*	*58e−2*	*75e−2*	2.2e4
1e−3	0	*96e−3*	*35e−3*	*14e−2*	6.3e3	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.
	f_{129} **in 5-D**, N=15, mFE=10040					f_{129} **in 20-D**, N=15, mFE=40040				
10	15	3.7e2	2.0e2	5.7e2	3.7e2	0	*72e+0*	*69e+0*	*77e+0*	1.3e4
1	1	1.5e5	7.5e4	>2e5	1.0e4	.	.	.	.	.
1e−1	0	*24e−1*	*14e−1*	*67e−1*	4.0e3	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
	f_{122} **in 5-D**, N=15, mFE=10040					f_{122} **in 20-D**, N=15, mFE=40040				
10	15	3.4e1	2.2e1	4.6e1	3.4e1	15	2.1e3	1.4e3	2.8e3	2.1e3
1	12	5.0e3	3.6e3	6.6e3	4.2e3	0	*63e−1*	*49e−1*	*77e−1*	1.0e4
1e−1	0	*57e−2*	*35e−2*	*12e−1*	5.6e3	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.
	f_{124} **in 5-D**, N=15, mFE=10040					f_{124} **in 20-D**, N=15, mFE=40040				
10	15	2.0e1	1.2e1	2.9e1	2.0e1	15	6.9e2	6.0e2	7.9e2	6.9e2
1	15	2.0e3	1.6e3	2.5e3	2.0e3	0	*28e−1*	*22e−1*	*45e−1*	2.8e4
1e−1	0	*27e−2*	*14e−2*	*60e−2*	7.1e3	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.
	f_{126} **in 5-D**, N=15, mFE=10040					f_{126} **in 20-D**, N=15, mFE=40040				
10	15	1.1e0	1.0e0	1.3e0	1.1e0	15	1.1e0	1.0e0	1.1e0	1.1e0
1	15	5.1e1	3.6e1	6.8e1	5.1e1	0	*13e−1*	*11e−1*	*16e−1*	7.9e3
1e−1	1	1.4e5	6.6e4	>1e5	1.0e4	.	.	.	.	.
1e−3	0	*21e−2*	*10e−2*	*27e−2*	2.8e3	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.
	f_{128} **in 5-D**, N=15, mFE=10040					f_{128} **in 20-D**, N=15, mFE=40040				
10	15	3.2e2	2.1e2	4.3e2	3.2e2	0	*66e+0*	*62e+0*	*71e+0*	2.2e4
1	6	1.9e4	1.3e4	3.0e4	8.6e3	.	.	.	.	.
1e−1	4	3.0e4	2.0e4	6.1e4	1.0e4	.	.	.	.	.
1e−3	4	3.1e4	2.0e4	6.2e4	1.0e4	.	.	.	.	.
1e−5	4	3.2e4	2.1e4	6.5e4	1.0e4	.	.	.	.	.
1e−8	4	3.3e4	2.1e4	6.7e4	1.0e4	.	.	.	.	.
	f_{130} **in 5-D**, N=15, mFE=10040					f_{130} **in 20-D**, N=15, mFE=40040				
10	15	2.5e2	1.9e2	3.1e2	2.5e2	11	3.1e4	2.4e4	4.0e4	2.3e4
1	8	9.9e3	6.6e3	1.5e4	5.7e3	1	5.8e5	2.8e5	>6e5	4.0e4
1e−1	6	1.7e4	1.2e4	3.0e4	7.5e3	0	*62e−1*	*22e−1*	*19e+0*	3.2e4
1e−3	4	3.3e4	2.0e4	6.9e4	8.3e3	.	.	.	.	.
1e−5	1	1.5e5	7.4e4	>1e5	1.0e4	.	.	.	.	.
1e−8	0	*94e−2*	*16e−5*	*20e−1*	7.1e3	.	.	.	.	.

Table 4: Shown are, for functions f_{121}-f_{130} and 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 1); 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 (RT_{succ}). 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 1 for the names of functions.

Proceedings of IEEE International Conference on, pages 1942–1948, 1995.

[9] K. V. Price, R. Storn, and J. Lampinen. *Differential Evolution: A practical Approach to Global Optimization*. Springer-Verlag, London, UK, 2005.

[10] A. Sinha, S. Tiwari, and K. Deb. A population-based, steady-state procedure for real-parameter optimization. *IEEE Congress on Evolutionary Computation*, 1:514–521, 2005.

[11] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari. Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. Technical report, Nanyang Technological University 2005.