



Benchmarking the NEWUOA on the BBOB-2009 Noisy Testbed

Raymond Ros

► **To cite this version:**

Raymond Ros. Benchmarking the NEWUOA on the BBOB-2009 Noisy Testbed. GECCO, Jul 2009, Montréal, Canada. 2009. <inria-00377083>

HAL Id: inria-00377083

<https://hal.inria.fr/inria-00377083>

Submitted on 20 Apr 2009

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Benchmarking the NEWUOA on the BBOB-2009 Noisy Testbed

Raymond Ros
Univ. Paris-Sud, LRI
UMR 8623 / INRIA Saclay, projet TAO
F-91405 Orsay, France
raymond.ros@lri.fr

ABSTRACT

The NEWUOA which belongs to the class of Derivative-Free optimization algorithms is benchmarked on the BBOB-2009 noisy testbed. A multistart strategy is applied with a maximum number of function evaluations of 10^4 times the search space dimension.

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

Benchmarking, Black-box optimization, Derivative-free optimization

1. ALGORITHM PRESENTATION

The NEWUOA (New Unconstrained Optimization Algorithm) [4] is a Derivative-Free Optimization (DFO) algorithm using the trust region paradigm. NEWUOA computes a quadratic interpolation of the objective function in the current trust region and performs a truncated conjugate gradient minimization of the surrogate model in the trust region. It then updates either the current best point or the radius of the trust region, based on the a posteriori interpolation error. The time complexity of the algorithm is $\mathcal{O}(m^2n)$ in the worst case but in practice closer to $\mathcal{O}(mn)$, where m is the number of interpolation points used for the determination of the quadratic model and n is the dimension of the search space. The number of interpolation points m is a parameter of the algorithm and needs to be chosen in the range $[n+2, \frac{(n+1)(n+2)}{2}]$. Other parameters of the algorithm

are the initial and final radii of the trust region, respectively governing the initial 'granularity' and the precision of the search. A simple stochastic independent restart procedure (as advised in [2]) was added to improve the probability of the algorithm reaching a given target function value.

2. EXPERIMENTAL PROCEDURE

The implementation used for our experiments is the one provided by Matthieu Guibert¹ which delivers Powell's original Fortran source code of the algorithm. This Fortran code has been integrated with the BBOB experimental paradigm. In this paper, the maximum number of points $m = \frac{(n+1)(n+2)}{2}$ has been used. Though the scaling of the algorithm time complexity is close to $\mathcal{O}(n^4)$, preliminary experiments showed the full model to deal with noisy functions better than a smaller model. The initial radius ρ_{beg} of the search region has been set to 10, the range of the search space. Preliminary experiments shows very few dependencies on this parameter, given it is not too small (ie. by many orders of magnitude) for the problem considered. A final radius $\rho_{\text{end}} = 10^{-16}$ was chosen close to the limit being the machine precision to prevent numerical errors. The starting point x_0 is chosen uniformly in $[-5, 5]^n$ where n denotes the dimension. The multistart strategy was used with at most 100 restarts to reduce the duration of an experiment. For the same reason, a run is limited to at most $10^4 \times n$ function evaluations. The algorithm used is presented in Figure 1. No parameter tuning was done, the CrE [2] is computed to zero.

3. RESULTS AND DISCUSSION

Results from experiments according to [2] on the benchmarks functions given in [1, 3] are presented in Figures 2 and 3 and in Tables 1 and 2. The algorithm solves some of the moderate noise function $f_{101}, f_{102}, f_{103}, f_{104}, f_{106}$. Else, $f_{105}, f_{107}, f_{109}, f_{112}, f_{113}, f_{115}, f_{125}, f_{127}, f_{128}, f_{130}$ are solved only for dimensions 2 or 3. Noise greatly affects such trust region method, especially the uniform noise model.

4. CPU TIMING EXPERIMENT

For the timing experiment, the proposed algorithm was run on f_8 and restarted until at least 30 seconds have passed (according to Figure 2 in [2]). The experiments were conducted with an Intel Core 2 6700 processor (2.66GHz) on Linux 2.6.24.7. The results were 200, 86, 45, 7.9, 3.7, 36

¹<http://www.inrialpes.fr/bipop/people/guilbert/newuoa/newuoa.html>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

GECCO'09, July 8–12, 2009, Montréal Québec, Canada.
Copyright 2009 ACM 978-1-60558-505-5/09/07 ...\$5.00.

Figure 1: Multistart NEWUOA

```
#include <stdlib.h>
#include <math.h>
#include <stdio.h>
#include "bbobStructures.h"

/* Call to the Fortran function */
extern void newuoa_(unsigned int* n, int* m, double* x0, double* rhobeg,
                  double* rhoend, int* verbose, int* maxfun,
                  double* W, double* ftarget);

/* The Multistart NEWUOA */
void newuoa(unsigned int dim, unsigned int maxfunevals, double ftarget)
{
    int m, iprint = 0, curmaxfun;
    double * x = malloc(sizeof(double) * dim);
    unsigned int iter = 0, i;
    double rhobeg = 10, rhoend = 1e-16;
    /* internal variable of NEWUOA */
    double * w = malloc(1000000 * sizeof(double));

    m = 2 * dim + 1;

    curmaxfun = maxfunevals - fgeneric_evaluations();
    while (curmaxfun > 0 && fgeneric_best() > ftarget && iter < 100)
    {
        /* Generate a starting point */
        for (i = 0; i < dim; i++)
            x[i] = 10. * ((double)rand() / RAND_MAX) - 5.;
        /* Call NEWUOA */
        newuoa_(&dim, &npt, x, &rhobeg, &rhoend, &iprint, &curmaxfun, w, &ftarget);
        /* Update */
        curmaxfun = maxfunevals - fgeneric_evaluations();
        iter++;
    }
    free(x);
    free(w);
}
```

$\times 10^{-3}$ seconds per function for the full model evaluations in dimension 2, 3, 5, 10, 20 and 40 respectively.

5. CONCLUSION

The NEWUOA which is a trust region method was tested with restarts on a noisy testbed. Method based on interpolation are expected to fail on noisy functions. Results of this algorithm do not disagree with this.

Acknowledgments

The first author would like to acknowledge the support, help, and work of the BBOB team with particular kudos to Anne Auger, Steffen Finck and Nikolaus Hansen.

6. REFERENCES

- [1] 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.
- [2] N. Hansen, A. Auger, S. Finck, and R. Ros. Real-parameter black-box optimization benchmarking 2009: Experimental setup. Technical Report RR-6828, INRIA, 2009.
- [3] 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.
- [4] M. J. D. Powell. The NEWUOA software for unconstrained optimization without derivatives. *Large Scale Nonlinear Optimization*, pages 255–297, 2006.

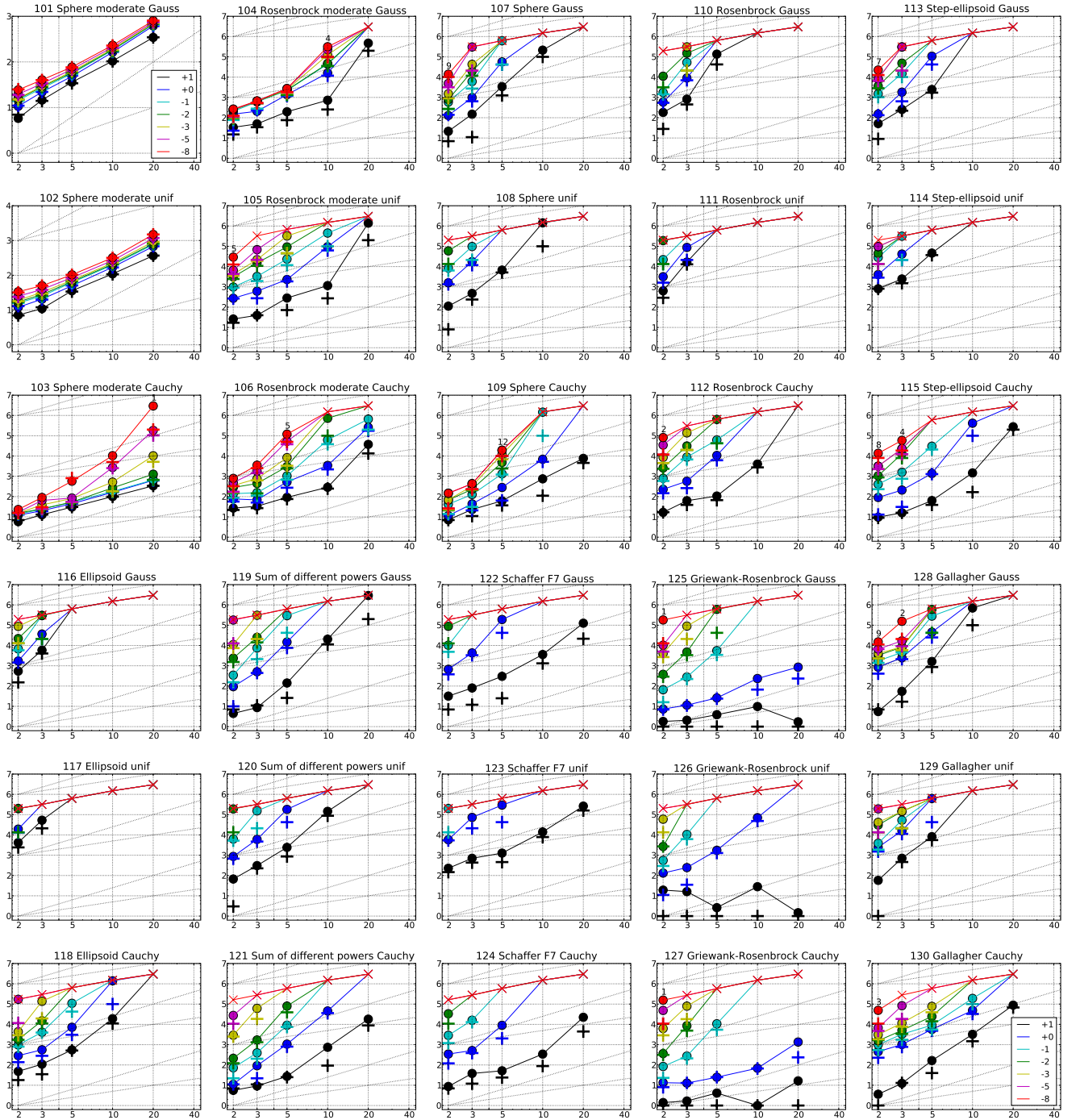


Figure 2: Expected Running Time (ERT, \bullet) to reach $f_{opt} + \Delta f$ and median number of function evaluations of successful trials (+), shown for $\Delta f = 10, 1, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-5}, 10^{-8}$ (the exponent is given in the legend of f_{101} and f_{130}) versus dimension in log-log presentation. The $ERT(\Delta f)$ equals to $\#FEs(\Delta f)$ divided by the number of successful trials, where a trial is successful if $f_{opt} + \Delta f$ was surpassed during the trial. The $\#FEs(\Delta f)$ are the total number of function evaluations while $f_{opt} + \Delta f$ 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 $\#FEs(-\infty)$. 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.

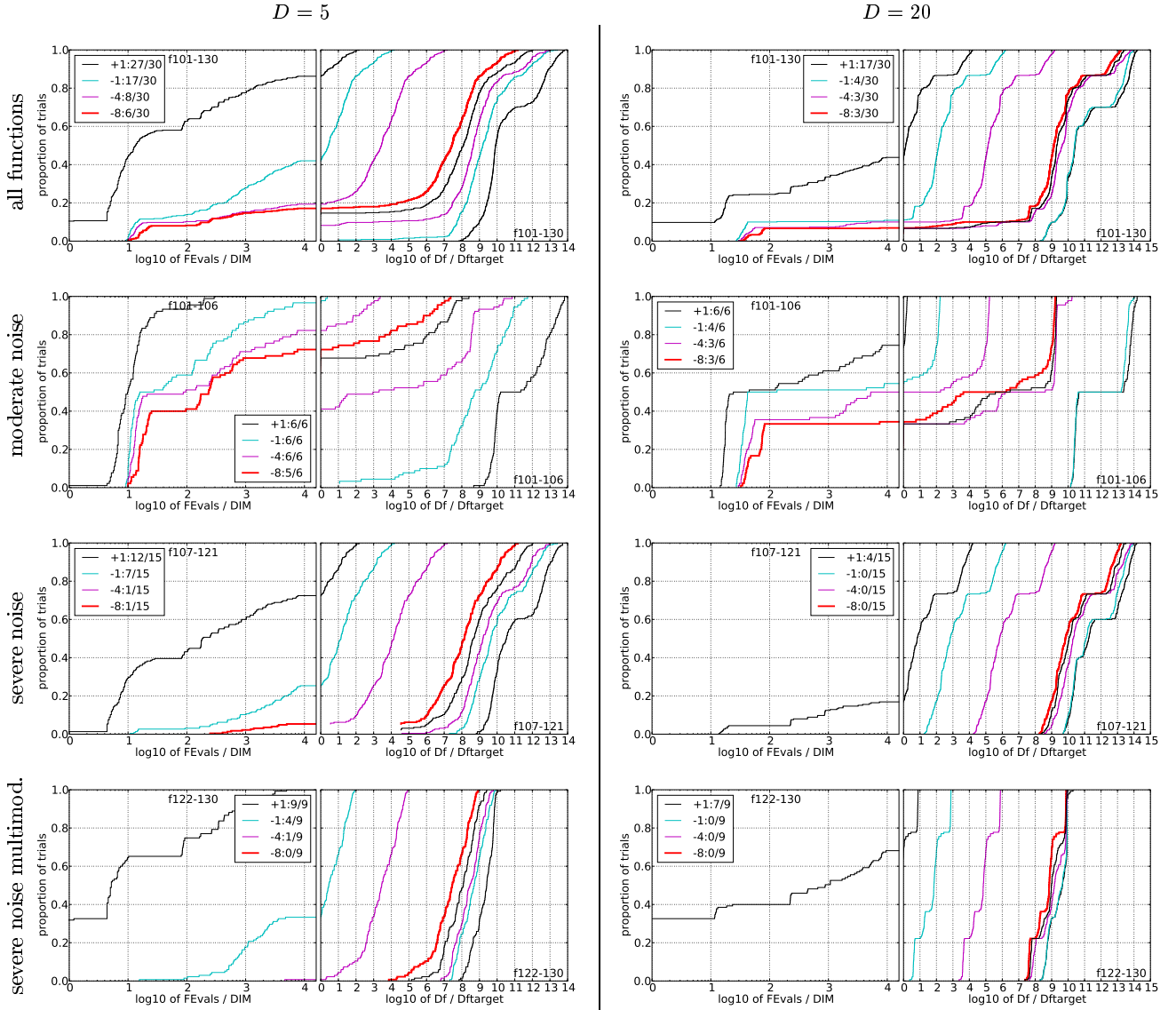


Figure 3: Empirical cumulative distribution functions (ECDFs), plotting the fraction of trials versus running time (left subplots) or versus Δf (right subplots). The thick red line represents the best achieved results. Left subplots: ECDF of the running time (number of function evaluations), divided 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^{-8} for running times of $D, 10D, 100D \dots$ 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	f_{121} in 5-D, N=15, mFE=39962					f_{121} in 20-D, N=15, mFE=200000					Δf	f_{122} in 5-D, N=15, mFE=42104					f_{122} in 20-D, N=15, mFE=200000						
	#	ERT	10%	90%	RT _{succ}	#	ERT	10%	90%	RT _{succ}		#	ERT	10%	90%	RT _{succ}	#	ERT	10%	90%	RT _{succ}		
10	15	2.7e1	2.6e1	3.0e1	2.7e1	15	1.8e4	1.3e4	2.9e4	1.8e4	10	15	3.0e2	7.7e1	4.7e2	3.0e2	10	1.3e5	7.4e4	1.6e5	8.4e4		
1	15	1.1e3	6.8e2	1.6e3	1.1e3	0	<i>53e-1</i>	<i>26e-1</i>	<i>79e-1</i>	1.3e5	1	3	1.9e5	1.9e5	2.1e5	3.6e4	0	<i>76e-1</i>	<i>60e-1</i>	<i>11e+0</i>	1.1e5		
1e-1	15	9.0e3	6.9e3	1.0e4	9.0e3	1e-1	0	<i>17e-1</i>	<i>77e-2</i>	<i>22e-1</i>	3.2e4		
1e-3	0	<i>13e-3</i>	<i>50e-4</i>	<i>42e-3</i>	2.5e4	1e-3		
1e-5	1e-5	
1e-8	1e-8	
Δf	f_{123} in 5-D, N=15, mFE=42805					f_{123} in 20-D, N=15, mFE=200000					Δf	f_{124} in 5-D, N=15, mFE=39598					f_{124} in 20-D, N=15, mFE=200000						
	#	ERT	10%	90%	RT _{succ}	#	ERT	10%	90%	RT _{succ}		#	ERT	10%	90%	RT _{succ}	#	ERT	10%	90%	RT _{succ}		
10	15	1.3e3	8.1e2	2.0e3	1.3e3	8	2.6e5	2.3e5	3.1e5	1.6e5	10	15	5.2e1	2.5e1	7.9e1	5.2e1	15	2.2e4	1.2e4	3.3e4	2.2e4		
1	2	3.0e5	2.7e5	3.2e5	4.3e4	0	<i>99e-1</i>	<i>83e-1</i>	<i>19e+0</i>	1.1e5	1	14	9.0e3	6.1e3	1.3e4	8.9e3	0	<i>66e-1</i>	<i>53e-1</i>	<i>80e-1</i>	7.9e4		
1e-1	0	<i>35e-1</i>	<i>91e-2</i>	<i>46e-1</i>	1.4e4	1e-1	0	<i>47e-2</i>	<i>22e-2</i>	<i>83e-2</i>	1.6e4		
1e-3	1e-3	
1e-5	1e-5
1e-8	1e-8
Δf	f_{125} in 5-D, N=15, mFE=42183					f_{125} in 20-D, N=15, mFE=200000					Δf	f_{126} in 5-D, N=15, mFE=43182					f_{126} in 20-D, N=15, mFE=200000						
	#	ERT	10%	90%	RT _{succ}	#	ERT	10%	90%	RT _{succ}		#	ERT	10%	90%	RT _{succ}	#	ERT	10%	90%	RT _{succ}		
10	15	3.9e0	1.2e0	5.4e0	3.9e0	15	1.7e0	1.0e0	2.5e0	1.7e0	10	15	2.6e0	1.0e0	4.2e0	2.6e0	15	1.5e0	1.0e0	2.4e0	1.5e0		
1	15	2.6e1	2.1e1	3.0e1	2.6e1	15	8.6e2	2.5e2	1.2e3	8.6e2	1	15	1.7e3	1.2e3	2.4e3	1.7e3	0	<i>17e-1</i>	<i>12e-1</i>	<i>19e-1</i>	1.0e5		
1e-1	15	5.6e3	3.7e3	7.3e3	5.6e3	0	<i>44e-2</i>	<i>33e-2</i>	<i>48e-2</i>	7.1e4	1e-1	0	<i>25e-2</i>	<i>16e-2</i>	<i>34e-2</i>	1.6e4		
1e-3	0	<i>41e-3</i>	<i>13e-3</i>	<i>59e-3</i>	1.4e4	1e-3	
1e-5	1e-5
1e-8	1e-8
Δf	f_{127} in 5-D, N=15, mFE=38685					f_{127} in 20-D, N=15, mFE=200000					Δf	f_{128} in 5-D, N=15, mFE=42174					f_{128} in 20-D, N=15, mFE=200000						
	#	ERT	10%	90%	RT _{succ}	#	ERT	10%	90%	RT _{succ}		#	ERT	10%	90%	RT _{succ}	#	ERT	10%	90%	RT _{succ}		
10	15	4.1e0	3.8e0	7.6e0	4.1e0	15	1.6e1	1.0e0	3.2e1	1.6e1	10	15	1.6e3	1.3e3	1.9e3	1.6e3	0	<i>70e+0</i>	<i>65e+0</i>	<i>73e+0</i>	8.9e4		
1	15	2.5e1	2.4e1	2.6e1	2.5e1	15	1.3e3	7.9e2	1.9e3	1.3e3	1	9	4.5e4	3.6e4	5.6e4	2.2e4		
1e-1	14	1.1e4	6.7e3	1.3e4	1.0e4	0	<i>40e-2</i>	<i>37e-2</i>	<i>45e-2</i>	7.9e4	1e-1	2	2.8e5	2.6e5	3.0e5	4.2e4	
1e-3	0	<i>59e-3</i>	<i>31e-3</i>	<i>84e-3</i>	2.0e4	1e-3	0	<i>16e-2</i>	<i>18e-3</i>	<i>20e-1</i>	1.4e4	
1e-5	1e-5
1e-8	1e-8
Δf	f_{129} in 5-D, N=15, mFE=42757					f_{129} in 20-D, N=15, mFE=200000					Δf	f_{130} in 5-D, N=15, mFE=39955					f_{130} in 20-D, N=15, mFE=200000						
	#	ERT	10%	90%	RT _{succ}	#	ERT	10%	90%	RT _{succ}		#	ERT	10%	90%	RT _{succ}	#	ERT	10%	90%	RT _{succ}		
10	15	8.2e3	6.8e3	1.1e4	8.2e3	0	<i>75e+0</i>	<i>72e+0</i>	<i>76e+0</i>	6.3e4	10	15	1.6e2	1.1e2	2.2e2	1.6e2	14	8.9e4	7.4e4	1.1e5	7.7e4		
1	1	6.2e5	6.1e5	6.4e5	4.3e4	1	15	5.7e3	4.6e3	7.3e3	5.7e3	0	<i>74e-1</i>	<i>20e-1</i>	<i>10e+0</i>	1.4e5		
1e-1	0	<i>55e-1</i>	<i>20e-1</i>	<i>78e-1</i>	2.0e4	1e-1	14	9.1e3	6.4e3	1.5e4	8.6e3	
1e-3	1e-3	6	7.7e4	6.6e4	8.4e4	3.2e4	
1e-5	1e-5	0	<i>14e-4</i>	<i>20e-5</i>	<i>56e-3</i>	2.0e4	
1e-8	1e-8

Table 2: 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 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 (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 2 for the names of functions.