

Benchmarking the Nelder-Mead Downhill Simplex Algorithm With Many Local Restarts

Example Paper*

The BBOBies

ABSTRACT

We have benchmarked the Nelder-Mead downhill simplex method on the noise-free BBOB-2010 testbed [2]. A multi-start strategy is applied on two levels. On a local level, at least ten restarts are conducted with a small number of iterations and reshaped simplex. On the global level independent restarts are launched until $10^5 D$ function evaluations are exceeded, for dimension $D \geq 20$ ten times less. For low search space dimensions the algorithm shows very good results on many functions. It solves 24, 18, 11 and 7 of 24 functions in 2, 5, 10 and 40-D.

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, Direct search, Evolutionary computation, Simplex downhill

1. INTRODUCTION

The Nelder-Mead method [6] is a real-parameter black-box optimization method that operates, similar to many evolutionary algorithms, on a set of solution points using only the *ranking* of solution. The latter implies that the algorithm is invariant under order-preserving transformations of the objective function values. The Nelder-Mead algorithm is independent of the choice of coordinate system and therefore exhibits more attractive invariance properties. In

*Submission deadline: March 25th.

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'10, July 7–11, 2010, Portland Oregon, United States of America.
Copyright 2010 ACM X-XXXXXX-XX-X/XX/XX ...\$5.00.

contrast to most evolutionary algorithms, the Nelder-Mead algorithm does not solely resort to selection for improving the average solution and it does not contain stochastic elements.

2. THE ALGORITHM

The Nelder-Mead method [6] operates on a set of $D + 1$ solution points, a simplex, where D is the search space dimension. Generally, a new solution is constructed by reflecting the worst solution on the center of the remaining D solutions. Depending on the quality of the new solution additional operations are performed, but details are omitted here. For solving global optimization problems sophisticated restart procedures have been proposed for example in [5]. In this paper, a few different restart mechanisms are used.

1. local restarts, where the recent best solution is used as initial solution for the restart and the projection of the range of the recent simplex is used for initialization of the new simplex. The default initialization procedure is applied, which places new simplex points by changing one coordinate at a time. At least ten restarts are conducted and the maximum number of iterations is $200 \times \sqrt{D}$ for all but the last. This number is as small that even on the sphere function f_1 usually some local restarts are conducted.
2. local restarts as above, where some additional perturbation is added to the simplex.
3. global restarts, which are completely independent of previous results.

The applied reshaping exploits the given coordinate system and improves the local search abilities for $D \geq 10$ on functions with comparatively low parameter dependencies. It is also effective on the Rosenbrock function in moderate dimension. All details are given in Figure 1 and in the next sections.

3. PARAMETER TUNING

Exemplary online experiments on f_2 and f_8 have been conducted to verify reasonable constants for the maximum iteration number of local restarts (constant 200 in Figure 1) and the number of local restarts (constants 10 and 0.1). The chosen dependencies on D have not been verified. We added add-hoc termination criteria, where `To1X` turned out to be useful. At most between $10^5 D$ and $2 \times 10^5 D$ function evaluations are conducted (input parameter `maxfunevals` was

Figure 1: Multistart procedure of Nelder-Mead in Matlab

```
function [x, ilaunch, f] = MY_OPTIMIZER(FUN, DIM, ftarget, maxfunevals)
% minimizes FUN in DIM dimensions by multistarts of fminsearch.
% ftarget and maxfunevals are additional external termination conditions,
% where at most 2 * maxfunevals function evaluations are conducted.
% fminsearch was modified to take as input variable usual_delta to
% generate the first simplex.

% set options, make sure we always terminate
% with restarts up to 2*maxfunevals are allowed
options = optimset('MaxFunEvals', min(1e9*DIM, maxfunevals), ...
    'MaxIter', 2e3*DIM, ... % overwritten later
    'Tolfun', 1e-11, ...
    'TolX', 1e-11, ...
    'OutputFcn', @callback, ...
    'Display', 'off');

ilocal = 0;

% multistart such that ftarget is reached with reasonable prob.
for ilaunch = 1:1e5; % relaunch optimizer up to 1e5 times
    % set initial conditions
    ilocal = ilocal + 1;
    if ilocal == 1 % (re-)start from scratch
        xstart = 8 * rand(DIM, 1) - 4; % random start solution
        usual_delta = 2;
        options = optimset(options, 'MaxIter', floor(200*sqrt(DIM)));
    else % refining restart run
        xstart = x; % try to improve found solution
        usual_delta = 10 * range(v,2); % a bit of regularization in given coordinate sys
        if rand(1,1) < 0.2 % a bit of desperation
            usual_delta = usual_delta + (1/ilocal) * (0.1/ilocal).^rand(DIM,1);
        end
        if rand(1,1) < 0.1 * (ilocal-10)/sqrt(DIM) % final run
            options = optimset(options, 'MaxIter', 500*DIM); % long run
            ilocal = 0; % real restart after this run
        end
    end
end

% try fminsearch from Matlab, modified to take usual_delta as arg
[x,f,e,o,v] = fminsearch_mod(FUN, xstart, usual_delta, options);
% disp(sprintf('%d %d: %e %e %e', ilocal, feval(FUN, 'evaluations'), f-ftarget, ...
% min(usual_delta), max(usual_delta)/min(usual_delta)));

if feval(FUN, 'fbest') < ftarget || ...
    feval(FUN, 'evaluations') >= maxfunevals
    break;
end
% if useful, modify more options here for next launch
end

function stop = callback(x, optimValues, state)
    stop = false;
    if optimValues.fval < ftarget
        stop = true;
    end
end

end
```

set to $10^5 D$), for $D \geq 20$ ten times less. The final parameter setting were identical on all functions and therefore the crafting effort [3] is $\text{CrE} = 0$.

4. METHODS

We have used the matlab function `fminsearch`, Revision 1.21.4.7, and made the variable `usual_delta` an additional input parameter. Onto this algorithm we have applied a multistart strategy as given in Figure 1. This procedure has been benchmarked on the noiseless BBOB-2009 testbed [1, 4] according to the experimental design from [3].

The initial solution from which the first simplex is constructed was chosen uniformly distributed in $[4, 4]^D$ or as the former best solution.

5. CPU TIMING EXPERIMENT

For the timing experiment the same multistart algorithm was run on f_8 and restarted until at least 30 seconds had passed (according to Figure 2 in [3]). These experiments have been conducted with an Intel dual core T5600 processor with 1.8 GHz under Linux 2.6.27-11 using Matlab R2008a. The results were 6.2; 5.8; 5.6; 5.7; 5.8; 5.9 and 6.3×10^{-4} seconds per function evaluation in dimension 2; 3; 5; 10; 20; 40 and 80, respectively. Up to 80-D a dependency of CPU time on the search space dimensionality is hardly visible.

6. RESULTS AND DISCUSSION

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

The method solves 24, 23, 18, 11, 8 and 7 out of 24 functions in 2, 3, 5, 10, 20 and 40-D (Figure 2). The expected number of function evaluations to reach a given target function value scales often quadratically with the dimension on unimodal functions and on functions 21 and 22 (Figure 2). The scaling is remarkably better only on f_2 in larger dimension, presumably due to the used simplex reshaping. The scaling is often worse than quadratical, not only but in particular on multi-modal functions, and the algorithm fails within the given budget for larger dimension.

Figures 3 and 4 reveal the algorithms main weaknesses on the multimodal functions 15–19. These multimodal functions have a large number of optima and an independent multistart algorithm cannot discover the overall function structure. The performance also deteriorates in larger dimension in particular on the ill-conditioned functions 10–14. In contrast, the performance is very good on the low dimensional ill-conditioned functions.

7. CONCLUSION

The Nelder-Mead algorithm, as implemented in Matlab, equipped with an additional input vector and applied in a multistart fashion, is a fast and reliable black-box search algorithm for low dimensional search spaces. The applied reshaping of the simplex extends its efficiency to larger dimension only for unimodal functions with little dependencies between variables. The multiple independent restarts allow to searching unstructured multi-modal landscapes comparatively effective, while a global topography within a multi-modal or rugged landscape is not well exploited.

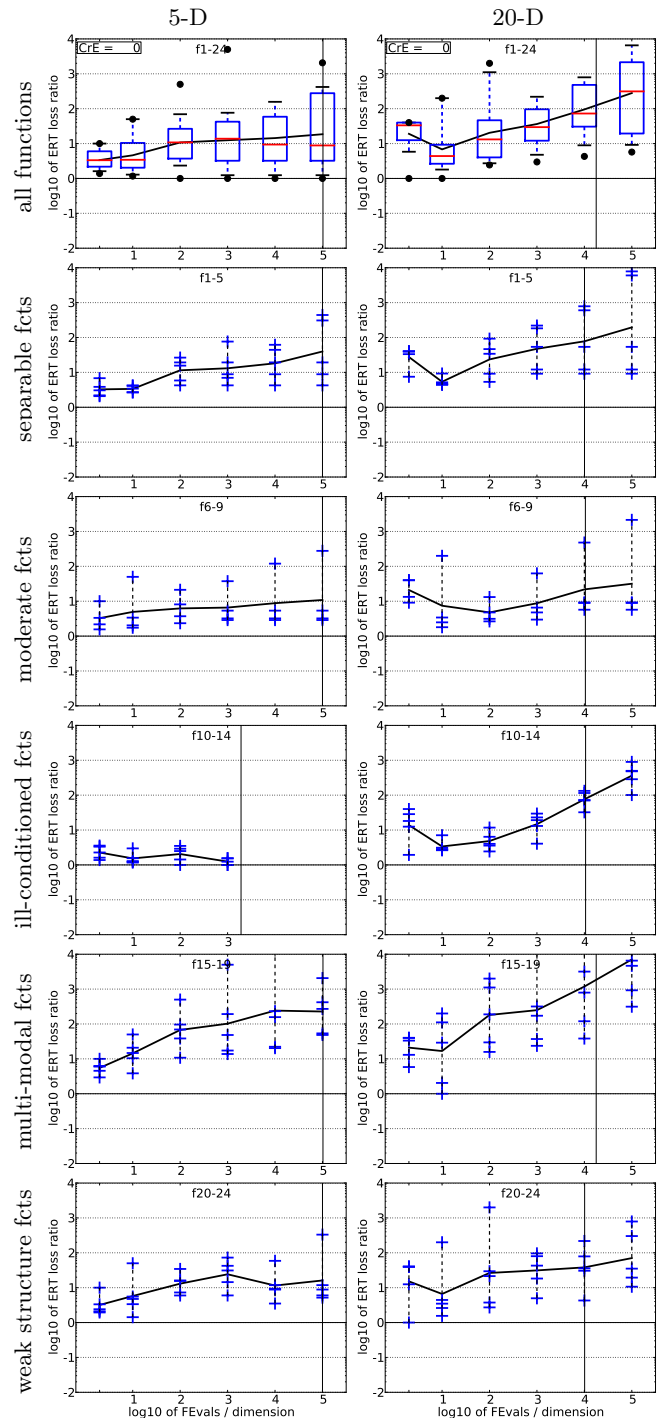


Figure 4: ERT loss ratio versus given budget FEvals. The target value f_t for ERT (see Figure 2) is the smallest (best) recorded function value such that $\text{ERT}(f_t) \leq \text{FEvals}$ for the presented algorithm. Shown is FEvals divided by the respective best $\text{ERT}(f_t)$ from BBOB-2009 for functions f_1 – f_{24} in 5-D and 20-D. Each ERT is multiplied by $\exp(\text{CrE})$ correcting for the parameter crafting effort. 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 this function subset.

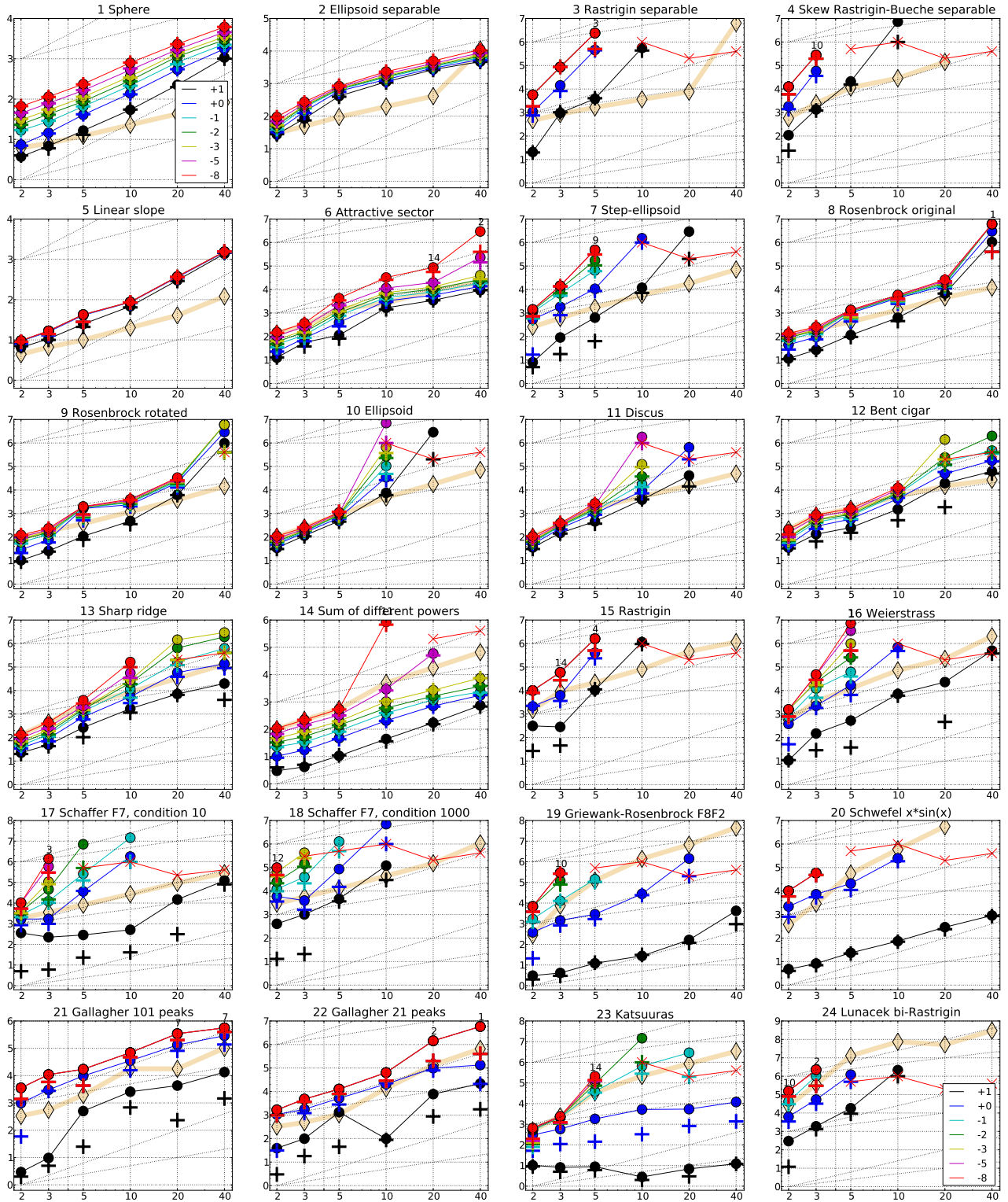


Figure 2: Expected Running Time (ERT, ●) to reach $f_{opt} + \Delta f$ and median number of f -evaluations from successful trials (+), for $\Delta f = 10^{\{+1,0,-1,-2,-3,-5,-8\}}$ (the exponent is given in the legend of f_1 and f_{24}) versus dimension in log-log presentation. For each function and dimension, $ERT(\Delta f)$ equals to $\#FEs(\Delta f)$ divided by the number of successful trials, where a trial is successful if $f_{opt} + \Delta f$ was surpassed. The $\#FEs(\Delta f)$ are the total number (sum) of f -evaluations while $f_{opt} + \Delta f$ was not surpassed in the trial, from all (successful and unsuccessful) trials, and f_{opt} is the optimal function value. Crosses (×) indicate the total number of f -evaluations, $\#FEs(-\infty)$, divided by the number of trials. Numbers above ERT-symbols indicate the number of successful trials. Y-axis annotations are decimal logarithms. The thick light line with diamonds shows the single best results from BBOB-2009 for $\Delta f = 10^{-8}$. Additional grid lines show linear and quadratic scaling.

f_1 in 5-D, N=15, mFE=279					f_1 in 20-D, N=15, mFE=2932					f_2 in 5-D, N=15, mFE=1678					f_2 in 20-D, N=15, mFE=6101						
Δf	#	ERT	10%	90%	RT _{succ}	#	ERT	10%	90%	RT _{succ}	#	ERT	10%	90%	RT _{succ}	#	ERT	10%	90%	RT _{succ}	
10	15	1.7e1	1.0e1	2.5e1	1.7e1	15	2.2e2	1.6e2	3.2e2	2.2e2	10	15	4.1e2	1.8e2	7.4e2	4.1e2	15	2.7e3	2.1e3	2.9e3	2.7e3
1	15	4.1e1	2.1e1	6.1e1	4.1e1	15	5.4e2	3.3e2	7.5e2	5.4e2	1	15	5.9e2	3.4e2	7.7e2	5.9e2	15	3.0e3	2.5e3	3.4e3	3.0e3
1e-1	15	6.6e1	5.0e1	8.3e1	6.6e1	15	8.2e2	6.1e2	1.3e3	8.2e2	1e-1	15	6.6e2	4.3e2	7.9e2	6.6e2	15	3.3e3	2.7e3	3.7e3	3.3e3
1e-3	15	1.1e2	8.1e1	1.4e2	1.1e2	15	1.4e3	1.0e3	1.6e3	1.4e3	1e-3	15	7.1e2	4.8e2	8.3e2	7.1e2	15	3.8e3	3.3e3	4.1e3	3.8e3
1e-5	15	1.6e2	1.5e2	1.8e2	1.6e2	15	1.7e3	1.5e3	2.1e3	1.7e3	1e-5	15	7.6e2	5.4e2	8.8e2	7.6e2	15	4.1e3	3.8e3	4.5e3	4.1e3
1e-8	15	2.4e2	2.1e2	2.6e2	2.4e2	15	2.3e3	2.1e3	2.8e3	2.3e3	1e-8	15	8.4e2	6.0e2	9.5e2	8.4e2	15	5.0e3	4.2e3	5.7e3	5.0e3
f_3 in 5-D, N=15, mFE=500412					f_3 in 20-D, N=15, mFE=201261					f_4 in 5-D, N=15, mFE=500615					f_4 in 20-D, N=15, mFE=201200						
Δf	#	ERT	10%	90%	RT _{succ}	#	ERT	10%	90%	RT _{succ}	#	ERT	10%	90%	RT _{succ}	#	ERT	10%	90%	RT _{succ}	
10	15	3.9e3	8.9e1	8.9e3	3.9e3	0	<i>81e+0</i>	<i>67e+0</i>	<i>97e+0</i>	<i>6.7e4</i>	10	15	2.1e4	7.3e3	4.2e4	2.1e4	0	<i>13e+1</i>	<i>11e+1</i>	<i>18e+1</i>	1.4e5
1	11	4.6e5	9.7e4	9.2e5	2.8e5						1	0	<i>30e-1</i>	<i>20e-1</i>	<i>40e-1</i>	1.8e5					
1e-1	3	2.4e6	3.6e5	5.8e6	4.0e5						1e-1										
1e-3	3	2.4e6	4.1e5	4.6e6	4.0e5						1e-3										
1e-5	3	2.4e6	4.4e5	5.2e6	4.0e5						1e-5										
1e-8	3	2.4e6	4.1e5	5.1e6	4.0e5						1e-8										
f_5 in 5-D, N=15, mFE=131					f_5 in 20-D, N=15, mFE=629					f_6 in 5-D, N=15, mFE=9819					f_6 in 20-D, N=15, mFE=210040						
Δf	#	ERT	10%	90%	RT _{succ}	#	ERT	10%	90%	RT _{succ}	#	ERT	10%	90%	RT _{succ}	#	ERT	10%	90%	RT _{succ}	
10	15	2.5e1	1.6e1	3.3e1	2.5e1	15	3.0e2	2.3e2	3.5e2	3.0e2	10	15	1.1e2	3.4e1	1.6e2	1.1e2	15	3.5e3	2.0e3	5.1e3	3.5e3
1	15	4.1e1	1.8e1	1.2e2	4.1e1	15	3.6e2	2.5e2	4.2e2	3.6e2	1	15	4.0e2	7.0e1	8.0e2	4.0e2	15	5.7e3	3.7e3	7.7e3	5.7e3
1e-1	15	4.2e1	1.8e1	4.9e1	4.2e1	15	3.7e2	2.7e2	4.1e2	3.7e2	1e-1	15	8.0e2	1.6e2	1.6e3	8.0e2	15	7.9e3	5.4e3	1.2e4	7.9e3
1e-3	15	4.2e1	2.1e1	1.1e2	4.2e1	15	3.7e2	2.7e2	4.5e2	3.7e2	1e-3	15	1.4e3	8.2e2	2.3e3	1.4e3	15	1.3e4	9.1e3	1.6e4	1.3e4
1e-5	15	4.2e1	2.1e1	1.1e2	4.2e1	15	3.7e2	2.7e2	6.3e2	3.7e2	1e-5	15	2.1e3	1.1e3	2.9e3	2.1e3	15	2.0e4	1.4e4	2.1e4	2.0e4
1e-8	15	4.2e1	1.8e1	8.0e1	4.2e1	15	3.7e2	2.7e2	4.1e2	3.7e2	1e-8	15	4.3e3	2.5e3	6.2e3	4.3e3	14	8.7e4	2.3e4	1.8e5	7.2e4
f_7 in 5-D, N=15, mFE=500446					f_7 in 20-D, N=15, mFE=201231					f_8 in 5-D, N=15, mFE=5587					f_8 in 20-D, N=15, mFE=51755						
Δf	#	ERT	10%	90%	RT _{succ}	#	ERT	10%	90%	RT _{succ}	#	ERT	10%	90%	RT _{succ}	#	ERT	10%	90%	RT _{succ}	
10	15	6.4e2	1.4e1	2.2e3	6.4e2	1	2.9e6	3.1e5	6.2e6	1.1e5	10	15	1.2e2	5.5e1	2.8e2	1.2e2	15	6.7e3	5.8e3	9.4e3	6.7e3
1	15	1.1e4	4.6e1	2.3e4	1.1e4	0	<i>16e+0</i>	<i>11e+0</i>	<i>26e+0</i>	<i>7.5e4</i>	1	15	1.0e3	1.2e2	5.3e3	1.0e3	15	1.5e4	7.5e3	2.4e4	1.5e4
1e-1	15	6.5e4	1.3e4	1.4e5	6.5e4						1e-1	15	1.1e3	2.2e2	2.7e3	1.1e3	15	1.7e4	1.0e4	3.5e4	1.7e4
1e-3	9	4.8e5	3.8e4	1.0e6	1.5e5						1e-3	15	1.2e3	3.1e2	2.7e3	1.2e3	15	2.1e4	1.4e4	4.9e4	2.1e4
1e-5	9	4.8e5	3.8e4	1.2e6	1.5e5						1e-5	15	1.3e3	3.7e2	2.8e3	1.3e3	15	2.3e4	1.8e4	4.2e4	2.3e4
1e-8	9	4.8e5	3.8e4	1.1e6	1.5e5						1e-8	15	1.4e3	4.2e2	2.9e3	1.4e3	15	2.6e4	2.0e4	4.6e4	2.6e4
f_9 in 5-D, N=15, mFE=7218					f_9 in 20-D, N=15, mFE=93783					f_{10} in 5-D, N=15, mFE=1999					f_{10} in 20-D, N=15, mFE=211907						
Δf	#	ERT	10%	90%	RT _{succ}	#	ERT	10%	90%	RT _{succ}	#	ERT	10%	90%	RT _{succ}	#	ERT	10%	90%	RT _{succ}	
10	15	1.1e2	5.3e1	2.1e2	1.1e2	15	6.2e3	3.9e3	1.1e4	6.2e3	10	15	4.8e2	2.0e2	8.2e2	4.8e2	1	2.9e6	2.5e5	7.0e6	4.0e4
1	15	1.6e3	1.5e2	6.2e3	1.6e3	15	2.1e4	9.4e3	4.7e4	2.1e4	1	15	6.6e2	4.5e2	1.0e3	6.6e2	0	<i>30e+0</i>	<i>11e+0</i>	<i>70e+0</i>	1.4e5
1e-1	15	1.8e3	1.3e2	6.3e3	1.8e3	15	2.4e4	1.4e4	3.8e4	2.4e4	1e-1	15	8.3e2	4.5e2	1.2e3	8.3e2					
1e-3	15	1.9e3	3.8e2	6.4e3	1.9e3	15	2.9e4	1.9e4	5.2e4	2.9e4	1e-3	15	9.2e2	5.0e2	1.4e3	9.2e2					
1e-5	15	1.9e3	4.3e2	7.1e3	1.9e3	15	3.2e4	1.9e4	7.2e4	3.2e4	1e-5	15	1.0e3	6.2e2	1.7e3	1.0e3					
1e-8	15	2.0e3	5.4e2	6.9e3	2.0e3	15	3.4e4	2.1e4	7.4e4	3.4e4	1e-8	15	1.1e3	6.2e2	1.9e3	1.1e3					
f_{11} in 5-D, N=15, mFE=5646					f_{11} in 20-D, N=15, mFE=212233					f_{12} in 5-D, N=15, mFE=4154					f_{12} in 20-D, N=15, mFE=205387						
Δf	#	ERT	10%	90%	RT _{succ}	#	ERT	10%	90%	RT _{succ}	#	ERT	10%	90%	RT _{succ}	#	ERT	10%	90%	RT _{succ}	
10	15	4.6e2	1.9e2	8.7e2	4.6e2	15	4.1e4	1.0e4	7.8e4	4.1e4	10	15	2.5e2	1.2e2	4.2e2	2.5e2	15	1.9e4	1.6e3	5.8e4	1.9e4
1	15	1.0e3	6.9e2	1.7e3	1.0e3	4	6.5e5	5.3e4	1.7e6	8.7e4	1	15	5.8e2	1.7e2	1.1e3	5.8e2	15	4.9e4	2.1e3	9.8e4	4.9e4
1e-1	15	1.3e3	9.8e2	1.8e3	1.3e3	0	<i>16e-1</i>	<i>70e-2</i>	<i>35e-1</i>	<i>9.7e4</i>	1e-1	15	8.0e2	4.0e2	1.4e3	8.0e2	11	1.6e5	1.6e4	4.7e5	8.2e4
1e-3	15	1.8e3	1.2e3	2.7e3	1.8e3						1e-3	15	1.0e3	4.8e2	2.3e3	1.0e3	2	1.4e6	1.0e5	3.3e6	8.4e4
1e-5	15	2.2e3	1.4e3	2.8e3	2.2e3						1e-5	15	1.3e3	5.2e2	2.0e3	1.3e3	0	<i>54e-4</i>	<i>87e-5</i>	<i>35e-2</i>	1.0e5
1e-8	15	2.7e3	1.6e3	4.4e3	2.7e3						1e-8	15	1.6e3	6.6e2	2.6e3	1.6e3	0	<i>44e-7</i>	<i>37e-7</i>	<i>66e-7</i>	1.2e5
f_{13} in 5-D, N=15, mFE=9607					f_{13} in 20-D, N=15, mFE=208304					f_{14} in 5-D, N=15, mFE=680					f_{14} in 20-D, N=15, mFE=209057						
Δf	#	ERT	10%	90%	RT _{succ}	#	ERT	10%	90%	RT _{succ}	#	ERT	10%	90%	RT _{succ}	#	ERT	10%	90%	RT _{succ}	
10	15	2.7e2	7.3e1	7.8e2	2.7e2	15	7.1e3	1.4e3	1.9e4	7.1e3	10	15	1.1e1	1.0e0	2.2e1	1.1e1	15	1.7e2	8.8e1	2.7e2	1.7e2
1	15	7.5e2	1.4e2	1.3e3	7.5e2	15	5.8e4	8.5e3	1.6e5	5.8e4	1	15	4.7e1	3.0e1	7.7e1	4.7e1	15	7.1e2	3.2e2	1.2e3	7.1e2
1e-1	15	1.3e3	6.1e2	2.2e3	1.3e3	11	1.6e5	2.0e4	3.9e5	9.0e4	1e-1	15	8.8e1	6.1e1	1.2e2	8.8e1	15	1.2e3	6.8e2	1.4e3	1.2e3
1e-3	15	1.7e3	9.6e2	2.3e3	1.7e3	2	1.4e6	1.8e5	3.4e6	1.3e5	1e-3	15	1.9e2	1.6e2	2.3e2	1.9e2	15	2.7e3	2.1e3	3.2e3	2.7e3
1e-5	15	2.2e3	1.0e3	3.6e3	2.2e3	0	<i>35e-3</i>	<i>34e-5</i>	<i>62e-2</i>	1.3e5	1e-5	15	3.3e2	2.9e2	4.1e2	3.3e2	15	5.9e4	1.8e4	1.2e5	5.9e4
1e-8	15	3.9e3	1.5e3	7.7e3	3.9e3						1e-8	15	5.4e2	4.5e2	6.3e2	5.4e2	0	<i>44e-7</i>	<i>37e-7</i>	<i>66e-7</i>	1.2e5
f_{15} in 5-D, N=15, mFE=500566					f_{15} in 20-D, N=15, mFE=201195					f_{16} in 5-D, N=15, mFE=500609					f_{16} in 20-D, N=15, mFE=204125						
Δf	#	ERT	10%	90%	RT _{succ}	#	ERT	10%	90%	RT _{succ}	#	ERT	10%	90%	RT _{succ}	#	ERT	10%	90%	RT _{succ}	
10	15	1.0e4	2.4e2	2.1e4	1.0e4	0	<i>80e+0</i>	<i>58e+0</i>	<i>93e+0</i>	1.3e5	10	15	5.3e2	2.2e1	3.4e3	5.3e2	15	2.3e4	1.7e2	5.9e4	2.3e4
1	15	4.0e5	1.8e4	1.0e6	1.5e5						1	15	1.7e4	6.1e1	5.5e4	1.7e4	0	<i>47e-1</i>	<i>29e-1</i>	<i>66e-1</i>	1.6e5
1e																					

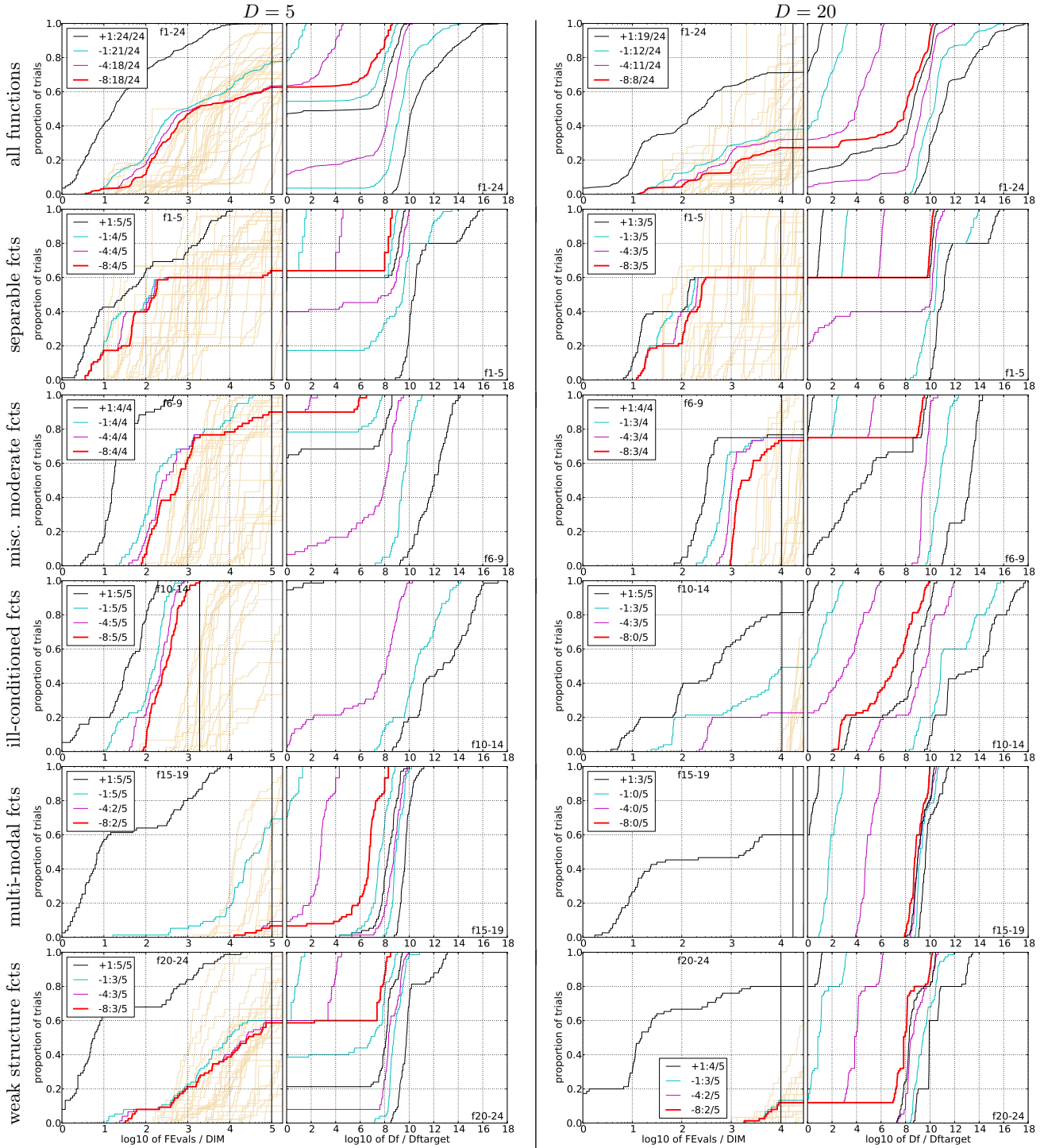


Figure 3: Empirical cumulative distribution functions (ECDFs), plotting the fraction of trials versus running time (left subplots) or versus Δf (right subplots). The thick red line represents the best achieved results. Left subplots: ECDF of the running time (number of function evaluations), divided 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^k (upper left lines in continuation of the left subplot), and 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). 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. Light brown lines in the background show ECDFs for target value 10^{-8} of all algorithms benchmarked during BBOB-2009.

Table 2: ERT loss ratio (see Figure 4) compared to the respective best result from BBOB-2009 for budgets given in the first column. The last row RL_{US}/D gives the number of function evaluations in unsuccessful runs divided by dimension. Shown are the smallest, 10%-ile, 25%-ile, 50%-ile, 75%-ile and 90%-ile value (smaller values are better).

<i>f1-f24 in 5-D, maxFE/D=102505</i>						
#FEs/D	best	10%	25%	med	75%	90%
2	1.4	1.6	2.1	3.2	5.3	10
10	1.2	1.3	1.9	3.4	8.0	50
100	1.0	2.2	3.6	9.4	24	72
1e3	1.0	1.2	3.1	11	40	88
1e4	1.0	1.2	3.1	9.1	52	1.6e2
1e5	1.0	1.2	3.1	8.8	2.7e2	4.2e2
RL_{US}/D	1e5	1e5	1e5	1e5	1e5	1e5
<i>f1-f24 in 20-D, maxFE/D=17757</i>						
#FEs/D	best	10%	25%	med	75%	90%
2	1.0	5.4	12	33	40	40
10	1.0	1.8	2.6	3.9	8.2	2.0e2
100	2.4	2.7	3.9	12	40	1.2e3
1e3	3.0	4.7	11	27	88	2.3e2
1e4	4.3	8.6	21	72	3.5e2	1.0e3
1e5	5.7	9.1	16	3.1e2	1.5e3	6.7e3
RL_{US}/D	1e4	1e4	1e4	1e4	1e4	1e4

8. REFERENCES

- [1] 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.
- [2] N. Hansen. Benchmarking the Nelder-Mead downhill simplex algorithm with many local restarts. In F. Rothlauf, editor, *GECCO (Companion)*, pages 2403–2408. ACM, 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.
- [5] M. Luersen and R. Le Riche. Globalized Nelder-Mead method for engineering optimization. *Computers and Structures*, 82(23-26):2251–2260, 2004.
- [6] J. Nelder and R. Mead. The downhill simplex method. *Computer Journal*, 7:308–313, 1965.