
Black-box optimization benchmarking of the GLOBAL method

László Pál pallaszlo@sapientia.siculorum.ro
Faculty of Business and Humanities, Sapientia – Hungarian University of
Transylvania, Miercurea-Ciuc, Romania

Tibor Csendes csendes@inf.u-szeged.hu
Institute of Informatics, University of Szeged, Hungary

Mihály Csaba Markót Mihaly.Markot@univie.ac.at
Faculty of Mathematics, University of Wien, Austria

Arnold Neumaier Arnold.Neumaier@univie.ac.at
Faculty of Mathematics, University of Wien, Austria

Abstract

GLOBAL is a multistart type stochastic method for bound constrained global optimization problems. Its goal is to find the best local minima that are potentially global. For this reason it involves a combination of sampling, clustering, and local search. The role of clustering is to reduce the number of local searches by forming groups of points around the local minimizers from a uniform sampled domain and to start few local searches in each of those groups. We evaluate the performance of the GLOBAL algorithm on the BBOB-2009 noiseless testbed, containing problems which reflect the typical difficulties arising in real-world applications. The results show that up to a small function evaluation budget, GLOBAL performs well. We improved the parametrization of it and compared the performance with the MATLAB R2010a GlobalSearch algorithm on the BBOB-2010 noiseless testbed between dimensions 2 and 20. According to the results the studied methods perform similar.

Keywords

Global optimization, stochastic search, clustering, multistart method, benchmarking.

1 Introduction

In this paper the global optimization problem is considered subject to variable bound constraints:

$$\min_{x \in X} f(x), \quad X \subset \mathbb{R}^n, \quad (1)$$

where $f(x)$ is the objective function and X is the feasible domain, defined by bounds on the variables. In general we assume that the objective function is twice continuously differentiable, although it is not necessary for the global optimization framework procedure, and with a proper local search algorithm also nondifferentiable problems can be solved.

Several stochastic strategies have been developed recently in the past in order to solve problem (1). Usually they consist of two phases: the global one and the local phase. During the global phase, random points are drawn from the domain of searches X according to a certain, often uniform, distribution. Then, the objective function is

evaluated in these points. During the local phase the sample points are manipulated by means of local search to yield a candidate global minimum. We assume that a proper local optimization method *Loc* is available. It can be started from an arbitrary point $x_0 \in X$ and then this algorithm generates the sequence of points in X which converges to some $x^* := Loc(x_0) \in X$, that is the local minimizer related to the starting point x_0 .

These methods are also called *Multistart* techniques, because they apply local searches to each point in a random sample drawn from the feasible region (Boender et al., 1982; Rinnooy Kan and Timmer, 1987a,b). However, the Multistart method is inefficient when it performs local searches starting from all sample points. That is, in such cases some local minimizer points will be found several times. Since local search is the most time consuming part of the method, it should ideally be invoked no more than once in every region of attraction.

Various improvements were proposed by diverse authors in order to reduce the number of local searches, see e.g. (Törn, 1978; Rinnooy Kan and Timmer, 1987a; Guss et al., 1995). The two most important methods which are aimed at reducing the number of performed local searches are: the *clustering* methods and the *Multi Level Single Linkage (MLSL)* algorithms.

The basic idea behind clustering methods is to form groups (clusters) around the local minimizers from a uniform sampled domain and to start as low number of local searches as possible in each of those groups. In other words, the procedure tries to identify the *regions of attraction* of the given function.

MLSL methods have been derived from clustering methods (Rinnooy Kan and Timmer, 1987b). In this algorithm the local search procedure is applied to every sample point, except if there is another sample point within some critical distance which has a lower function value.

Random Linkage (RL) multistart algorithms introduced by Locatelli and Schoen (Locatelli and Schoen, 1999) retain the good convergence properties of MLSL. Uniformly distributed points are generated one by one, and *Loc* is started from each point with a probability given by a nondecreasing function $\phi(d)$, where d is the distance from the current sample point to the closest of the previous sample points with a better function value.

The multistart clustering global optimization method called GLOBAL (Csendes, 1988) has been introduced in the 80s for bound constrained global optimization problems with black-box type objective functions. The algorithm is based on Boender's algorithm (Boender et al., 1982), and its goal is to find the best local minimizer points that are potentially global. The local search procedure used by GLOBAL was originally either a quasi-Newton procedure with the DFP update formula or a random walk type direct search method called UNIRANDI (for details see (Järvi, 1973)). GLOBAL was originally coded in the Fortran and C languages.

Based on the old GLOBAL method, we introduced a new version (Csendes et al., 2008) coded in MATLAB. The algorithm was carefully analyzed and it was modified in some places to achieve better reliability and efficiency while allowing higher dimensional problems to be solved. In the new version we use the quasi-Newton local search method with the BFGS update instead of the earlier DFP. All three versions (Fortran, C, MATLAB) of the algorithm are freely available for academic and nonprofit purposes at

www.inf.u-szeged.hu/~csendes/regist.php

(after registration and limited for low dimensional problems).

The aim of the present work is to benchmark the GLOBAL algorithm and compare the performance with the MATLAB GlobalSearch solver on a testbed which reflects

the typical difficulties arising in real-world applications. The remainder of the paper is organized as follows: The GLOBAL method is presented in Section 2, while the test environment in Section 3. The benchmarking on the BBOB-2009 noiseless testbed (Finck et al., 2009a; Hansen et al., 2009b) is done in Section 4. During this section, we also describe the parameters and its settings in the test. The CPU timing experiment is presented in Section 4.2, while the discussion of the results are done in Section 4.3.

In Section 5, we compare GLOBAL with the MATLAB GlobalSearch method on the BBOB-2010 noiseless testbed. The new parameter settings of the GLOBAL method are presented in Section 5.1, while the GlobalSearch method overview in Section 5.2. Section 5.3 contains discussion of the comparison results.

2 The GLOBAL algorithm presentation

The GLOBAL method has two phases: a global and a local one. The global phase consists of sampling and clustering, while the local phase is based on local searches. The local minimizer points are found by means of a local search procedure, starting from appropriately chosen points from the sample drawn uniformly within the set of feasibility. In an effort to identify the region of attraction of a local minimizer, the procedure invokes a clustering procedure. The role of clustering is to reduce the number of local searches by forming groups of points around the local minimizers from a uniform sampled domain and start local searches as few times as possible in each of those groups.

GLOBAL uses the Single Linkage clustering rule (Boender et al., 1982; Rinnooy Kan and Timmer, 1987a), which is constructed in such a way that the probability that a local method will not be applied to a point that would lead to an undiscovered local minimizer diminishes to zero when the size of the sample grows. The main steps of GLOBAL are summarized in Algorithm 1.

In line 2, the X^* and $X^{(1)}$ sets are initialized, where X^* is a set containing the local minimizer points that were found so far, while $X^{(1)}$ is a set containing sample points to which the local search procedure has been applied unsuccessfully in the sense that already know local minimizer was found again. The algorithm contains a main iteration loop and the steps from line 3 to line 20 will be repeated until some global stopping rule is satisfied. In line 5, N points are drawn uniformly on X . In line 6, a reduced sample is constructed by taking those γkN points of the current sample that have the lowest function values. A clustering procedure is applied then to the transformed sample. The elements of X^* are first chosen as seed points, followed by the elements of $X^{(1)}$.

Between lines 8 and 18 we iterate over the unclustered points from the reduced sample and apply a local search procedure to them to find a local minimizer point x^* . If x^* is a new local minimizer point, then we add it to X^* (line 12) and choose it as the next seed point (line 13), otherwise we add $x^{(1)}$ to $X^{(1)}$ (line 15) and use it as the next seed point for local search (line 16). In line 18, we add all unclustered reduced sample points which are within a critical distance from the cluster initiated by the seed point x_s . The critical distance r_k will be chosen to depend on kN only so as to minimize the probabilities of two possible failures of the method: the probability that a local search is started, although the related minimum is known already, and the probability that no local search is started in a level set which contains reduced sample points. In line 21, the smallest local minimum value is returned.

One of the questions in applying a stochastic method is when to stop it. Several approaches based on different assumptions about the properties of possible objective functions f and using some stochastic techniques have been proposed to design a proper stopping rule.

Algorithm 1: The GLOBAL algorithm

```

1. function GLOBAL( $f, X$ )
2.    $k := 0; X^* := \emptyset; X^{(1)} := \emptyset$ 
3.   repeat
4.      $k := k + 1$ 
5.     Generate  $N$  points  $x_{(k-1)N+1}, \dots, x_{kN}$  with uniform distribution on  $X$ 
6.     Determine the reduced sample consisting of the  $\gamma k N$  best points from
       the sample  $x_1, \dots, x_{kN}$ 
7.     Clustering to  $X^*$  and  $X^{(1)}$ 
8.     while Not all points from the reduced sample have been assigned to a cluster
       do
9.       let  $x^{(1)}$  be an unclustered point from the reduced sample with the
       smallest objective value
10.       $x^* := Loc(x^{(1)})$ 
11.      if  $x^* \notin X^*$  then
12.         $X^* := X^* \cup \{x^*\}$ 
13.        choose  $x_s := x^*$  as the next seed point
14.      else
15.         $X^{(1)} := X^{(1)} \cup \{x^{(1)}\}$ 
16.        choose  $x_s := x^{(1)}$  as the next seed point
17.      end
18.      Add all unclustered reduced sample points which are within
       distance  $r_k$  of a point already in the cluster initiated by the seed
       point  $x_s$ 
19.    end
20.  until Some global stopping rule is satisfied
21.  return The smallest local minimum value found

```

A Bayesian stopping rule for the Multistart algorithm has been introduced by (Zieliński, 1981) and further developed by (Boender and Zieliński, 1982; Boender and Rinnooy Kan, 1987, 1991; Betrò and Schoen, 1992) and others.

Most Bayesian stopping rules for multistart techniques are based on the collected knowledge about the size of the sample and the number of local minimizers detected. In our GLOBAL algorithm we stop the search when in the actual iteration step it does not find any new local minimizer point.

3 The test environment description

In this paper, the numerical experiments are conducted on a testbed comprising twenty-four noiseless test functions (Finck et al., 2009a; Hansen et al., 2009b). These functions have been constructed so that they reflect the real-world application difficulties and are split into several groups like separable functions, functions with low or moderate conditioning, functions with bad conditioning and unimodal, multi-modal with adequate global structure, multi-modal with weak global structure. All functions are scalable with the dimension, hence we can use 2, 3, 5, 10, 20, and 40 dimensional search space, respectively. Additionally, all functions have their optimum in the $[-5; 5]^n$ range, which can be a reasonable setting for the search domain. Each function is tested

over five different instances. The experiments will be repeated three times for each instance. The algorithm performance is evaluated over all 15 trials.

The success criterion of a run is to reach the $f_{target} = f_{opt} + \Delta f$ target value, where f_{opt} is the optimal function value, and Δf is the precision to reach.

In order to quantify the search cost of an algorithm, a performance measure should be provided. The main performance measure adopted in this paper (Hansen et al., 2009a; Price, 1997) is the runtime ERT, Expected Running Time. The ERT number depends on a given target function value, $f_t = f_{opt} + \Delta f$, and is computed over all relevant trials as the number of function evaluations used during the trials while the best function value did not reach f_t , summed over all trials and divided by the number of trials that actually reached f_t . In other words, the ERT value estimates the expected number of function evaluations for the target function value to be reached if the algorithm is restarted until a single success can be reached.

The results are also presented using the empirical cumulative distribution function of the distribution of ERT divided by n to reach a given target function value. It shows the empirical cumulated probability of success on the problems considered depending on the allocated budget.

The statistical significance is tested with the rank-sum test for a given target Δf_t using, for each trial, either the number of needed function evaluations to reach Δf_t (inverted and multiplied by -1), or, if the target was not reached, the best Δf -value achieved, measured only up to the smallest number of overall function evaluations for any unsuccessful trial under consideration.

For a more detailed environment and experimental description see (Hansen et al., 2009a, 2010a).

4 Benchmarking GLOBAL on the BBOB-2009 noiseless testbed

4.1 Parameter tuning and setup

GLOBAL has six parameters to set: the number of sample points set within an iteration step, the number of best points selected for the transformed sample, the stopping criterion parameter for the local search, the maximum number of function evaluations allowed for local search, the maximum number of local minima to be applied, and the type of used local method. All these parameters have a default value and usually it is enough to change only the first three of them.

In all dimensions and for all functions we used 300 sample points, and the 2 best points were kept for the transformed sample. In 2, 3, and 5 dimensions we used the Nelder-Mead simplex method (Nelder and Mead, 1965) implemented by (Kelley, 1999) as a local search with 10^{-8} as tolerance value and with 5,000 as the maximum number of function evaluations. In 10 and 20 dimensions with the $f_3, f_4, f_7, f_{16}, f_{23}$ functions we used the previous settings with a local search tolerance of 10^{-9} . Finally, in the case of the remained functions we used the previous parameters with the MATLAB `fminunc` function as local search method using the BFGS update formula and with 10,000 as the maximum number of function evaluations.

As it can be observed, during the parameter tuning we used two different settings. In lower dimensions we used the Nelder-Mead method while in higher dimensions the BFGS local search was applied except five functions. Although this kind of a priori parameter settings are not suggested, the two important parameters of GLOBAL (the number of sample points, the number of best points selected) were the same on the entire testbed. The different settings may be characterized with the entropy measure crafting effort (Price, 1997; Hoos and Stützle, 1998) for each dimensionality in the fol-

lowing way:

$$\text{CrE} = - \sum_{k=1}^K \frac{n_k}{n} \ln \left(\frac{n_k}{n} \right)$$

where $n = \sum_{k=1}^K n_k$ is the number of functions in the testbed and n_k is the number of functions, where the parameter setting with index k was used for $k = 1, \dots, K$, K is the number of different parameter settings.

The crafting effort calculated in case of our settings is: $\text{CrE}_{10} = \text{CrE}_{20} = -\left(\frac{5}{24} \ln \frac{5}{24} + \frac{19}{24} \ln \frac{19}{24}\right) = 0.5117$.

4.2 CPU timing experiment

For the timing experiment the GLOBAL algorithm was run on the test function f_8 , and restarted until at least 30 seconds had passed (according to Figure 2 in (Hansen et al., 2009a)). These experiments have been conducted with an Intel Core 2 Duo 2.00 GHz processor computer under Windows XP using the MATLAB 7.6.0.324 version. We have completed two experiments using the BFGS and the simplex local search methods. The other algorithm parameters were the same. In the first case (BFGS) the results were $(2.8, 2.9, 3.0, 3.0, 3.2, 3.2) \cdot 10^{-4}$ seconds, while in the second case (Nelder-Mead simplex) they were $(2.6, 2.9, 3.4, 4.6, 7.5, 21.0) \cdot 10^{-4}$ seconds per function evaluation in dimensions 2, 3, 5, 10, 20, and 40, respectively. The CPU time of a function evaluation of the BFGS search grows sub-linearly with the dimension. For the Nelder-Mead simplex method, the CPU time increases with the dimension linearly up to 20 dimensional problems, while for 40 dimensional functions a rapid increase can be observed.

4.3 Results and discussion

The GLOBAL method have been tested in a black-box scenario on 24 noiseless benchmark functions. Results from experiments according to (Hansen et al., 2009a) on the benchmark functions given in (Finck et al., 2009a; Hansen et al., 2009b) are presented in the Figure 1 and Tables 2 and 3.

For low search space dimensions the algorithm shows good results on many functions. The number of solved functions amounts to 18, 16, 11, 8, 5 out of 24 functions for dimensions 2, 3, 5, 10, 20. We can notice that GLOBAL obtains the highest number of successful trials in separable, moderate, illconditioned and weak structure noiseless functions, specifically for $f_1, f_2, f_5, f_6, f_8, f_9, f_{10}, f_{11}, f_{12}, f_{21}$ and f_{22} in dimensions 2, 3, and 5. For f_1, f_2, f_5, f_8 and f_9 , the method obtained successful trials for all dimensions (see Figure 1 and Tables 2 and 3).

The scaling of the expected number of function evaluations with the problem dimension is closely linear for f_8, f_9 , is ca. quadratic for f_2 . For f_1 and f_5 we can observe a decreasing tendency (see Figure 1). These results are due to the stochastic nature of the method, usually the ERT grows sub-linearly on these functions. The running times to reach the final target function value in case of the solved problems in 20 dimension range between $25n$ and $2,500n$.

Considering the different function subgroups, the best behavior of the GLOBAL algorithm can be observed on the separable (except f_3 and f_4), moderate (except f_7) and ill-conditioned functions. The good performance on these subgroups are due to the unimodal property of the objective functions. On the other hand, most of these functions have a quite large region of attraction of the global optimum (i.e. f_8 and f_9), hence there is a high chance to sample in this ones. In case of the f_6 (Attractive sector)

and f_{11} (Discus) functions in dimension 20 up to the target precision 10^{-1} , all the 15 trials were successful, but the method fails to reach $\Delta f = 10^{-3}$. The results on the attractive sector function can be improved by increasing the function evaluation limit of the BFGS method, while for the Discuss function f_{11} one cannot find better target precision value than 10^{-1} in 20-D due to a problem of the BFGS local search. In this case the local search method stops too early because it cannot decrease the objective function along the current search direction. Finding the final target function values for f_8 and f_9 are mainly due to the BFGS local search and partly to the property of these functions presented previously. GLOBAL performs also well on the Gallagher's multimodal functions f_{21} and f_{22} with weak global structure. Compared to the best algorithm from BBOB-2009, the GLOBAL method can improve the ERT in dimensions 5 and 20 on the latter functions.

The hardest problems for which the method didn't reach the solution in higher dimensions are the multimodal Rastrigin functions f_3, f_4, f_{15} , and f_{24} . In case of the last one even in 2-D we cannot find a better target precision value than 10^{-2} , while in the case of f_3, f_4, f_{15} functions the Δf_{best} value is not better than 10 in 5-D and 10^2 in 20-D, respectively. The common feature of these functions is that they have more than 10^n local optima. Therefore the algorithm cannot discover the overall function structure. Moreover, the size of the region attraction of the global optimum is small for this problems, and hence the algorithm fails to satisfactorily sample in these regions. GLOBAL also fail to reach a target value below 1 on the f_{17} and f_{19} multimodal functions with adequate global structure in 5-D and 10 in 20-D. The reason is the same as presented above.

Considering the individual maximum number of function evaluations, GLOBAL performs well on ill-conditioned functions and on the multimodal weakly structured functions for a budget smaller than a thousand times n . Similar conclusion has been reached in (Hansen et al., 2010b), where 31 algorithms were compared on a testbed of functions in dimensions up to 40. GLOBAL was ranked together with NEWUOA (Powell, 2006) and MCS (Huyer and Neumaier, 1999) best for a function evaluation budget of up to $500n$ function values, but was no longer competitive when the budget was significantly larger.

5 Comparing GLOBAL with the MATLAB GlobalSearch solver on the BBOB-2010 noiseless testbed

5.1 Improved parameter settings in GLOBAL

The most important parameters used in GLOBAL are the number of sample points to be drawn uniformly in one iteration cycle, the number of best points selected from the actual sample, and the maximal number of function evaluations allowed for a local search. In our previous works (Csendes et al., 2008; Pál et al., 2009), we tuned this parameters empirically and individually for a set of problems without directly including information like the dimension of the problem or the maximal function evaluation budget.

Although, GLOBAL has its own stopping criteria, we introduced a new parameter (`maxfunevals`) with which we control the total number of function evaluations. This parameter is also used in setting the default value of the sample number and the maximal number of function evaluations for local search in the following way:

$$\text{number_of_sample_points} = \min(50 * n, \text{maxfunevals} * 1\%),$$

Algorithm 2: The GlobalSearch solver steps

1. **function** GlobalSearch(f, X, x_0)
 2. Run `fmincon` from x_0
 3. Generate N_1 trial points using the scatter-search mechanism on X
 4. Stage 1: Start a local search from the best trial point among the first N_2 points
 5. Initialize the regions of attraction, counters, threshold, based on the point find in Stage 1 and in the first step of the algorithm
 6. **repeat**
 7. Stage 2: Examine the a remaining trial points and run a local search from a point x , if x is not in any existing basin and $f(x) < threshold$
 8. **until** Reaching *MaxTime seconds* or *running out of trial points*
 9. **return** *The smallest local minimum value found*
-

$$\text{func_eval_nr_in_local_search} = \text{maxfunevals} * 10\%,$$

where n is the dimension of the problem. In our tests, the function evaluation budget is equal to $\text{maxfunevals} = 5 * 10^3 * n$, hence the upper limit of the function evaluations in 20 dimension is 10^5 . On the whole testbed we use the MATLAB `fmincon` local search method with the interior-point algorithm and with $\text{TolFun} = 10^{-12}$ termination tolerance parameter value. The number of the selected best points is 2. The corresponding crafting-effort is equal to $\text{CrE} = 0$.

5.2 Overview of the MATLAB GlobalSearch solver

The GlobalSearch solver is introduced in the MATLAB R2010a version and is available in the new Global Optimization Toolbox. It is a multistart type method which runs a local solver from a variety of starting points in order to find a global minimum, or multiple local minima. The solver uses a scatter-search mechanism for generating start points. It analyzes them and rejects those that are unlikely to improve the best local minimum found so far. Essentially, GlobalSearch accepts a start point only when it determines that the point has a good chance of obtaining a global minimum.

The method uses the `fmincon` function as the local search method, which is a gradient type method. That is, GlobalSearch can be applied mostly to problems with a smooth objective function. The main steps of the GlobalSearch method are summarized in Algorithm 2.

As it can be observed, the algorithm is very similar to GLOBAL procedure, involving cluster formation by identifying the regions of attraction and avoiding local searches from every trial point. For a more detailed description of the algorithm, see (Ugray et al., 2007).

The most important parameters in the GlobalSearch solver are the number of trial points (`NumTrialPoints`) with a default value of 1,000, and the number of points used in Stage 1 (`NumStageOnePoints`) with a default value of 200. We do not have any possibility to control the total function evaluation budget, but we can impose a running time limit using the `MaxTime` parameter. The starting point (used in line 2, Algorithm 2) chosen by us is the center of the search domain $[-5; 5]^n$, but it can be selected also randomly. In the conducted experiments we used the default parameters of the method. For more parameters and settings see the "Using GlobalSearch and MultiStart" explanation in the Global Optimization Toolbox Users Guide for the MATLAB R2010a version.

Algorithm	2-D	3-D	5-D	10-D	20-D	40-D	80-D
GLOBAL	6.8 (215)	6.1 (147)	5.3 (87)	4.3 (43)	3.8 (17)	3.5 (5)	3.6 (2)
GlobalSearch	9.8 (7)	9.5 (8)	7.9 (7)	5.8 (5)	4.6 (2)	3.8 (1)	4.1 (1)

Table 1: CPU time per function evaluation in seconds* 10^{-4} and the corresponding restarts numbers

5.3 Results

In this section we show the comparison results obtained for the GLOBAL and GlobalSearch algorithms. The conducted experiment results according to (Hansen et al., 2010a) on the benchmark functions given in (Finck et al., 2009b; Hansen et al., 2009c) are presented in the Figures 2, 3, 4 and 5 and Tables 4 and 5.

In Tables 4 and 5 we can follow the running time in terms of the number of function evaluations for dimension 5 and 20 in comparison with the respective best algorithm of BBOB-2009. Both algorithms perform very similar considering the number of solved problems in 5 and 20-D. The GlobalSearch method solves 9 and 5 out of 24 functions in 5 and 20-D, while GLOBAL solves the same problems and in addition to that the f_{22} function in 20-D. Compared to the best algorithm from BBOB-2009, the GlobalSearch method can improve the ERT in dimensions 5 and 20 on $f_9, f_{10}, f_{11}, f_{13},$ and f_{14} , while GLOBAL on f_{11} and f_{22} .

Considering the ERT numbers, the GlobalSearch solver is significantly better than GLOBAL on $f_1, f_2, f_5, f_9,$ and f_{14} in dimension 5 as well as in dimension 20. These improvements can be observed usually on unimodal functions and are due to the started local search in line 2 of the Algorithm 2. In contrast to that, the GLOBAL starts the first local search after evaluating all the sample points drawn. On the rest of the functions we can observe similar efficiency values, except those ones (f_{21}, f_{22}), where GLOBAL performs better. Usually these results appear not statistically significant.

Figures 2, 3, 4 and 5 show the empirical cumulative distributions of the runtime in number of function evaluations and of the runtime ratio between the two algorithms in dimensions 5 and 10. The x -values in the figures show a given budget (a given number of function evaluations, divided by dimension), while the y -value gives the proportion of problems where the Δf_t -value was reached within the given budget. As it can be observed, the runtime is shorter for GlobalSearch on separable, moderate and ill-conditioned functions for the reason given previously. A shorter runtime can be observed for GLOBAL on multi-modal functions. The latter fact is due to the way of cluster formation of the two methods. The GLOBAL algorithm uses the Single Linkage method which approximates the level sets more accurately, while GlobalSearch makes the assumption that basins of attraction are spherical. In this case again is true that both of the algorithms work best in the beginning of the search up to a budget of $1,000n$.

The timing experiment results for the two algorithms can be found in Table 1. It contains the CPU per function evaluations and the corresponding number of restarts. As it can be observed, for both algorithms the necessary CPU time decreases with increasing dimension up to 40-D. In the case of the GLOBAL method, this is most likely due to a larger number of initialization procedures for the required multiple runs of the algorithm until thirty seconds have passed, while in the case of the GlobalSearch method there is no dependency to be recognized between the number of restarts and CPU decreasing. The relative small number of restarts are due to the lack of a proper termination criteria.

Using the new parameter settings, GLOBAL is more adaptive to the problem dimension and the found results are similar to the ones obtained on the BBOB-2009 testbed.

6 Summary and conclusion

We have benchmarked the GLOBAL algorithm on the BBOB-2009 noiseless testbed. As a result of the evaluation, we can state that the investigated method performs well on ill-conditioned and multimodal, weakly structured functions up to a small budget, but on multimodal functions the results are usually poorer. We have tried new parameters for the GLOBAL method and compared it with the MATLAB GlobalSearch solver on the BBOB-2010 noiseless testbed. The two methods performed similar except the runtime measured in number of function evaluations. In these cases the GlobalSearch is better on a few problems. The GLOBAL with the new parameters are more adaptive to the dimension of the problem.

Acknowledgements. This work was supported by the grant TÁMOP-4.2.2/08/1/2008-0008 of the Hungarian National Development Agency.

References

- Betrò, B. and Schoen, F. (1992). Optimal and sub-optimal stopping rules for the multistart algorithm in global optimization. *Mathematical Programming*, 57:445–458.
- Boender, C. G. E. and Rinnooy Kan, A. H. G. (1987). Bayesian stopping rules for multistart global optimization methods. *Mathematical Programming*, 37:59–80.
- Boender, C. G. E. and Rinnooy Kan, A. H. G. (1991). On when to stop sampling for the maximum. *Journal of Global Optimization*, 1:331–340.
- Boender, C. G. E., Rinnooy Kan, A. H. G., Timmer, G. T., and Stougie, L. (1982). A stochastic method for global optimization. *Mathematical Programming*, 22:125–140.
- Boender, C. G. E. and Zielinski, R. (1982). A sequential bayesian approach to estimating the dimension of a multinomial distribution. In *Sequential Methods in Statistics. Banach Center Publications*, volume 16. PWN–Polish Scientific Publisher, Warsaw.
- Csendes, T. (1988). Nonlinear parameter estimation by global optimization—efficiency and reliability. *Acta Cybernetica*, 8(4):361–370.
- Csendes, T., Pál, L., Sendin, J. O. H., and Banga, J. R. (2008). The GLOBAL Optimization Method Revisited. *Optimization Letters*, 2:445–454.
- Finck, S., Hansen, N., Ros, R., and Auger, A. (2009a). Real-parameter black-box optimization benchmarking 2009: Presentation of the noiseless functions. Technical Report 2009/20, Research Center PPE.
- Finck, S., Hansen, N., Ros, R., and Auger, A. (2009b). Real-parameter black-box optimization benchmarking 2009: Presentation of the noiseless functions. Technical Report 2009/20, Research Center PPE. Updated February 2010.
- Guss, C., Boender, C. G. E., and Romeijn, H. E. (1995). Stochastic methods. In Horst, R. and Pardalos, P., editors, *Handbook of Global Optimization*, pages 829–869. Kluwer Academic Publishers, Dordrecht.
- Hansen, N., Auger, A., Finck, S., and Ros, R. (2009a). Real-parameter black-box optimization benchmarking 2009: Experimental setup. Technical Report RR-6828, INRIA.
- Hansen, N., Auger, A., Finck, S., and Ros, R. (2010a). Real-parameter black-box optimization benchmarking 2010: Experimental setup. Technical Report RR-7215, INRIA.

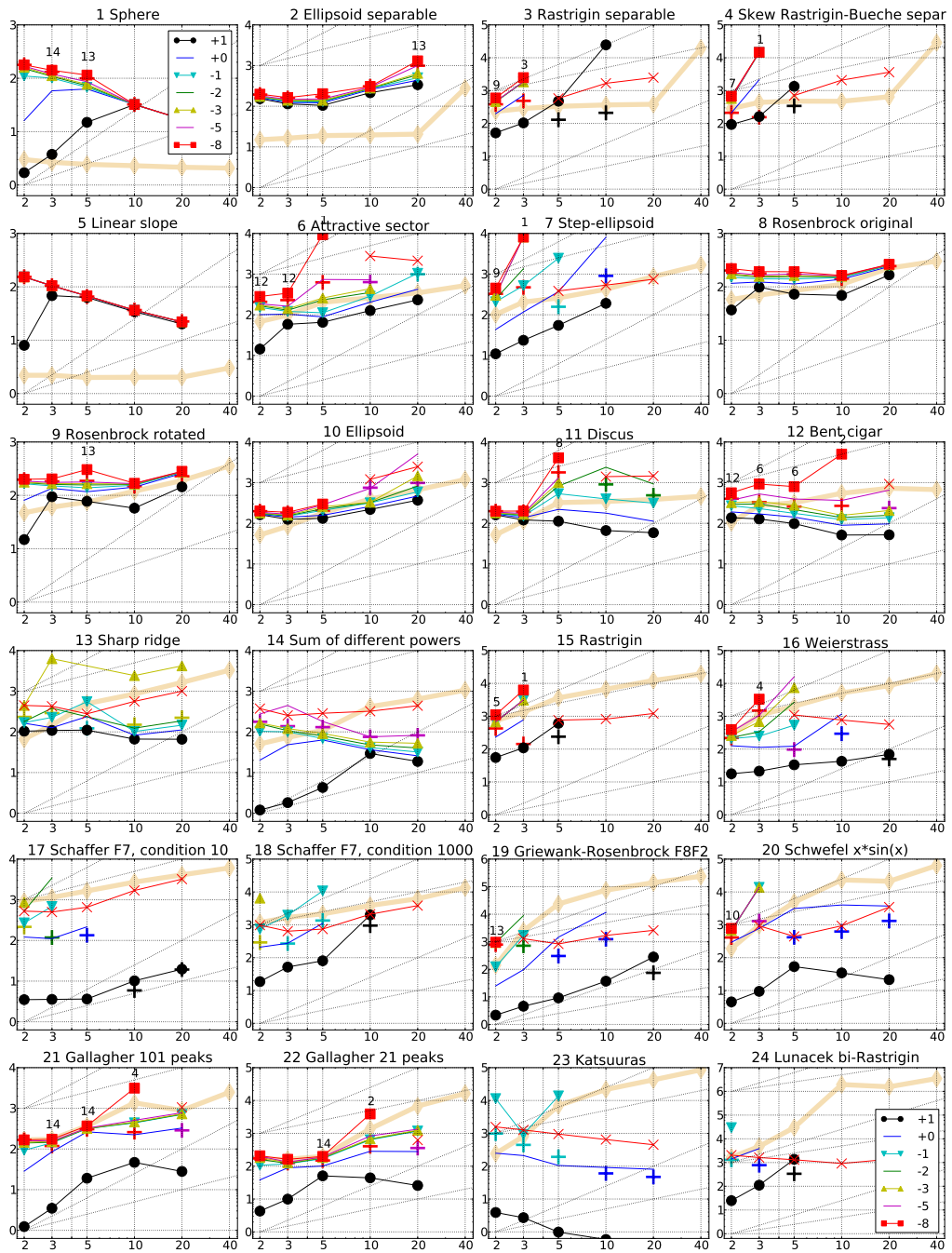


Figure 1: Expected running time (ERT) divided by dimension versus dimension in log-log presentation. Shown are different target values $f_{\text{opt}} + \Delta f$, where $\Delta f = 10^{\{+1,0,-1,-2,-3,-5,-8\}}$ and the exponent is given in the legend of f_1 and f_{24} . Plus symbols (+) show the median number of f -evaluations for the best reached target value. Crosses (x) 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.

Δf	1e+1	1e+0	1e-1	1e-3	1e-5	1e-7	#succ
f_1	11	12	12	12	12	12	15/15
	6.8(9.4)	26(0.61)	28(0.74)	32(1.2)	35(1.0)	39(1.4)	13/15
f_2	83	87	88	90	92	94	15/15
	6.3(1.9)	6.9(1.9)	7.3(1.8)	7.8(1.5)	8.2(1.4)	8.5(1.4)	15/15
f_3	716	1622	1637	1646	1650	1654	15/15
	3.3(3.7)	∞	∞	∞	∞	∞ 2600	0/15
f_4	809	1633	1688	1817	1886	1903	12/15
	8.3(9.2)	∞	∞	∞	∞	∞ 3200	0/15
f_5	10	10	10	10	10	10	15/15
	32(1.3)	33(2.6)	34(2.4)	34(2.4)	34(2.4)	34(2.4)	15/15
f_6	114	214	281	525	723	919	15/15
	2.9(0.21)	2.1(0.60)	2.0(0.50)	2.5(1.7)	5.1(5.4)	51(56)	1/15
f_7	24	283	824	1081	1081	1121	15/15
	12(6.7)	6.5(6.0)	15(17)	∞	∞	∞ 1900	0/15
f_8	73	273	336	391	410	422	15/15
	5.0(0.34)	2.1(1.3)	2.1(1.1)	2.1(0.86)	2.1(0.86)	2.2(0.81)	15/15
f_9	35	127	214	300	335	369	15/15
	11(2.2)	4.6(1.3)	3.2(0.80)	2.8(0.74)	2.7(0.60)	2.7(1.2)	13/15
f_{10}	349	500	574	626	829	880	15/15
	1.9(0.70)	1.6(0.49)	1.8(0.71)	2.0(1.5)	1.7(1.1)	1.7(1.1)	15/15
f_{11}	143	202	763	1177	1397	1535	15/15
	4.0(1.5)	5.5(2.6)	3.5(3.2)	4.3(5.3)	4.7(6.8)	8.3(7.1)	8/15
f_{12}	108	268	371	461	1303	1494	15/15
	4.6(1.2)	2.7(0.61)	2.4(0.82)	3.0(2.1)	1.5(1.2)	1.6(1.4)	6/15
f_{13}	132	195	250	1310	1752	2255	15/15
	4.2(2.5)	6.1(5.2)	11(11)	∞	∞	∞ 1300	0/15
f_{14}	8.1	41	58	139	251	476	15/15
	2.7(2.3)	7.7(0.22)	5.9(0.28)	3.3(0.40)	3.6(2.1)	∞ 1300	0/15
f_{15}	511	8295	15816	16520	17216	17805	14/15
	6.0(7.0)	∞	∞	∞	∞	∞ 2700	0/15
f_{16}	120	612	2663	8585	9704	9990	15/15
	1.4(1.3)	1(0.53)	1(1.2)	4.3(5.0)	8.2(9.5)	8.0(9.0)	0/15
f_{17}	5.2	153	475	2493	5172	7934	15/15
	3.5(3.1)	7.1(5.6)	∞	∞	∞	∞ 3100	0/15
f_{18}	103	378	1635	6176	8749	10418	15/15
	3.9(1.7)	15(14)	33(33)	∞	∞	∞ 2600	0/15
f_{19}	1	1	242	1.14e5	1.16e5	1.17e5	15/15
	46(44)	7329(8170)	∞	∞	∞	∞ 4300	0/15
f_{20}	16	851	16445	22406	23232	24043	10/15
	17(4.9)	18(19)	∞	∞	∞	∞ 2300	0/15
f_{21}	41	1157	1571	1601	1625	1653	14/15
	2.3(2.2)	1.1(0.87)	1(0.81)	1(0.88)	1(0.87)	1(0.86)	14/15
f_{22}	71	386	850	921	953	980	14/15
	3.6(1.7)	1.3(0.90)	1(0.51)	1(0.49)	1(0.45)	1(0.44)	14/15
f_{23}	3.0	518	14249	31654	33030	34256	15/15
	1.6(2.0)	1.0(0.48)	4.8(5.7)	∞	∞	∞ 4900	0/15
f_{24}	1622	74856	95339	1.47e5	1.48e5	1.49e5	4/15
	4.2(4.7)	∞	∞	∞	∞	∞ 6400	0/15

Table 2: Expected running time (ERT) and half-interquantile range (90% – 10%) in number of function evaluations divided by the best ERT measured during BBOB 2009 and 2010 (given in the respective first row) for different Δf values for functions f_1 – f_{24} in 5-D.

Δf	1e+1	1e+0	1e-1	1e-3	1e-5	1e-7	#succ
f_1	43	43	43	43	43	43	15/15
	8.0	8.0	8.0	8.0	8.0	8.0	15/15
f_2	385	386	387	390	391	393	15/15
	18(3.7)	23(3.0)	26(13)	33(14)	51(40)	63(65)	13/15
f_3	5066	7626	7635	7643	7646	7651	15/15
	∞	∞	∞	∞	∞	$\infty 5.0e4$	0/15
f_4	4722	7628	7666	7700	7758	12953	9/15
	∞	∞	∞	∞	∞	$\infty 7.8e4$	0/15
f_5	41	41	41	41	41	41	15/15
	10(0.52)	11(0.78)	11(0.78)	11(0.78)	11(0.78)	11(0.78)	15/15
f_6	1293	1744	2456	3679	4955	6172	15/15
	3.6(1.00)	4.9(0.99)	8.5(4.2)	∞	∞	$\infty 4.1e4$	0/15
f_7	1351	4274	9503	16321	16321	16969	15/15
	∞	∞	∞	∞	∞	$\infty 1.4e4$	0/15
f_8	1959	3745	4040	4219	4371	4447	15/15
	1.7(0.33)	1.3(0.17)	1.2(0.16)	1.2(0.15)	1.2(0.14)	1.2(0.14)	15/15
f_9	1716	3102	3277	3455	3594	3727	15/15
	1.7(0.28)	1.7(0.89)	1.6(0.84)	1.6(0.79)	1.6(0.77)	1.5(0.74)	15/15
f_{10}	7413	8661	10735	11907	12487	13082	15/15
	1(0.22)	1.1(0.15)	1.1(0.53)	2.4(2.1)	8.1(8.9)	$\infty 4.1e4$	0/15
f_{11}	1002	2228	5478	6201	6868	7445	15/15
	1.2(0.42)	1.0(0.60)	1.1(0.96)	∞	∞	$\infty 2.7e4$	0/15
f_{12}	1042	1938	2740	4140	12407	13827	15/15
	1(0.85)	1(0.88)	1(0.70)	1(0.49)	1.1(1.1)	3.4(3.4)	0/15
f_{13}	652	2021	2751	15081	24455	30201	15/15
	2.0(0.34)	1.1(0.08)	1.1(0.04)	5.6(6.5)	∞	$\infty 1.9e4$	0/15
f_{14}	75	239	304	932	1648	10447	15/15
	5.0(0.28)	2.2(0.22)	2.1(0.21)	1.1(0.08)	1(0.04)	$\infty 8500$	0/15
f_{15}	24927	1.47e5	2.15e5	2.24e5	2.33e5	2.42e5	15/15
	∞	∞	∞	∞	∞	$\infty 2.4e4$	0/15
f_{16}	1384	7325	38236	1.50e5	1.62e5	1.68e5	15/15
	1(0.72)	∞	∞	∞	∞	$\infty 1.2e4$	0/15
f_{17}	63	912	2012	23262	46594	70330	15/15
	6.2(1.2)	∞	∞	∞	∞	$\infty 6.9e4$	0/15
f_{18}	621	3024	14625	61478	1.02e5	1.21e5	15/15
	∞	∞	∞	∞	∞	$\infty 7.5e4$	0/15
f_{19}	1	1	2.15e5	2.35e6	2.74e6	2.79e6	15/15
	5601(3531)	∞	∞	∞	∞	$\infty 5.3e4$	0/15
f_{20}	82	5372	2.41e5	3.22e5	3.60e5	4.15e5	15/15
	5.2(0.38)	14(15)	∞	∞	∞	$\infty 7.8e4$	0/15
f_{21}	561	6541	14103	14643	15567	17589	15/15
	1(0.26)	1(1.3)	1(1.2)	1(1.1)	1(1.0)	2.1(2.6)	0/15
f_{22}	467	5580	23491	24948	26847	1.35e5	12/15
	1.1(0.54)	1(1.5)	1(1.1)	1(1.1)	1(0.96)	1.3(1.5)	0/15
f_{23}	3.2	1614	67457	4.89e5	8.11e5	8.38e5	15/15
	2.8(2.7)	1(0.93)	∞	∞	∞	$\infty 9300$	0/15
f_{24}	1.34e6	4.87e6	3.11e7	3.12e7	3.12e7	3.12e7	3/15
	∞	∞	∞	∞	∞	$\infty 2.8e4$	0/15

Table 3: Expected running time (ERT) and half-interquantile range (90% – 10%) in number of function evaluations divided by the best ERT measured during BBOB 2009 and 2010 (given in the respective first row) for different Δf values for functions f_1 – f_{24} in 20-D.

Δf	1e+1	1e+0	1e-1	1e-3	1e-5	1e-7	#succ
f_1	11	12	12	12	12	12	15/15
0: myd	1.2 ^{*2}	2.0 (0.49) ^{*3}	2.5 ^{*3}	3.2 (0.25) ^{*3}	3.9 (0.25) ^{*3}	4.5 (0.49) ^{*3}	15/15
1: myd	6.4(5.0)	22	22(0.49)	23(0.74)	24(0.98)	24(0.98)	15/15
f_2	83	87	88	90	92	94	15/15
0: myd	1.5 (0.69) ^{*3}	1.6 (0.62) ^{*3}	1.7 (0.65) ^{*3}	1.8 (0.65) ^{*3}	2.3 (0.65) ^{*3}	3.0 (0.72) ^{*3}	15/15
1: myd	4.2(0.56)	4.1(0.57)	4.2(0.60)	4.4(0.66)	5.0(0.70)	5.8(1.0)	15/15
f_3	716	1622	1637	1646	1650	1654	15/15
0: myd	11(11)	∞	∞	∞	∞	∞ 6100	0/15
1: myd	3.6 (3.2) ^{*2}	∞	∞	∞	∞	∞ 1700	0/15
f_4	809	1633	1688	1817	1886	1903	12/15
0: myd	22(23)	∞	∞	∞	∞	∞ 5200	0/15
1: myd	7.5(8.7)	27(29)	∞	∞	∞	∞ 2800	0/15
f_5	10	10	10	10	10	10	15/15
0: myd	3.1 ^{*3}	4.9 ^{*3}	6.1 ^{*3}	6.7 ^{*3}	7.3 ^{*3}	86(64)	13/15
1: myd	26(0.30)	27(0.60)	29(0.75)	30(0.60)	30(0.80)	32(3.6)	6/15
f_6	114	214	281	525	723	919	15/15
0: myd	1.3 (0.45) ^{*2}	1.1 (0.29) ^{*3}	1.0 (0.15) ^{*3}	1.1(0.27)	1.1(0.25)	1.1(0.30)	15/15
1: myd	2.6(0.27)	1.8(0.34)	1.6(0.40)	1.1(0.21)	1.1(0.36)	1.2(0.28)	14/15
f_7	24	283	824	1081	1081	1121	15/15
0: myd	40(55)	∞	∞	∞	∞	∞ 2300	0/15
1: myd	50(52)	∞	∞	∞	∞	∞ 1600	0/15
f_8	73	273	336	391	410	422	15/15
0: myd	1.1 (0.14) ^{*3}	1.9(2.6)	1.7(2.1)	1.5(1.8)	1.5(1.8)	1.5(1.7)	15/15
1: myd	4.3(0.27)	2.0(1.2)	1.8(1.00)	1.6(0.85)	1.6(0.84)	1.6(0.80)	14/15
f_9	35	127	214	300	335	369	15/15
0: myd	1.2 (0.13) ^{*3}	0.71 (0.07) ^{*3\downarrow3}	0.64 (0.04) ^{*3\downarrow4}	0.58 (0.03) ^{*3\downarrow4}	0.57 (0.02) ^{*3\downarrow4}	0.55 (0.03) ^{*3\downarrow4}	15/15
1: myd	9.3(0.84)	3.5(1.2)	2.3(0.81)	1.8(0.57)	1.6(0.52)	1.5(0.47)	15/15
f_{10}	349	500	574	626	829	880	15/15
0: myd	0.28 (0.04) ^{*3\downarrow4}	0.22 (0.04) ^{*3\downarrow4}	0.23 (0.05) ^{*3\downarrow4}	0.24 (0.05) ^{*3\downarrow4}	3.7(6.8)	42(52)	0/15
1: myd	0.96(0.07)	0.70(0.05) ^{\downarrow2}	0.62(0.04) ^{\downarrow2}	0.64(0.10) ^{\downarrow2}	2.1(2.9)	16(18)	0/15
f_{11}	143	202	763	1177	1397	1535	15/15
0: myd	0.28 (0.07) ^{*3\downarrow4}	0.25 (0.05) ^{*3\downarrow4}	0.07 (0.01) ^{*3\downarrow4}	0.22 (0.26) ^{\downarrow4}	39(48)	∞ 7.2e4	0/15
1: myd	2.0(0.11)	1.5(0.12)	0.42(0.04) ^{\downarrow4}	0.43(0.30) ^{\downarrow2}	19(23)	∞ 8300	0/15
f_{12}	108	268	371	461	1303	1494	15/15
0: myd	1.3 (0.28) ^{*3}	0.94 (0.46) [*]	0.81(0.48)	0.79(0.58)	2.3(2.4)	28(31)	1/15
1: myd	3.3(0.33)	1.5(0.41)	1.3(0.35)	1.3(0.39)	1.4(1.4)	2.6(2.7)	3/15
f_{13}	132	195	250	1310	1752	2255	15/15
0: myd	0.74 (0.09) ^{*3\downarrow}	0.76 (0.07) ^{*3\downarrow3}	0.81 (0.06) ^{*3\downarrow4}	0.66(1.1)	60(56)	∞ 7200	0/15
1: myd	2.6(0.15)	2.0(0.09)	1.7(0.08)	0.78(0.56)	6.3(6.5)	∞ 1400	0/15
f_{14}	8.1	41	58	139	251	476	15/15
0: myd	1.3(1.5)	0.62 (0.28) ^{*3\downarrow2}	0.71 (0.26) ^{*3\downarrow}	0.64 (0.24) ^{*3\downarrow2}	0.75 (0.15) ^{*3}	∞ 8500	0/15
1: myd	2.0(2.4)	6.4(0.16)	4.8(0.11)	2.4(0.09)	1.6(0.14)	∞ 1200	0/15
f_{15}	511	8295	15816	16520	17216	17805	14/15
0: myd	15(16)	∞	∞	∞	∞	∞ 4700	0/15
1: myd	5.3 (5.8) ^{*3}	∞	∞	∞	∞	∞ 1700	0/15
f_{16}	120	612	2663	8585	9704	9990	15/15
0: myd	6.8(2.9)	212(260)	∞	∞	∞	∞ 1.8e4	0/15
1: myd	2.2 (1.1) ^{*2}	37(41)	∞	∞	∞	∞ 1.7e4	0/15
f_{17}	5.2	153	475	2493	5172	7934	15/15
0: myd	47(53)	343(401)	∞	∞	∞	∞ 2.7e4	0/15
1: myd	1.5(1.3)	198(190)	∞	∞	∞	∞ 7200	0/15
f_{18}	103	378	1635	6176	8749	10418	15/15
0: myd	24(3.7)	∞	∞	∞	∞	∞ 2.1e4	0/15
1: myd	13(17)	∞	∞	∞	∞	∞ 7500	0/15
f_{19}	1	1	242	1.14e5	1.16e5	1.17e5	15/15
0: myd	1 ^{*3}	1 ^{*3}	0.21 (0.04) ^{*3\downarrow4}	∞	∞	∞ 3.6e4	0/15
1: myd	37(51)	4499(5482)	261(317)	∞	∞	∞ 2.0e4	0/15
f_{20}	16	851	16445	22406	23232	24043	10/15
0: myd	1.8 [*]	5.8(5.0)	∞	∞	∞	∞ 3200	0/15
1: myd	11(8.1)	4.6 (5.5) ^{*2}	∞	∞	∞	∞ 1100	0/15
f_{21}	41	1157	1571	1601	1625	1653	14/15
0: myd	2.1(4.4)	1.5(1.7)	2.6(3.9)	2.7(3.4)	2.7(3.8)	2.7(3.7)	9/15
1: myd	3.2(3.1)	0.45(0.45)	0.66(0.97)	0.67(0.94)	0.70(0.93)	0.78(0.94)	11/15
f_{22}	71	386	850	921	953	980	14/15
0: myd	4.6(5.1)	5.8(7.0)	4.4(4.0)	4.1(3.6)	4.0(3.6)	4.1(3.4)	12/15
1: myd	2.4(1.7)	1.0 (0.67) ^{*3}	0.75 (0.59) ^{*3}	0.74 (0.55) ^{*3}	0.84 (0.53) ^{*3}	0.98 (0.56) ^{*3}	9/15
f_{23}	3.0	518	14249	31654	33030	34256	15/15
0: myd	10(11)	3.7(4.5)	1.3(1.3)	11(11)	∞	∞ 2.4e4	0/15
1: myd	3.0(3.0)	2.0(2.5)	1.4(1.7)	∞	∞	∞ 2.5e4	0/15
f_{24}	1622	74856	95339	1.47e5	1.48e5	1.49e5	4/15
0: myd	2.6(2.1)	∞	∞	∞	∞	∞ 5300	0/15
1: myd	5.1(5.4)	∞	∞	∞	∞	∞ 3600	0/15

Table 4: ERT and half-interquartile range (90% – 10%) divided by the best ERT measured during BBOB 2009 and 2010 for different Δf values for functions f_1 – f_{24} in 5-D where 0: myd is GlobalSearch and 1: myd is GLOBAL

Black-box optimization benchmarking of the GLOBAL method

Δf	1e+1	1e+0	1e-1	1e-3	1e-5	1e-7	#succ
f_1	43	43	43	43	43	43	15/15
0: myd	1.2 (0.24)* ³	2.3 (0.49)* ³	2.5 * ³	3.4 * ³	4.4 (0.49)* ³	5.3 (0.49)* ³	15/15
1: myd	24(0.24)	25(0.73)	25(0.24)	27(0.49)	27(0.24)	28(0.49)	15/15
f_2	385	386	387	390	391	393	15/15
0: myd	4.6(1.4)	5.2 (1.1)*	5.6(1.1)	6.4(1.2)	8.6(1.7)	12(7.3)	12/15
1: myd	8.7(2.8)	9.1(2.9)	11(14)	12(14)	14(15)	17(16)	12/15
f_3	5066	7626	7635	7643	7646	7651	15/15
0: myd	∞	∞	∞	∞	∞	∞ 6.3e4	0/15
1: myd	∞	∞	∞	∞	∞	∞ 6.3e4	0/15
f_4	4722	7628	7666	7700	7758	12953	9/15
0: myd	∞	∞	∞	∞	∞	∞ 6.9e4	0/15
1: myd	∞	∞	∞	∞	∞	∞ 7.0e4	0/15
f_5	41	41	41	41	41	41	15/15
0: myd	4.2 * ³	5.7 * ³	6.2 * ³	7.3 * ³	7.8 * ³	380(374)	0/15
1: myd	28(0.52)	30(0.79)	31(0.79)	32(0.79)	32(0.53)	238(249)	0/15
f_6	1293	1744	2456	3679	4955	6172	15/15
0: myd	1.4 (0.39)* ³	1.7(0.39)	1.9(0.45)	2.4(0.32)	70(73)	∞ 5.3e5	0/15
1: myd	2.2(0.50)	2.2(0.61)	2.1(0.65)	3.5(1.8)	41(48)	∞ 4.2e4	0/15
f_7	1351	4274	9503	16321	16321	16969	15/15
0: myd	∞	∞	∞	∞	∞	∞ 3000	0/15
1: myd	∞	∞	∞	∞	∞	∞ 1.8e4	0/15
f_8	1959	3745	4040	4219	4371	4447	15/15
0: myd	0.92 (0.31)* ³	0.99(0.75)	0.96(0.70)	0.96(0.67)	0.95(0.64)	0.94(0.63)	15/15
1: myd	1.5(0.28)	1.1(0.41)	1.1(0.38)	1.1(0.36)	1.0(0.36)	1.0(0.36)	15/15
f_9	1716	3102	3277	3455	3594	3727	15/15
0: myd	0.19 (0.02)* ³ \downarrow ⁴	0.34 (0.02)* ³ \downarrow ⁴	0.38 (0.02)* ³ \downarrow ⁴	0.40 (0.02)* ³ \downarrow ⁴	0.40 (0.02)* ³ \downarrow ⁴	0.40 (0.02)* ³ \downarrow ⁴	15/15
1: myd	1.7(0.20)	1.4(0.45)	1.3(0.42)	1.3(0.41)	1.3(0.39)	1.2(0.38)	15/15
f_{10}	7413	8661	10735	11907	12487	13082	15/15
0: myd	0.20 (0.17) \downarrow ⁴	0.19 (0.17) \downarrow ⁴	0.16 (0.14) \downarrow ⁴	0.17 (0.16) \downarrow ⁴	29(43)	∞ 3.0e5	0/15
1: myd	0.30 (0.04) \downarrow ⁴	0.27 (0.04) \downarrow ⁴	0.23 (0.04) \downarrow ⁴	0.22 (0.04) \downarrow ⁴	3.5(5.0)	∞ 4.2e4	0/15
f_{11}	1002	2228	5478	6201	6868	7445	15/15
0: myd	0.16 (0.05)* ³ \downarrow ⁴	0.10 (0.02)* ³ \downarrow ⁴	0.05 (0.01)* ³ \downarrow ⁴	0.43(0.77)	150(159)	∞ 3.0e5	0/15
1: myd	1.2(0.06)	0.57(0.03)	0.24(0.01) \downarrow ⁴	0.77(0.94)	85(91)	∞ 4.2e4	0/15
f_{12}	1042	1938	2740	4140	12407	13827	15/15
0: myd	0.57 (0.14)* ³	0.75 (0.45)*	0.71 (0.42)*	0.66 (0.17)* ²	1.0(1.2)	6.0(7.3)	0/15
1: myd	2.1(1.1)	1.6(0.83)	1.4(0.58)	1.1(0.38)	0.85(0.61)	3.4(3.5)	0/15
f_{13}	652	2021	2751	15081	24455	30201	15/15
0: myd	1.2 (0.19)* ³	0.47 (0.07)* ³ \downarrow ⁴	0.46 (0.17)* ³ \downarrow ⁴	0.44 (0.43) \downarrow	∞	∞ 8.5e4	0/15
1: myd	2.6(0.22)	1.0(0.15)	1.1(0.49)	0.74(0.64)	∞	∞ 1.1e4	0/15
f_{14}	75	239	304	932	1648	10447	15/15
0: myd	0.92 (0.14)* ³	0.54 (0.09)* ³ \downarrow ³	0.73 (0.14)* ³ \downarrow	0.66 (0.06)* ³ \downarrow ⁴	0.68 (0.14)* ³ \downarrow ⁴	∞ 4.8e4	0/15
1: myd	14(0.14)	4.7(0.09)	3.9(0.11)	1.7(0.07)	1.3(0.07)	∞ 8600	0/15
f_{15}	24927	1.47e5	2.15e5	2.24e5	2.33e5	2.42e5	15/15
0: myd	∞	∞	∞	∞	∞	∞ 4.9e4	0/15
1: myd	∞	∞	∞	∞	∞	∞ 8.0e4	0/15
f_{16}	1384	7325	38236	1.50e5	1.62e5	1.68e5	15/15
0: myd	314(343)	∞	∞	∞	∞	∞ 2.3e5	0/15
1: myd	176(185)	∞	∞	∞	∞	∞ 1.0e5	0/15
f_{17}	63	912	2012	23262	46594	70330	15/15
0: myd	75(140)	∞	∞	∞	∞	∞ 2.2e5	0/15
1: myd	13(7.9)	∞	∞	∞	∞	∞ 1.0e5	0/15
f_{18}	621	3024	14625	61478	1.02e5	1.21e5	15/15
0: myd	2111(2453)	∞	∞	∞	∞	∞ 1.8e5	0/15
1: myd	∞	∞	∞	∞	∞	∞ 1.0e5	0/15
f_{19}	1	1	2.15e5	2.35e6	2.74e6	2.79e6	15/15
0: myd	1 * ³	1 * ³	0.00 * ³ \downarrow ⁴	∞	∞	∞ 7.1e4	0/15
1: myd	2063(2050)	4.02e5(4.67e5)	∞	∞	∞	∞ 1.0e5	0/15
f_{20}	82	5372	2.41e5	3.22e5	3.60e5	4.15e5	15/15
0: myd	1.7 * ³	57(61)	∞	∞	∞	∞ 1.8e4	0/15
1: myd	15(1.4)	24(25)	∞	∞	∞	∞ 2.6e4	0/15
f_{21}	561	6541	14103	14643	15567	17589	15/15
0: myd	3.8(3.0)	6.0(6.0)	6.7(7.8)	6.5(7.4)	6.1(7.0)	5.5(6.0)	3/15
1: myd	2.6(0.93)	1.2(1.5)	0.79(0.87)	0.78(0.90)	0.76(0.78)	0.74(0.74)	7/15
f_{22}	467	5580	23491	24948	26847	1.35e5	12/15
0: myd	0.28 (0.24)* ³	7.1(9.2)	8.7(9.0)	8.2(8.5)	7.7(7.9)	1.6(1.7)	0/15
1: myd	4.7(5.4)	0.85(1.1)	1.1(1.1)	1.0(1.0)	0.98(0.99)	0.30(0.29)	1/15
f_{23}	3.2	1614	67457	4.89e5	8.11e5	8.38e5	15/15
0: myd	13(14)	3.4(3.0)	15(17)	∞	∞	∞ 3.6e5	0/15
1: myd	2.8(3.9)	1.9(1.4)	∞	∞	∞	∞ 1.0e5	0/15
f_{24}	1.34e6	4.87e6	3.11e7	3.12e7	3.12e7	3.12e7	3/15
0: myd	∞	∞	∞	∞	∞	∞ 5.2e4	0/15
1: myd	∞	∞	∞	∞	∞	∞ 3.6e4	0/15

Table 5: ERT and half-interquartile range (90% – 10%) divided by the best ERT measured during BBOB 2009 and 2010 for different Δf values for functions f_1 – f_{24} in 20-D where 0: myd is GlobalSearch and 1: myd is GLOBAL

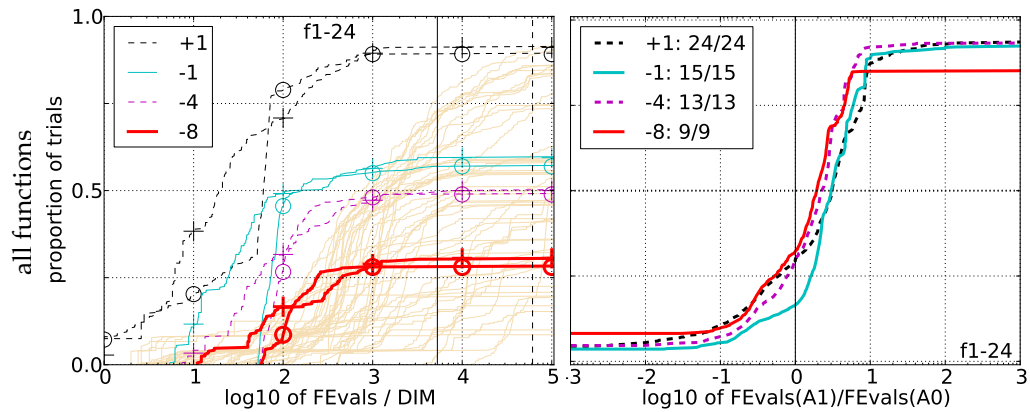


Figure 2: Noiseless functions 5-D. Left: Empirical Cumulative Distribution Function (ECDF) of the running time (number of function evaluations) for GLOBAL (o) and GlobalSearch (+), divided by search space dimension D , to fall below $f_{\text{opt}} + \Delta f$ with $\Delta f = 10^k$ where k is the value in the legend. The vertical black lines indicate the maximum number of function evaluations. Light brown lines in the background show ECDFs for target value 10^{-8} of all algorithms benchmarked during BBOB 2009 and 2010. Right subplots: ECDF of ERT of GLOBAL over ERT of GlobalSearch for different Δf

Hansen, N., Auger, A., Ros, R., Finck, S., and Pošík, P. (2010b). Comparing results of 31 algorithms from the black-box optimization benchmarking bbob-2009. In *GECCO '10: Proceedings of the 12th annual conference comp on Genetic and evolutionary computation*, pages 1689–1696, New York, NY, USA. ACM.

Hansen, N., Finck, S., Ros, R., and Auger, A. (2009b). Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions. Technical Report RR-6829, INRIA.

Hansen, N., Finck, S., Ros, R., and Auger, A. (2009c). Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions. Technical Report RR-6829, INRIA. Updated February 2010.

Hoos, H. H. and Stützle, T. (1998). Evaluating Las Vegas algorithms—pitfalls and remedies. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 238–245.

Huyer, W. and Neumaier, A. (1999). Global optimization by multilevel coordinate search. *J. of Global Optimization*, 14(4):331–355.

Järvi, T. (1973). A random search optimizer with an application to a max-min problem. *Publications of the Institute for Applied Mathematics*, (3). University of Turku, Finland.

Kelley, C. T. (1999). *Iterative Methods for Optimization*. SIAM in Philadelphia.

Locatelli, M. and Schoen, F. (1999). Random linkage: a family of acceptance/rejection algorithms for global optimization. *Mathematical Programming*, 2:379–396.

Nelder, J. and Mead, R. (1965). The downhill simplex method. *Computer Journal*, 7:308–313.

Pál, L., Csendes, T., Markót, M. C., and Neumaier, A. (2009). BBO-benchmarking of the GLOBAL method for the noiseless function testbed. <http://www.mat.univie.ac.at/~neum/papers.html>. P. 986.

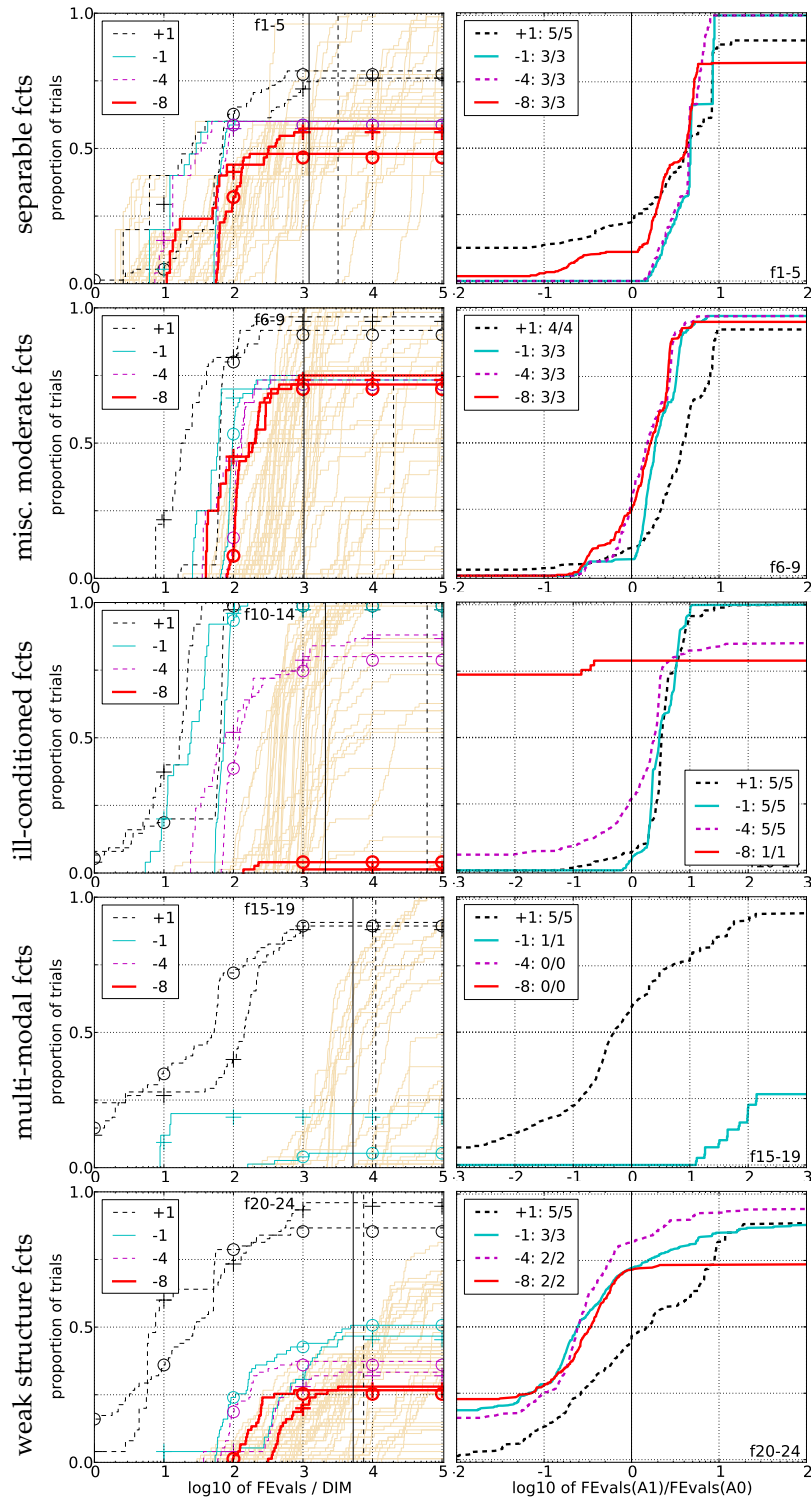


Figure 3: Subgroups of functions 5-D. See caption of Figure 2

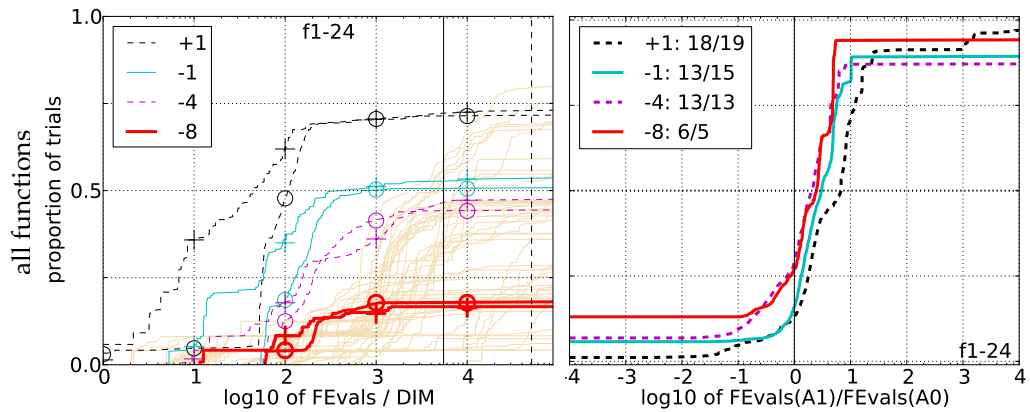


Figure 4: Noiseless functions 20-D. See caption of Figure 2

Powell, M. J. D. (2006). The NEWUOA software for unconstrained optimization without derivatives. *Large Scale Nonlinear Optimization*, pages 255–297.

Price, K. (1997). Differential evolution vs. the functions of the second ICEO. In *Proceedings of the IEEE International Congress on Evolutionary Computation*, pages 153–157.

Rinnooy Kan, A. H. G. and Timmer, G. T. (1987a). Stochastic global optimization methods Part I: Clustering methods. *Mathematical Programming*, 39:27–56.

Rinnooy Kan, A. H. G. and Timmer, G. T. (1987b). Stochastic global optimization methods Part I: Clustering methods. *Mathematical Programming*, 39:57–78.

Törn, A. A. (1978). A search clustering approach to global optimization. In Dixon, L. and Szegő, G., editors, *Towards Global Optimization 2*. North-Holland, Amsterdam, The Netherlands.

Ugray, Z., Lasdon, L., Plummer, J., Glover, R., Kelly, J., and Marti, R. (2007). Scatter search and local nlp solvers: A multistart framework for global optimization. *INFORMS Journal on Computing*, 19(3):328340.

Zieliński, R. (1981). A stochastic estimate of the structure of multi-extremal problems. *Mathematical Programming*, 21:348–356.

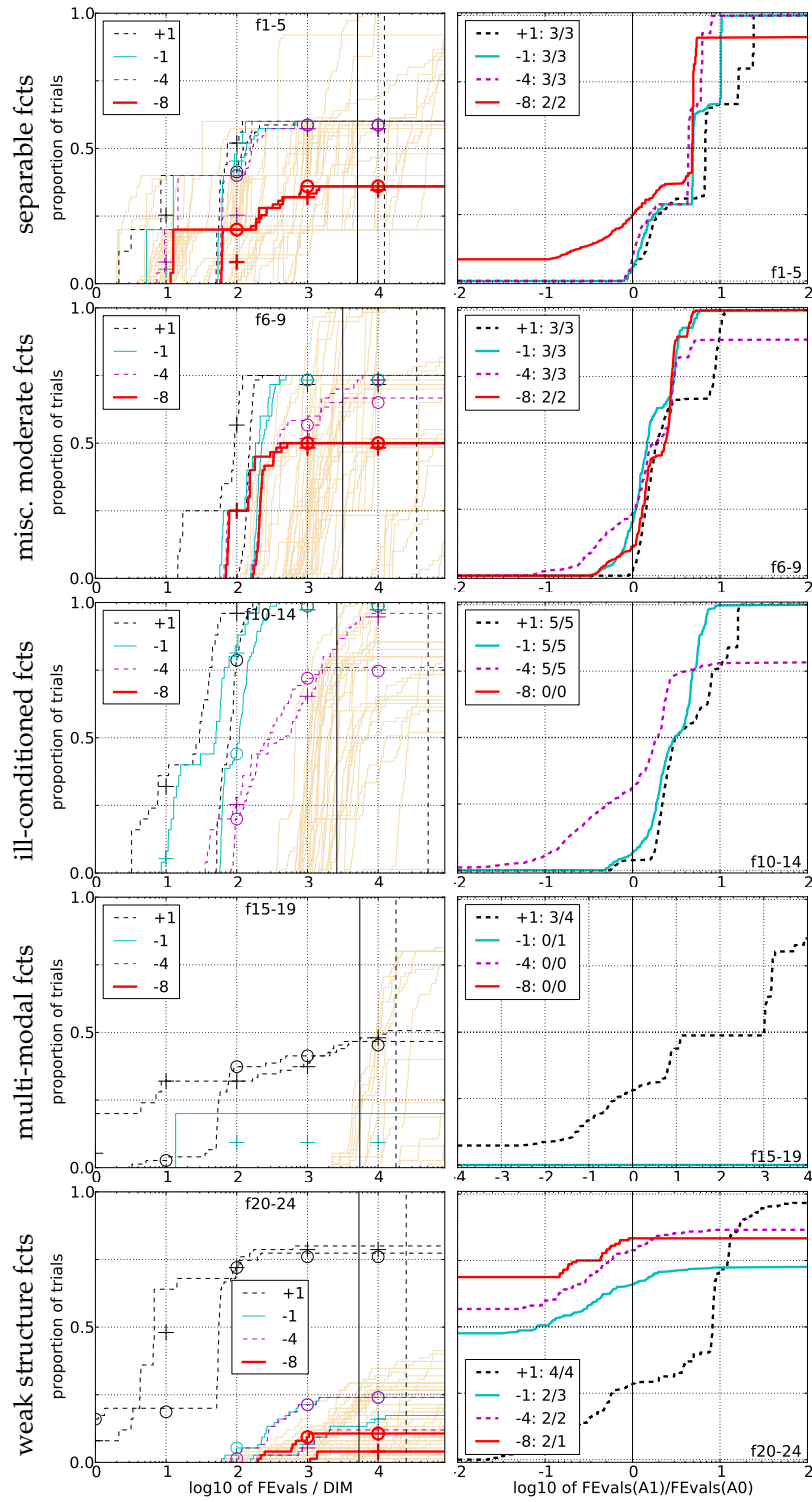


Figure 5: Subgroups of functions 20-D. See caption of Figure 2