

BBOB: Nelder-Mead with Resize and Halfruns

Benjamin Doerr
Max-Planck-Institut für
Informatik
Campus E1.4
66123 Saarbrücken

Mahmoud Fouz
Max-Planck-Institut für
Informatik
Campus E1.4
66123 Saarbrücken

Martin Schmidt
Universität des Saarlandes
Campus E1.3
66123 Saarbrücken

Magnus Wahlström
Max-Planck-Institut für
Informatik
Campus E1.4
66123 Saarbrücken

ABSTRACT

Using the BBOB template, we investigate how the Nelder-Mead simplex algorithm can be combined with evolutionary ideas to give a competitive hybrid approach to optimize continuous functions. We significantly improve the performance of the algorithm in higher dimension by the addition of a *re-shaping* step of the search, to correct for a known problem in the simplex search behaviour. We also give a reasonably good population-based approach in which only a third of the individuals is fully matured, with a bias towards fitter individuals, via a variant of the Nelder-Mead method.

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, Evolutionary computation

1. INTRODUCTION AND MAIN RESULTS

The Nelder-Mead simplex algorithm (not to be mixed up with Dantzig's simplex algorithm for linear optimization) is a simple, in practice often successful method for finding local optima of continuous functions. This raises the question

if and to what extent it can be combined with advanced randomized search heuristics to yield a good hybrid approach. This is what we try to answer in this paper. Our investigation is greatly aided by the BBO-Benchmarking framework developed by Auger, Fink, Hansen and Ros [1, 2, 3].

The simplest way of using the Nelder-Mead algorithm to search for the global optimum of a continuous function with possibly more than one local optimum is to run it with several initial points chosen uniformly at random from the domain (*random restarts*). In fact, this works surprisingly well as our first experiments show.

We then tried to combine the Nelder-Mead algorithm with different ideas stemming from evolutionary computation. To our surprise, this plan turned out to be very difficult. In the end, we succeeded in obtaining a hybrid approach slightly superior to the simple random restart approach. It builds on the (simple) idea of stopping the Nelder-Mead algorithm prior to convergence and only fully developing solutions that, at this premature stage, look promising. This selection is done by keeping a population of solutions and selecting a fitter fraction for full maturing by continuing the Nelder-Mead algorithm from the previous setting. This simple idea seems to save a number of fitness evaluations. It also seems to reduce the run-times of runs that take much longer than the average, which is a nice feature as well.

2. NELDER-MEAD SIMPLEX METHOD

The Nelder-Mead method, also known as downhill simplex search, is a non-gradient based method for optimizing nonlinear functions [4]. It searches a d -dimensional space by maintaining a *simplex*, i.e., a set of $d + 1$ points not lying on a common hyperplane.

In each iteration of the search, the point which currently has the worst function value is replaced by a new point, which is taken from the line going through the point and the center of the simplex. Specifically, first the point's reflection relative to the center of the other points is evaluated. Based on the quality of this point, either this reflection point is taken, or a point twice as far out, causing the simplex to grow (this happens if the reflected point is the best seen so far and the extra point is even better), or some point closer to the center, causing the simplex to shrink (in case

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.

the function value of the reflection point is poor compared to the other points of the simplex).

Repeating this step causes the simplex to wander around the function space, ideally moving towards a region with a good function value and there converging towards a local optimum point.

The search starts with a simplex constructed from a starting point x and a size parameter δ set to 2. This simplex consists of the point x and for each $i \in \{1, \dots, d\}$ one point x_i with $x_i(i) = x(i) + \delta$, $x_i(j) = x(j)$ for $j \neq i$.

3. RESHAPING

However, for reasons that are not entirely clear, it can sometimes happen that the shape of the simplex becomes degenerately flat, leading to a search process that makes very little progress in the direction towards the optimum. At this point, the simplex can converge to a point that is not a local optimum. This is a known effect in the non-linear optimization literature, and we did observe traces of it in higher dimension (even for seemingly benign functions like the sphere function f_1).

To counteract this effect, we introduce a *reshaping* step to the search. In such a step, we replace the present simplex by a fat simplex, built from the best corner point of the previous one and with the same average size value (i.e., the same average distance from a point to the center of the simplex), but with no other relation to the shape of the old simplex. More precisely, the reshaped simplex has the same shape as our starting simplex (see above), constructed from the best corner point of the previous simplex and with the average size of the previous simplex as its size parameter δ . For simplicity, we simply perform this step every 1000 iterations. We observe this to make a large difference for the higher-dimensional tests.

The base of our implementation is the Nelder-Mead simplex function of the GNU Scientific Library, version 1.12. We use three criteria to terminate a simplex search: (i) 2000 d iterations have been used for the simplex, where d is the dimension; (ii) the size of the simplex (i.e. the average distance of a point to the center of the simplex) is less than 10^{-11} ; or (iii) the function values of the simplex points deviate by less than 10^{-11} .

4. HALFRUNS

We observed surprising difficulty in combining the Nelder-Mead algorithm with classical evolutionary ideas; many typical ideas failed to have any positive impact. However, we did have some success with a population-based approach.

The idea we successfully tried is being stingy with function evaluations. To that aim, we shall not run the Nelder-Mead algorithms until convergence for all solutions. More precisely, when starting the Nelder-Mead algorithm on a newly generated random point, we only conduct what we call a halfrun. That is, we run the Nelder-Mead algorithm until at most 30 d iterations have been spent on this run.

For some of these half-developed solutions (described by the so-far computed simplex) we will later resume the Nelder-Mead algorithm (and run it till convergence as described in Section 2). With the aim of being stingy with function evaluations, we shall try to select the more promising solutions for that (and thus, compared to the simple random restart heuristics, save the function evaluations needed to complete

the Nelder-Mead optimization for the other solutions).

The selection process is organized by keeping a population containing all half and fully developed solutions. Whenever we have at least twice as many half-developed solutions as fully developed ones, we select one half-developed solution for full development. This choice is done in a randomized fashion. With probability 50%, we choose greedily the half-developed solution with the current best value. Here the value of a solution is the function value of the best of its simplex points. With the remaining 50% chance, we choose a half-developed solution uniformly at random. The hope of this approach is that there is a bias towards more promising solutions. However, for functions where our view of being more promising is misleading, the solutions chosen uniformly at random avoid being misled.

The algorithm terminates after 100 fully developed individuals have been produced or after 20,000 d function evaluations have been used, whichever comes first. The total execution time of the experiment (excluding the 40-dimensional tests) was roughly one hour.

Our experiments show that the halfrun system (data given in Fig. 1 and 2 as well as Table 1) does save us some function evaluations (compared to the simple random restart approach (its data is omitted for reasons of space)), but the gain is not consistent for all functions and the difference in the ERT-value is mostly less than 20%. However, for a good proportion of the functions where all runs were successful, we do observe that using halfruns decreases the mFE-value and the 90%-tiles by a significant amount. At seems that using halfruns in particular reduces the variation of the run-times.

5. CONCLUSIONS AND DISCUSSION

We added the two components of reshaping steps and halfruns to the Nelder-Mead simplex search algorithm, and found the algorithm to generally perform well, giving reasonable results even in higher dimension. The reshaping step in particular significantly improved the performance of the algorithm in higher dimension, where an unimproved simplex search would otherwise frequently get stuck. The effects of the halfruns were less drastic on average, but we did find their inclusion to reduce the worst-case running time (e.g., with halfruns the risk of being unlucky and getting a “very bad” run is reduced).

With the variant of the Nelder-Mead algorithms described above working sufficiently well on the benchmark set of functions, we also tried a number of ways to incorporate it into some heuristic framework taking ideas from evolutionary computation. Unfortunately, all attempts, be it simple mutation-like ideas or more complex approaches like simulated annealing, failed to show any significant improvements.

As a partial explanation for this, we present data suggesting that the Nelder-Mead optimization is not very sensitive to the choice of starting point. Consider the outcome of the ill-natured experiment, given in Table 2. In this experiment, for each starting point we choose, we perform eight consecutive restarts from the same point, varying only the δ -value (taken uniformly from 0 to 4). Despite these repetitions, we only observe a small reduction in the efficiency of the procedure. This holds in the setting with halfruns (data for the good run given in Table 1 and the ill-natured version in Table 2) and the setting without halfruns (just random restarts). In both settings, we seldom observe the ill-natured

version to need twice as many function evaluations (as measured by the ERT). Rather, the typical difference is a small 2-digit percentage. For quite a number of values (both with and without reruns), we even see the ill-natured version being faster.

Hence instead of an expected increase of around a factor of eight, we only see a very small negative effect of reusing the same starting point several times. Of course, we do not suggest to do so, but it clearly demonstrates the choice of starting point in the Nelder-Mead algorithm not to be critical.

6. 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] J. A. Nelder and R. Mead. A Simplex Method for Function Minimization. *The Computer Journal*, 7(4):308–313, 1965.

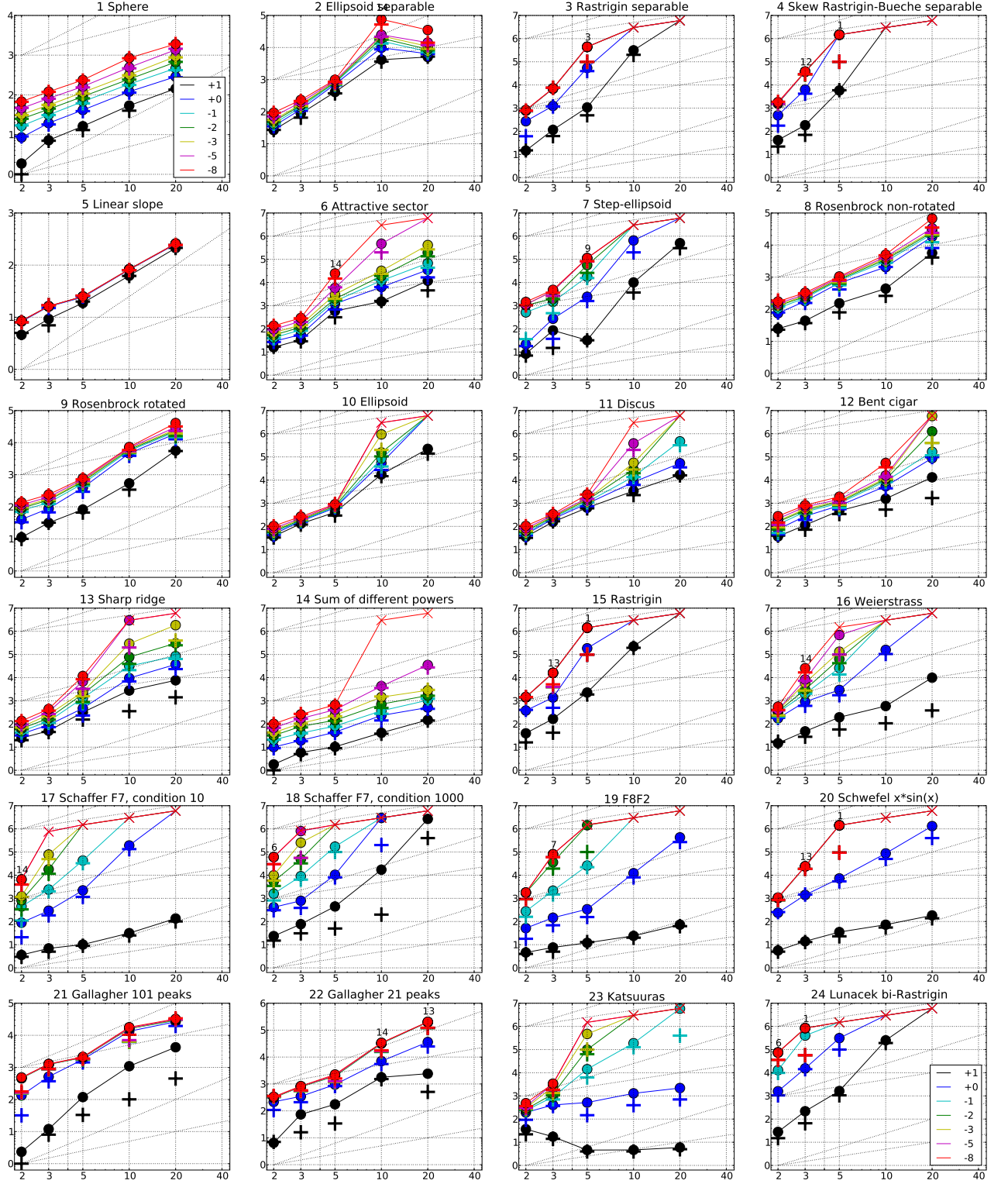


Figure 1: Expected Running Time (ERT, \bullet) to reach $f_{\text{opt}} + \Delta f$ and median number of function evaluations of successful trials (+), shown for $\Delta f = 10, 1, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-5}, 10^{-8}$ (the exponent is given in the legend of f_1 and f_{24}) versus dimension in log-log presentation. The $\text{ERT}(\Delta f)$ 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_{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.

f1 in 5-D, N=15, mFE=373						f1 in 20-D, N=15, mFE=3048						f2 in 5-D, N=15, mFE=2109						f2 in 20-D, N=15, mFE=166164									
Δf	#	ERT	10%	90%	RTsucc	#	ERT	10%	90%	RTsucc	Δf	#	ERT	10%	90%	RTsucc	#	ERT	10%	90%	RTsucc	#	ERT	10%	90%	RTsucc	
10	15	1.6e1	1.3e1	2.0e1	1.6e1	15	1.4e2	1.2e2	1.6e2	1.4e2	10	15	4.1e2	3.3e2	5.0e2	4.1e2	15	5.1e3	4.8e3	5.3e3	5.1e3	15	5.1e3	4.8e3	5.3e3	5.1e3	
1	15	4.2e1	3.7e1	4.6e1	4.2e1	15	2.9e2	2.6e2	3.1e2	2.9e2	1	15	7.1e2	5.9e2	8.4e2	7.1e2	15	6.4e3	5.9e3	6.9e3	6.4e3	15	6.4e3	5.9e3	6.9e3	6.4e3	
1e-1	15	6.8e1	6.3e1	7.3e1	6.8e1	15	4.7e2	4.2e2	5.2e2	4.7e2	1e-1	15	7.9e2	6.6e2	9.3e2	7.9e2	15	7.5e3	6.9e3	8.1e3	7.5e3	15	7.5e3	6.9e3	8.1e3	7.5e3	
1e-3	15	1.2e2	1.1e2	1.2e2	1.2e2	15	9.2e2	8.2e2	1.0e3	9.2e2	1e-3	15	8.6e2	7.3e2	1.0e3	8.6e2	15	1.1e4	9.5e3	1.3e4	1.1e4	15	1.1e4	9.5e3	1.3e4	1.1e4	
1e-5	15	1.6e2	1.5e2	1.7e2	1.6e2	15	1.4e3	1.3e3	1.5e3	1.4e3	1e-5	15	9.2e2	7.7e2	1.1e3	9.2e2	15	1.4e4	1.2e4	1.6e4	1.4e4	15	1.4e4	1.2e4	1.6e4	1.4e4	
1e-8	15	2.4e2	2.3e2	2.5e2	2.4e2	15	1.9e3	1.8e3	2.0e3	1.9e3	1e-8	15	9.9e2	8.5e2	1.1e3	9.9e2	15	3.5e4	2.3e4	4.8e4	3.5e4	15	3.5e4	2.3e4	4.8e4	3.5e4	
f3 in 5-D, N=15, mFE=100012						f3 in 20-D, N=15, mFE=400048						f4 in 5-D, N=15, mFE=100019						f4 in 20-D, N=15, mFE=400048									
Δf	#	ERT	10%	90%	RTsucc	#	ERT	10%	90%	RTsucc	Δf	#	ERT	10%	90%	RTsucc	#	ERT	10%	90%	RTsucc	Δf	#	ERT	10%	90%	RTsucc
10	15	1.1e3	4.5e2	1.7e3	1.1e3	0	47e+0	34e+0	62e+0	2.8e5	10	15	5.8e3	4.5e3	7.0e3	5.8e3	0	11e+1	78e+0	15e+1	1.3e5	10	15	5.8e3	4.5e3	7.0e3	5.8e3
1	13	5.4e4	4.0e4	6.7e4	5.0e4	1	1	1.5e6	1.4e6	1.5e6	1.0e5	1	1	1.5e6	1.4e6	1.5e6	1.0e5
1e-1	3	4.4e5	3.9e5	4.9e5	9.8e4	1e-1	1	1.5e6	1.4e6	1.5e6	1.0e5	1e-1	1	1.5e6	1.4e6	1.5e6	1.0e5
1e-3	3	4.4e5	3.9e5	4.9e5	9.8e4	1e-3	1	1.5e6	1.4e6	1.5e6	1.0e5	1e-3	1	1.5e6	1.4e6	1.5e6	1.0e5
1e-5	3	4.4e5	3.9e5	4.9e5	9.8e4	1e-5	1	1.5e6	1.4e6	1.5e6	1.0e5	1e-5	1	1.5e6	1.4e6	1.5e6	1.0e5
1e-8	3	4.4e5	3.9e5	4.9e5	9.9e4	1e-8	1	1.5e6	1.4e6	1.5e6	1.0e5	1e-8	1	1.5e6	1.4e6	1.5e6	1.0e5
f5 in 5-D, N=15, mFE=42						f5 in 20-D, N=15, mFE=467						f6 in 5-D, N=15, mFE=100001						f6 in 20-D, N=15, mFE=400020									
Δf	#	ERT	10%	90%	RTsucc	#	ERT	10%	90%	RTsucc	Δf	#	ERT	10%	90%	RTsucc	#	ERT	10%	90%	RTsucc	Δf	#	ERT	10%	90%	RTsucc
10	15	1.9e1	1.7e1	2.0e1	1.9e1	15	2.1e2	1.9e2	2.3e2	2.1e2	10	15	5.9e2	4.0e2	7.8e2	5.9e2	15	1.2e4	7.3e3	1.7e4	1.2e4	10	15	5.9e2	4.0e2	7.8e2	5.9e2
1	15	2.4e1	2.2e1	2.6e1	2.4e1	15	2.5e2	2.3e2	2.8e2	2.5e2	1	15	1.2e3	8.3e2	1.6e3	1.2e3	15	3.5e4	2.0e4	5.3e4	3.5e4	1	15	1.2e3	8.3e2	1.6e3	1.2e3
1e-1	15	2.5e1	2.3e1	2.7e1	2.5e1	15	2.6e2	2.4e2	2.8e2	2.6e2	1e-1	15	1.6e3	1.2e3	2.0e3	1.6e3	15	6.8e4	4.8e4	8.9e4	6.8e4	1e-1	15	1.6e3	1.2e3	2.0e3	1.6e3
1e-3	15	2.5e1	2.3e1	2.7e1	2.5e1	15	2.6e2	2.4e2	2.8e2	2.6e2	1e-3	15	3.1e3	2.2e3	3.9e3	3.1e3	10	4.1e5	3.6e5	4.7e5	2.6e5	1e-3	15	3.1e3	2.2e3	3.9e3	3.1e3
1e-5	15	2.5e1	2.3e1	2.7e1	2.5e1	15	2.6e2	2.4e2	2.8e2	2.6e2	1e-5	15	5.8e3	4.5e3	7.1e3	5.8e3	0	46e-5	42e-6	11e-3	2.5e5	1e-5	15	5.8e3	4.5e3	7.1e3	5.8e3
1e-8	15	2.5e1	2.3e1	2.7e1	2.5e1	15	2.6e2	2.3e2	2.8e2	2.6e2	1e-8	14	2.5e4	1.7e4	3.3e4	2.2e4	1e-8	14	2.5e4	1.7e4	3.3e4	2.2e4
f7 in 5-D, N=15, mFE=100017						f7 in 20-D, N=15, mFE=400048						f8 in 5-D, N=15, mFE=1955						f8 in 20-D, N=15, mFE=220692									
Δf	#	ERT	10%	90%	RTsucc	#	ERT	10%	90%	RTsucc	Δf	#	ERT	10%	90%	RTsucc	#	ERT	10%	90%	RTsucc	Δf	#	ERT	10%	90%	RTsucc
10	15	3.2e1	3.0e1	3.5e1	3.2e1	8	5.0e5	4.1e5	5.9e5	2.7e5	10	15	1.5e2	1.2e2	1.9e2	1.5e2	15	5.7e3	3.3e3	8.4e3	5.7e3	10	15	1.5e2	1.2e2	1.9e2	1.5e2
1	15	2.4e3	1.4e3	3.6e3	2.4e3	0	97e-1	53e-1	20e+0	2.2e5	1	15	6.6e2	5.1e2	8.1e2	6.6e2	15	1.8e4	9.6e3	2.9e4	1.8e4	1	15	6.6e2	5.1e2	8.1e2	6.6e2
1e-1	15	1.8e4	1.3e4	2.3e4	1.8e4	1e-1	15	8.0e2	6.7e2	9.6e2	8.0e2	15	2.2e4	1.3e4	3.2e4	2.2e4	1e-1	15	8.0e2	6.7e2	9.6e2	8.0e2
1e-3	9	1.1e5	9.3e4	1.3e5	7.9e4	1e-3	15	9.0e2	7.7e2	1.1e3	9.0e2	15	2.7e4	1.9e4	3.7e4	2.7e4	1e-3	15	9.0e2	7.7e2	1.1e3	9.0e2
1e-5	9	1.1e5	9.3e4	1.3e5	7.9e4	1e-5	15	9.7e2	8.2e2	1.1e3	9.7e2	15	3.3e4	2.5e4	4.3e4	3.3e4	1e-5	15	9.7e2	8.2e2	1.1e3	9.7e2
1e-8	9	1.1e5	9.5e4	1.3e5	7.9e4	1e-8	15	1.1e3	9.1e2	1.2e3	1.1e3	15	6.7e4	4.8e4	9.0e4	6.7e4	1e-8	15	1.1e3	9.1e2	1.2e3	1.1e3
f9 in 5-D, N=15, mFE=1462						f9 in 20-D, N=15, mFE=91153						f10 in 5-D, N=15, mFE=1345						f10 in 20-D, N=15, mFE=400002									
Δf	#	ERT	10%	90%	RTsucc	#	ERT	10%	90%	RTsucc	Δf	#	ERT	10%	90%	RTsucc	#	ERT	10%	90%	RTsucc	Δf	#	ERT	10%	90%	RTsucc
10	15	8.2e1	6.5e1	1.0e2	8.2e1	15	5.7e3	5.3e3	6.0e3	5.7e3	10	15	4.0e2	3.4e2	4.7e2	4.0e2	12	2.2e5	1.7e5	2.8e5	1.7e5	10	15	4.0e2	3.4e2	4.7e2	4.0e2
1	15	3.9e2	3.2e2	4.8e2	3.9e2	15	1.9e4	1.4e4	2.5e4	1.9e4	1	15	6.3e2	5.6e2	6.9e2	6.3e2	0	57e-1	26e-1	17e+0	2.5e5	1	15	6.3e2	5.6e2	6.9e2	6.3e2
1e-1	15	5.3e2	4.5e2	6.0e2	5.3e2	15	2.2e4	1.6e4	2.8e4	2.2e4	1e-1	15	6.7e2	6.0e2	7.4e2	6.7e2	1e-1	15	6.7e2	6.0e2	7.4e2	6.7e2
1e-3	15	6.4e2	5.7e2	7.2e2	6.4e2	15	2.6e4	2.1e4	3.2e4	2.6e4	1e-3	15	7.8e2	7.1e2	8.5e2	7.8e2	1e-3	15	7.8e2	7.1e2	8.5e2	7.8e2
1e-5	15	7.0e2	6.2e2	7.7e2	7.0e2	15	3.0e4	2.5e4	3.6e4	3.0e4	1e-5	15	8.3e2	7.6e2	9.0e2	8.3e2	1e-5	15	8.3e2	7.6e2	9.0e2	8.3e2
1e-8	15	7.7e2	7.0e2	8.5e2	7.7e2	15	4.1e4	3.4e4	4.8e4	4.1e4	1e-8	15	9.0e2	8.4e2	9.7e2	9.0e2	1e-8	15	9.0e2	8.4e2	9.7e2	9.0e2
f11 in 5-D, N=15, mFE=5209						f11 in 20-D, N=15, mFE=400002						f12 in 5-D, N=15, mFE=3937						f12 in 20-D, N=15, mFE=400049									
Δf	#	ERT	10%	90%	RTsucc	#	ERT	10%	90%	RTsucc	Δf	#	ERT	10%	90%	RTsucc	#	ERT	10%	90%	RTsucc	Δf	#	ERT	10%	90%	RTsucc
10	15	6.4e2	5.0e2	7.9e2	6.4e2	15	1.7e4	1.5e4	1.9e4	1.7e4	10	15	4.7e2	3.3e2	6.3e2	4.7e2	15	1.3e4	7.1e3	2.0e4	1.3e4	10	15	4.7e2	3.3e2	6.3e2	4.7e2
1	15	9.6e2	8.0e2	1.1e3	9.6e2	15	5.3e4	4.2e4	6.4e4	5.3e4	1	15	7.8e2	5.4e2	1.0e3	7.8e2	15	5.8e4	7.3e4	1.0e5	8.8e4	1	15	7.8e2	5.4e2	1.0e3	7.8e2
1e-1	15	1.1e3	9.8e2	1.3e3	1.1e3	9	4.7e5	4.0e5	5.3e5	2.8e5	1e-1	15	9.9e2	7.5e2	1.2e3	9.9e2	14	1.7e5	1.3e5	2.1e5	1.4e5	1e-1	15	9.9e2	7.5e2	1.2e3	9.9e2
1e-3	15	1.4e3	1.3e3	1.6e3	1.4e3	0	75e-3	34e-3	53e-2	2.5e5	1e-3	15	1.3e3	1.0e3	1.5e3	1.3e3	1	5.8e6	5.5e6	6.0e6	4.0e5	1e-3	15	1.3e3	1.0e3	1.5e3	1.3e3
1e-5	15	1.6e3	1.5e3	1.8e3	1.6e3	1e-5	15	1.5e3	1.1e3	1.8e3	1.5e3	0	21e-3	29e-4	77e-3	1.6e5	1e-5	15	1.5e3	1.1e3	1.8e3	1.5e3
1e-8	15	2.4e3	2.0e3	2.8e3	2.4e3	1e-8	15	1.9e3	1.6e3	2.3e3	1.9e3	1e-8	15	1.9e3	1.6e3	2.3e3	1.9e3
f13 in 5-D, N=15, mFE=47566						f13 in 20-D, N=15, mFE=400024						f14 in 5-D, N=15, mFE=1105						f14 in 20-D, N=15, mFE=400002									
Δf	#	ERT	10%	90%	RTsucc	#	ERT	10%	90%	RTsucc	Δf	#	ERT	10%	90%	RTsucc	#	ERT	10%	90%	RTsucc	Δf	#	ERT	10%	90%	RTsucc
10	15	3.2e2	1.9e2	4.6e2	3.2e2	15	7.7e3	3.6e3	1.2e4	7.7e3	10	15	1.0e1	8.2e0	1.2e1	1.0e1	15	1.5e2	1.3e2	1.7e2	1.5e2	10	15	1.0e1	8.2e0	1.2e1	1.0e1
1	15	4.7e2	3.2e2	6.4e2	4.7e2	15	3.7e4	2.2e4																			

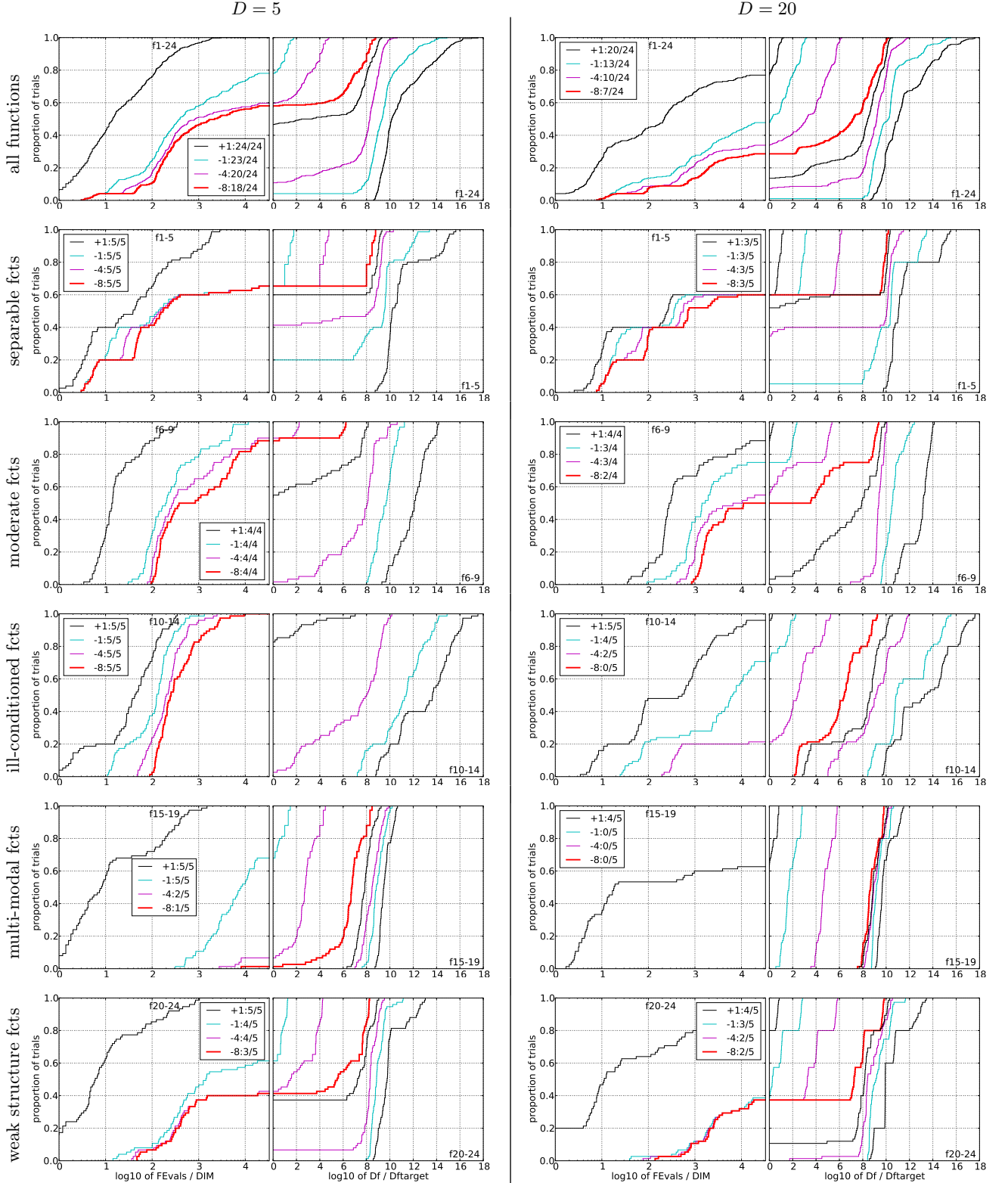


Figure 2: Empirical cumulative distribution functions (ECDFs), plotting the fraction of trials versus running time (left) or Δf . Left subplots: ECDF of the running time (number of function evaluations), divided by search space dimension D , to fall below $f_{\text{opt}} + \Delta f$ with $\Delta f = 10^k$, 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). Top row: all results from all functions; second row: separable functions; third row: 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.

f1 in 5-D, N=15, mFE=371						f1 in 20-D, N=15, mFE=4948						f2 in 5-D, N=15, mFE=1459						f2 in 20-D, N=15, mFE=401954					
Δf	#	ERT	10%	90%	RTsucc	#	ERT	10%	90%	RTsucc	Δf	#	ERT	10%	90%	RTsucc	#	ERT	10%	90%	RTsucc		
	10	15	1.5e1	1.3e1	1.8e1	1.5e1	15	1.2e2	1.1e2	1.3e2	1.2e2	10	15	2.3e2	2.0e2	2.5e2	2.3e2	15	4.6e3	3.1e3	6.3e3	4.6e3	
	1	15	3.9e1	3.6e1	4.2e1	3.9e1	15	3.3e2	2.9e2	3.6e2	3.3e2	1	15	3.5e2	3.0e2	3.9e2	3.5e2	15	1.2e4	8.0e3	1.7e4	1.2e4	
1e-1	15	6.4e1	6.0e1	6.8e1	6.4e1	15	5.9e2	5.4e2	6.5e2	5.9e2	1e-1	15	4.4e2	4.0e2	4.8e2	4.4e2	15	1.9e4	1.3e4	2.5e4	1.9e4		
1e-3	15	1.1e2	1.1e2	1.2e2	1.1e2	15	1.2e3	1.1e3	1.3e3	1.2e3	1e-3	15	5.7e2	5.4e2	6.1e2	5.7e2	14	7.9e4	4.2e4	1.2e5	7.8e4		
1e-5	15	1.6e2	1.5e2	1.7e2	1.6e2	15	1.8e3	1.7e3	2.0e3	1.8e3	1e-5	15	6.9e2	6.6e2	7.2e2	6.9e2	14	1.0e5	6.2e4	1.5e5	1.0e5		
1e-8	15	2.3e2	2.3e2	2.4e2	2.3e2	15	2.8e3	2.7e3	3.0e3	2.8e3	1e-8	15	9.5e2	8.9e2	1.0e3	9.5e2	14	1.1e5	6.8e4	1.5e5	1.1e5		
f3 in 5-D, N=15, mFE=43304						f3 in 20-D, N=15, mFE=400634						f4 in 5-D, N=15, mFE=45211						f4 in 20-D, N=15, mFE=400414					
Δf	#	ERT	10%	90%	RTsucc	#	ERT	10%	90%	RTsucc	Δf	#	ERT	10%	90%	RTsucc	#	ERT	10%	90%	RTsucc		
10	15	3.6e2	2.5e2	4.8e2	3.6e2	0	50e+0	37e+0	57e+0	5.6e4	10	15	3.9e3	2.9e3	4.9e3	3.9e3	0	11e+1	72e+0	14e+1	3.2e4		
1	7	6.2e4	5.1e4	7.3e4	2.7e4	1	1	6.4e5	6.2e5	6.5e5	4.5e4		
1e-1	1	6.1e5	5.8e5	6.3e5	4.2e4	1e-1	0	40e-1	30e-1	70e-1	1.4e4		
1e-3	1	6.1e5	5.8e5	6.3e5	4.2e4	1e-3		
1e-5	1	6.1e5	5.8e5	6.3e5	4.2e4	1e-5		
1e-8	1	6.1e5	5.8e5	6.3e5	4.2e4	1e-8		
f5 in 5-D, N=15, mFE=45						f5 in 20-D, N=15, mFE=389						f6 in 5-D, N=15, mFE=1081						f6 in 20-D, N=15, mFE=401955					
Δf	#	ERT	10%	90%	RTsucc	#	ERT	10%	90%	RTsucc	Δf	#	ERT	10%	90%	RTsucc	#	ERT	10%	90%	RTsucc		
10	15	1.8e1	1.6e1	2.0e1	1.8e1	15	2.1e2	1.9e2	2.3e2	2.1e2	10	15	1.3e2	1.1e2	1.5e2	1.3e2	15	4.0e4	1.0e4	7.3e4	4.0e4		
1	15	2.3e1	2.1e1	2.5e1	2.3e1	15	2.5e2	2.3e2	2.7e2	2.5e2	1	15	1.9e2	1.7e2	2.2e2	1.9e2	15	1.1e5	7.1e4	1.4e5	1.1e5		
1e-1	15	2.3e1	2.1e1	2.5e1	2.3e1	15	2.5e2	2.4e2	2.7e2	2.5e2	1e-1	15	2.7e2	2.4e2	3.0e2	2.7e2	11	3.0e5	2.3e5	3.6e5	2.2e5		
1e-3	15	2.3e1	2.1e1	2.5e1	2.3e1	15	2.5e2	2.3e2	2.7e2	2.5e2	1e-3	15	4.2e2	4.0e2	4.4e2	4.2e2	6	7.4e5	6.2e5	8.4e5	2.5e5		
1e-5	15	2.3e1	2.1e1	2.5e1	2.3e1	15	2.5e2	2.4e2	2.7e2	2.5e2	1e-5	15	5.7e2	5.4e2	6.0e2	5.7e2	4	1.3e6	1.2e6	1.4e6	3.0e5		
1e-8	15	2.3e1	2.1e1	2.5e1	2.3e1	15	2.5e2	2.4e2	2.8e2	2.5e2	1e-8	15	7.7e2	7.3e2	8.1e2	7.7e2	0	73e-4	17e-7	32e-2	2.8e5		
f7 in 5-D, N=15, mFE=39387						f7 in 20-D, N=15, mFE=401513						f8 in 5-D, N=15, mFE=13021						f8 in 20-D, N=15, mFE=401934					
Δf	#	ERT	10%	90%	RTsucc	#	ERT	10%	90%	RTsucc	Δf	#	ERT	10%	90%	RTsucc	#	ERT	10%	90%	RTsucc		
10	15	6.6e1	4.9e1	8.3e1	6.6e1	9	3.6e5	2.7e5	4.4e5	2.1e5	10	15	1.0e2	8.4e1	1.2e2	1.0e2	15	4.4e3	2.2e3	6.9e3	4.4e3		
1	15	8.5e2	6.2e2	1.1e3	8.5e2	0	94e-1	68e-1	16e+0	1.4e5	1	15	1.7e3	9.3e2	2.6e3	1.7e3	15	9.3e4	6.3e4	1.2e5	9.3e4		
1e-1	15	3.7e3	2.3e3	5.2e3	3.7e3	1e-1	15	2.4e3	1.6e3	3.3e3	2.4e3	14	1.5e5	1.0e5	2.1e5	1.5e5		
1e-3	5	4.6e4	3.6e4	5.5e4	1.0e4	1e-3	15	3.4e3	2.5e3	4.3e3	3.4e3	4	1.2e6	1.0e6	1.4e6	2.8e5		
1e-5	5	4.6e4	3.6e4	5.5e4	1.0e4	1e-5	15	4.3e3	3.4e3	5.3e3	4.3e3	0	56e-4	16e-5	75e-3	2.2e5		
1e-8	5	4.6e4	3.6e4	5.5e4	1.0e4	1e-8	15	6.2e3	5.2e3	7.3e3	6.2e3		
f9 in 5-D, N=15, mFE=6895						f9 in 20-D, N=15, mFE=401955						f10 in 5-D, N=15, mFE=100566						f10 in 20-D, N=15, mFE=401966					
Δf	#	ERT	10%	90%	RTsucc	#	ERT	10%	90%	RTsucc	Δf	#	ERT	10%	90%	RTsucc	#	ERT	10%	90%	RTsucc		
10	15	8.8e1	7.6e1	1.0e2	8.8e1	15	8.6e3	8.0e3	9.2e3	8.6e3	10	15	5.4e3	3.3e3	7.7e3	5.4e3	0	53e+0	21e+0	17e+1	2.2e5		
1	15	7.6e2	4.5e2	1.1e3	7.6e2	15	5.1e4	3.0e4	7.2e4	5.1e4	1	13	4.1e4	2.9e4	5.5e4	3.5e4		
1e-1	15	9.8e2	6.7e2	1.3e3	9.8e2	12	1.7e5	1.1e5	2.3e5	1.4e5	1e-1	9	1.1e5	9.5e4	1.3e5	6.3e4		
1e-3	15	1.6e3	1.2e3	1.9e3	1.6e3	0	25e-3	41e-4	30e-2	1.6e5	1e-3	1	1.5e6	1.5e6	1.5e6	1.0e5		
1e-5	15	2.0e3	1.7e3	2.4e3	2.0e3	1e-5	0	44e-3	19e-4	20e-1	6.3e4		
1e-8	15	2.6e3	2.3e3	3.1e3	2.6e3	1e-8		
f11 in 5-D, N=15, mFE=100570						f11 in 20-D, N=15, mFE=401955						f12 in 5-D, N=15, mFE=100571						f12 in 20-D, N=15, mFE=401954					
Δf	#	ERT	10%	90%	RTsucc	#	ERT	10%	90%	RTsucc	Δf	#	ERT	10%	90%	RTsucc	#	ERT	10%	90%	RTsucc		
10	15	3.0e3	2.1e3	3.9e3	3.0e3	0	52e+0	33e+0	77e+0	1.8e5	10	14	2.6e4	1.6e4	3.7e4	2.5e4	15	4.7e4	2.8e4	6.6e4	4.7e4		
1	13	3.5e4	2.3e4	4.7e4	3.2e4	1	10	8.7e4	6.9e4	1.1e5	5.4e4	13	2.0e5	1.5e5	2.5e5	2.0e5		
1e-1	5	2.5e5	2.2e5	2.8e5	9.5e4	1e-1	4	3.2e5	2.9e5	3.5e5	7.8e4	14	1.3e6	1.1e6	1.4e6	4.0e5		
1e-3	0	22e-2	73e-4	10e-1	5.0e4	1e-3	0	22e-2	11e-3	65e-1	7.1e4	1	5.9e6	5.7e6	6.0e6	4.0e5		
1e-5	1e-5	0	32e-2	54e-3	18e-1	2.2e5		
1e-8	1e-8		
f13 in 5-D, N=15, mFE=77706						f13 in 20-D, N=15, mFE=401282						f14 in 5-D, N=15, mFE=100566						f14 in 20-D, N=15, mFE=401960					
Δf	#	ERT	10%	90%	RTsucc	#	ERT	10%	90%	RTsucc	Δf	#	ERT	10%	90%	RTsucc	#	ERT	10%	90%	RTsucc		
10	15	4.9e2	3.2e2	6.7e2	4.9e2	15	5.6e3	3.6e3	7.7e3	5.6e3	10	15	7.9e0	6.0e0	9.6e0	7.9e0	15	1.3e2	1.2e2	1.5e2	1.3e2		
1	15	5.0e3	3.2e3	6.8e3	5.0e3	15	3.7e4	2.6e4	4.9e4	3.7e4	1	15	4.6e1	4.0e1	5.3e1	4.6e1	15	4.5e2	3.8e2	5.2e2	4.5e2		
1e-1	15	1.0e4	7.3e3	1.4e4	1.0e4	14	1.8e5	1.4e5	2.3e5	1.7e5	1e-1	15	8.4e1	7.7e1	8.9e1	8.4e1	15	8.3e2	7.2e2	9.5e2	8.3e2		
1e-3	4	2.1e5	1.9e5	2.4e5	5.7e4	3	1.8e6	1.7e6	2.0e6	4.0e5	1e-3	15	2.2e2	2.0e2	2.5e2	2.2e2	15	4.4e3	3.9e3	4.8e3	4.4e3		
1e-5	0	30e-4	10e-5	14e-3	2.5e4	0	34e-3	31e-5	96e-3	1.6e5	1e-5	15	4.9e3	3.5e3	6.4e3	4.9e3	0	40e-6	28e-6	60e-6	2.2e5		
1e-8	1e-8	0	13e-7	50e-8	43e-7	5.0e4		
f15 in 5-D, N=15, mFE=47860						f15 in 20-D, N=15, mFE=401304						f16 in 5-D, N=15, mFE=84467						f16 in 20-D, N=15, mFE=401745					
Δf	#	ERT	10%	90%	RTsucc	#	ERT	10%	90%	RTsucc	Δf	#	ERT	10%	90%	RTsucc	#	ERT	10%	90%	RTsucc		
10	15	1.2e3	8.2e2	1.7e3	1.2e3	0	47e+0	34e+0	63e+0	4.5e4	10	15	2.2e2	6.7e1	3.6e2	2.2e2	15	1.7e4	1.0e4	2.3e4	1.7e4		
1	4	1.5e5	1.4e5	1.7e5	3.5e4	1	15	5.2e3	3.7e3	7.1e3	5.2e3	0	38e-1	24e-1	52e-1	2.5e5		
1e-1	0	20e-1	99e-2	60e-1	2.0e4	1e-1	13	3.1e4	2.0e4	4.1e4	3.0e4		
1e-3	1e-3	0	20e-3	22e-4	14e-2	2.5e4		
1e-5	1e-5		
1e-8	1e-8		
f17 in 5-D, N=15, mFE=100223						f17 in 20-D, N=15, mFE=401954						f18 in 5-D, N=15, mFE=100475						f18 in 20-D, N=15, mFE=401967					
Δf	#	ERT	10%	90%	RTsucc	#	ERT	10%	90%	RTsucc	Δf	#	ERT	10%	90%	RTsucc	#	ERT	10%	90%	RTsucc		
10	15	8.5e0	6.6e0	1.0e1	8.5e0	15	9.6e1	7.8e1	1.2e2	9.6e1	10	15	1.7e3	2.4e2	3.1e3	1.7e3	9	4.4e5	3.7e5	5.2e5	2.4e5		
1	15	1.6e3	1.1e3	2.1e3	1.6e3	0	25e-1	12e-1	51e-1	2.2e5	1	14	2.7e4	1.6e4	3.8e4								