

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

Nikolaus Hansen  
Microsoft Research–INRIA Joint Centre  
28 rue Jean Rostand  
91893 Orsay Cedex, France  
Nikolaus.Hansen@inria.fr

## ABSTRACT

We benchmark the Nelder-Mead downhill simplex method on the noise-free BBOB-2009 testbed. A multistart 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 [5] 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 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 [5] 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 [4]. 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` is 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 [2] is `CrE` = 0.

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 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

```

---

## 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, 3] according to the experimental design from [2].

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 [2]). 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  $6.3 \times 10^{-4}$  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

The results are presented in Table 1 and Figures 2 and 3. 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.

Figure 3 reveals 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 is also poor 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 multimodal or rugged landscape is not well exploited.

## Acknowledgments

The author would like to acknowledge the great and hard work of the BBOB team with particular kudos to Raymond Ros, Steffen Finck and Anne Auger, and Anne Auger and Marc Schoenauer for their kind and persistent support.

## 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, 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: Noiseless functions definitions. Technical Report RR-6829, INRIA, 2009.
- [4] M. Luersen and R. Le Riche. Globalized Nelder-Mead method for engineering optimization. *Computers and Structures*, 82(23-26):2251–2260, 2004.
- [5] J. Nelder and R. Mead. The downhill simplex method. *Computer Journal*, 7:308–313, 1965.

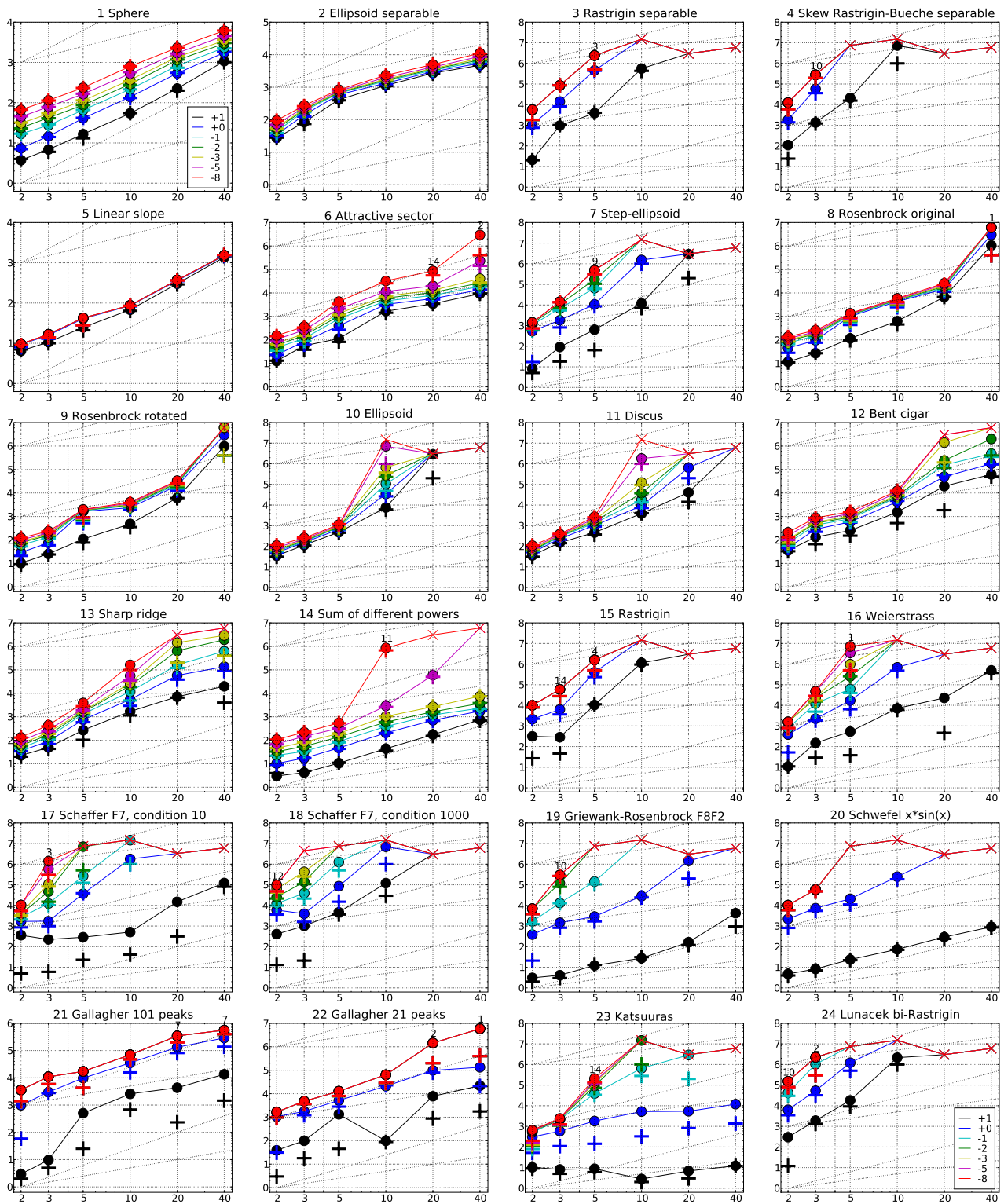


Figure 2: Expected Running Time (ERT, ●) to reach  $f_{\text{opt}} + \Delta f$  and median number of function evaluations of successful trials (+), shown for  $\Delta f = 10, 1, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-5}, 10^{-8}$  (the exponent is given in the legend of  $f_1$  and  $f_{24}$ ) versus dimension in log-log presentation. The ERT( $\Delta f$ ) equals to  $\#FEs(\Delta f)$  divided by the number of successful trials, where a trial is successful if  $f_{\text{opt}} + \Delta f$  was surpassed during the trial. The  $\#FEs(\Delta f)$  are the total number of function evaluations while  $f_{\text{opt}} + \Delta f$  was not surpassed during the trial from all respective trials (successful and unsuccessful), and  $f_{\text{opt}}$  denotes the optimal function value. Crosses (×) 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.



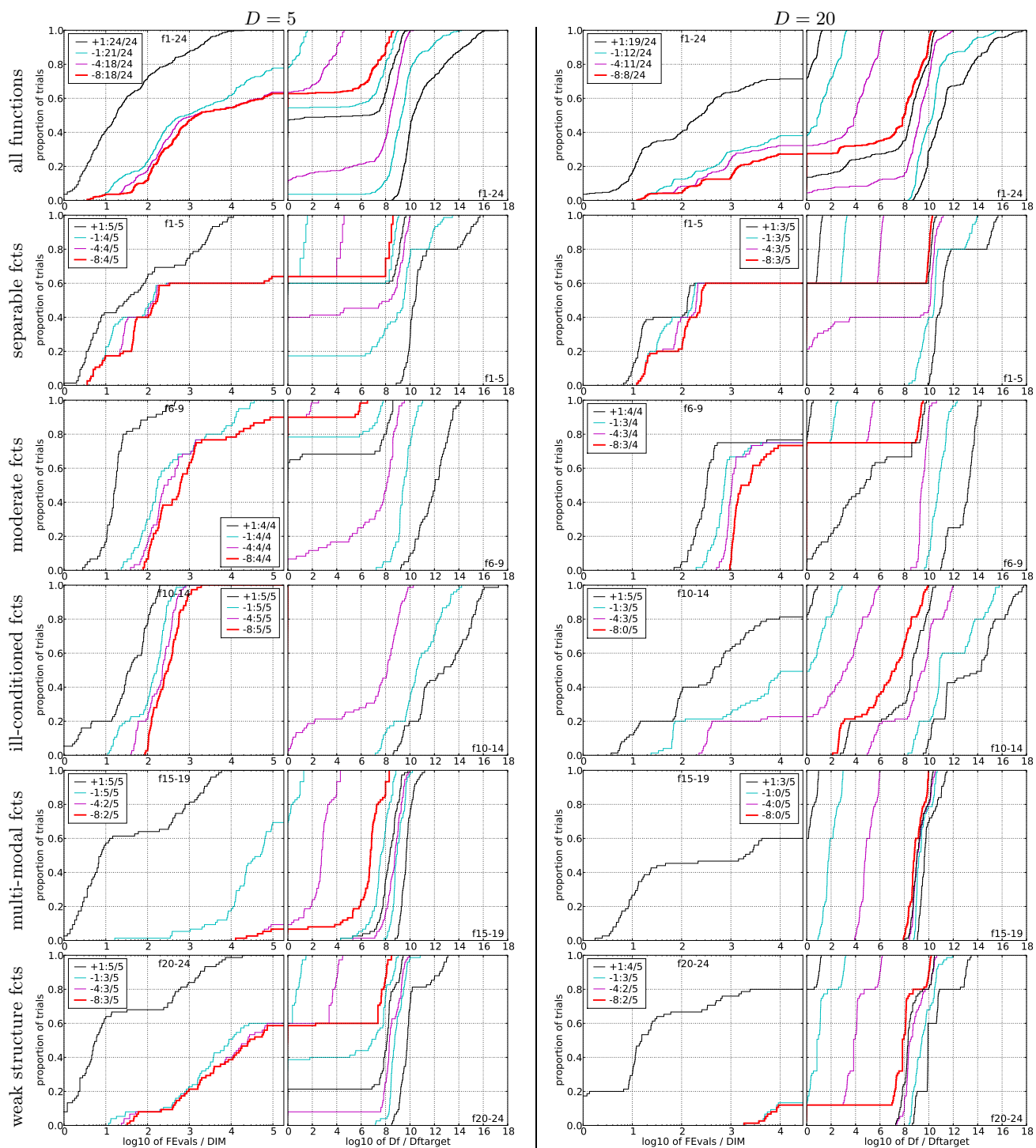


Figure 3: Empirical cumulative distribution functions (ECDFs), plotting the fraction of trials versus running time (left subplots) or versus  $\Delta f$  (right subplots). The thick red line represents the best achieved results. Left subplots: ECDF of the running time (number of function evaluations), 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  $\Delta f$  divided by  $10^k$  (upper left lines in continuation of the left subplot), and best achieved  $\Delta f$  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: 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  $\Delta f$  and Df denote the difference to the optimal function value.