Benchmarking a Hybrid Multi Level Single Linkage Algorithm on the BBOB Noiseless Testbed

László Pál
Sapientia - Hungarian University of Transylvania
530104 Miercurea-Ciuc, Piata Libertatii, Nr. 1, Romania
pallaszlo@sapientia.siculorum.ro

ABSTRACT

Multi Level Single Linkage (MLSL) is a well known stochastic global optimization method. In this paper, a new hybrid variant (HMLSL) of the MLSL algorithm is presented. The most important improvements are related to the sampling phase: the sample is generated from a Sobol quasi-random sequence and a few percent of the population is further improved by using crossover and mutation operators like in a traditional differential evolution (DE) method.

The aim of this study is to evaluate the performance of the new HMLSL algorithm on the testbed of 24 noiseless functions. The new algorithm is also compared against a simple MLSL and a traditional DE in order to identify the benefits of the applied improvements.

The results confirm that the HMLSL outperforms the MLSL and DE methods. The new method has a larger probability of success and usually is faster especially in the final stage of the optimization than the other two algorithms.

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, Multi level methods, Differential evolution

1. INTRODUCTION

The Multi Level Single Linkage (MLSL) [12] method has been derived from clustering [1] methods which enable the exploration of the whole feasible region through random sampling followed by local search methods. It is considered one of the best known and efficient stochastic algorithm for global optimization problems of moderate size of dimensions. A similar clustering type algorithm [2] achieved good results [5] on the BBOB-2009 functions with moderate number of local minima using a small budget of function evaluations.

In this paper, we introduce a new hybrid variant of the MLSL method denoted by HMLSL. The most important improvements are related two the sampling phase: the sample is generated from a Sobol quasi-random sequence [7] and a few percent of the sample is further improved using crossover and mutation operators like in a traditional differential evolution (DE) [13] method.

The purpose of this paper is to evaluate the performance of the HMLSL algorithm using the COCO framework [4] and to assess the benefits of the introduced improvements. We also compare the HMLSL method against a simple MLSL and a traditional DE method.

The rest of this article is organized as follows. Section 2 reviews the MLSL algorithm, and also presents the new hybrid version of the MLSL and DE methods. In Section 3, we describe the experiment design together with the algorithms parameter settings. The results are presented in Section 4 and discussed in Section 5. Section 6 concludes the paper and points out some directions for future work.

2. ALGORITHM PRESENTATION

Similarly to the clustering methods, MLSL has two phases: a global and a local one. The global phase consists of sampling, while the local phase is based on local searches. The local minimizer points are found by means of a local search procedure (LS), starting from appropriately chosen points from the sample drawn uniformly within the set of feasibility. The local search procedure is applied to every sample point from the reduced sample, except if there is another sample point within some critical distance r_k, which has a lower function value (see Algorithm 1). The reduced sample consists of the γkN best points ($0<\gamma \leq 1$) from the cumulated sample $x_1, \ldots, x_{kN}$. The critical distance 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 resulting minimum is known already, and the probability that no local search is started in a level set which contains reduced sample points.

The critical distance is given by the following formula

$$r_k(x) = \frac{1}{\sqrt{\pi}}\left(\Gamma(1+\frac{n}{2})\cdot m(X)\cdot\frac{\zeta\ln(kN)}{kN}\right)^{1/n},$$

GECCO'13 Companion, July 6–10, 2013, Amsterdam, The Netherlands.

Algorithm 1: The MLSL algorithm

$X^* \leftarrow \varnothing$; $k \leftarrow 0$
repeat
  $k \leftarrow k+1$
  Generate N points $x_{(k-1)N+1}, \dots, x_{kN}$ with uniform distribution on X.
  Determine the reduced sample (X_r) consisting of the γkN best points from the cumulated sample $x_1, \dots, x_{kN}$.
  for $i \leftarrow 1$ **to** length(X_r) **do**
    if NOT (there is such a j that $f(x_j) < f(x_i)$ and $\|x_j - x_i\| < r_k$) **then**
      Start a local search method (LS) from x_i.
      $x^* \leftarrow LS(x_i)$
      $X^* \leftarrow X^* \cup \{x^*\}$
until Some global stopping rule is satisfied.
return The smallest local minimum value found.

where Γ is the gamma function, n is the number of variables of the problem, $m(X)$ is the Lebesgue measure of the domain X, kN is the total number of sampled points, k is the iteration counter and ζ is some positive constant.

The algorithm continues repeating the global and local phases until some stopping rule is satisfied. It has been proved that the algorithm has good asymptotic properties (depending on the ζ value): the asymptotic probabilistic correctness and probabilistic guarantee of finding all local minimizers. In our calculations the parameter ζ was taken to be 2.

Based on the presented MLSL method, we introduced some improvements which are mainly related to the global step of the algorithm. Low-discrepancy sequences have been used instead of purely random samples. We use sample points from Sobol quasi-random sequences [7] which fill the space more uniformly. Sobol low-discrepancy sequences are superior to pseudorandom sampling especially for low and moderately dimensional problems [8].

Furthermore a few percent of the sample points are improved by using crossover and mutation operators similar to those used in the DE method. In other words, a few DE iterations are applied to the best points of the actual sample. This last step is executed in each iteration before the local phase of the optimization. The aim of these improvements are to help the MLSL method to overcome the difficulties arising in problems with a large number of local optima or in the cases when the local search method cannot make further improvements.

3. EXPERIMENTAL PROCEDURE

The main purpose of the experiment is to assess the benefits of the improvements applied to the MLSL method. Thus we compare the three algorithms on the noiseless function testbed. Each of the algorithms was run on 15 instances of all the 24 functions in dimensions 2, 3, 5, 10, and 20. The evaluations budget was set to $2 \cdot 10^4 D$ for each run. The applied budget is enough to capture all relevant features of the three algorithms.

MLSL has four parameters to set: the number of sample points in an iteration, the size of the reduced sample, the maximum number of function evaluations for local search, and the used local search procedure. The sample size in each iteration was set to $50D$, while the size of the reduced sample to $5D$. This latter setting is also motivated by the same population size of the DE method (see below). On the whole testbed we use the MATLAB's `fmincon` local search method in all dimensions. `fmincon` is an interior-point algorithm for constrained nonlinear problems which approximates the gradient using the finite difference method and based on a recent study [9], it performed well on most of the test functions. The maximum number of function evaluations for local search was set to 10% of the total budget, while the termination tolerance parameter value was set to 10^{-12}.

The HMLSL method posses the same parameter settings as the MLSL algorithm. Additionally we apply $4D$ DE iterations to the reduced sample. The iterations number was selected after a small systematic study and provides a good balance between the two methods.

The population size for DE was set to $5D$ while the crossover and mutation rates to 0.5. Similar population size was also applied in [10]. The crossover strategy is the exponential one and the mutation operator combines the best member with other two randomly chosen individuals.

4. RESULTS

Results from experiments according to [4] on the benchmark functions given in [3, 6] are presented in Figures 1, 2 and 3 and in Tables 1 and 2. The **expected running time (ERT)**, used in the figures and table, 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 executed during each trial 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 [4, 11]. **Statistical significance** is tested with the rank-sum test for a given target Δf_t (10^{-8} as in Figure 1) 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.

4.1 CPU Timing Experiments

The three algorithms were run on the test function f_8, and restarted until at least 30 seconds had passed. These experiments were carried out on a machine with Intel Dual-Core processor, 2.6 Ghz, with 2 GB RAM, on Windows 7 64bit in MATLAB R2011b 64bit. The average time per function evaluation in 2, 3, 5, 10, 20, 40 dimensions was about 14, 9.6, 6.7, 5.0, 2.9, 3.7×10^{-4} s for HMLSL, about 13, 8.9, 6.9, 5.2, 3.9, 3.7×10^{-4} s for MLSL, and about 2.1, 2.1, 2.1, 2.1, 2.2, 2.1×10^{-4} s for DE.

5. DISCUSSION

As a result of the hybridization the HMLS method is usually better than the MLSL algorithm in terms of the ERT needed to find the $\Delta f = 10^{-8}$. Moreover HMLSL is significantly faster than MLSL on the f_3, f_4, f_7, f_{10}, f_{11}, f_{13}, f_{14}, f_{16}, f_{17}, and f_{18} functions (see Figure 1). Compared to the DE method, HMLSL is significantly faster on the f_1, f_8, f_9, f_{21}, and f_{22} functions.

Considering the proportion of solved instances we can state that the new HMLSL method inherits the speed of the simple MLSL algorithm on the initial phase of the optimization, while the use of the DE method inside the MLSL provides a better performance in the final stage.

In 5-D (see Figure 2), the general aspect is that the HMLSL method is as fast as the MLSL algorithm in the initial stage (#FEs $< 100D$) of the optimization, while in the middle and final phases (#FEs $> 100D$) it is usually faster than the MLSL and DE methods. These properties can be nicely followed in the figure with all functions aggregated and on the multi-modal functions subgroup. On the weakly structured multi-modal functions the MLSL is slightly faster than HMLSL in the initial stage (200D<#FEs $< 700D$) of the optimization. After $700D$ evaluations the HMLSL method becomes the leader and up to the final budget solves around 78% of the problems.

As a result of the hybridization, the HMLSL method is significantly better then the MLSL on the separable, moderate and multi-modal function subgroups. This increase is caused by solving the f_3, f_4, f_{15}, f_{17}, f_{18}, and f_{19} functions, where MLSL was able to solve only the problems with loose target levels.

In the 20-D space, similar aspects can be observed as in 5-D (see Figure 3). Considering all functions aggregated for larger budgets than 10^4D, HMLSL is the best algorithm, solving almost 70% of the problems, followed by MLSL, and DE solving about 58%, and 45% of the problems, respectively. Significant improvements can be observed on the moderate functions subgroup where HMLSL solved 100% of the problems, followed by MLSL and DE (75% and 70%, respectively). This is due to the one solved instance of the f_7 function by HMLSL. The lowest percentage (about 35%) of the solved problems by HMLSL can be observed on the multi-modal functions subgroup. This is due to the difficulties of the MLSL and DE methods on these functions.

On the ill-conditioned and weakly structured functions the MLSL method is slightly faster in the middle stage of the optimization, while on the moderate and ill-conditioned function subgroups the HMLSL method (with MLSL) is even faster than the best algorithm from BBOB-2009 on the initial phase ($D <$ #FEs $< 200D$) of the optimization.

6. CONCLUSIONS

We benchmarked the HMLSL algorithm, a hybrid version of the classic MLSL and DE methods. The new hybrid algorithm differs from MLSL in that it applies a few DE iterations in the global phase. The new algorithm was extensively compared with the MLSL and DE methods on the testbed of 24 noiseless functions in order to reveal the benefits of the new improvements.

The results show that the HMLSL outperforms the MLSL and DE methods. The new method has a larger success probability and is as fast as the MLSL method in the initial and middle phase while in the final stage of the optimization it is usually faster than the other two algorithms.

Further improvements by using adaptive DE remains to be investigated as a future work.

Acknowledgements

This work was supported by the Sapientia Foundation - Institute for Scientific Research with the grant No. 101/9/2013.

7. REFERENCES

[1] C. G. E. Boender, A. H. G. Rinnooy Kan, G. T. Timmer, and L. Stougie. A stochastic method for global optimization. *Mathematical Programming*, 22:125–140, 1982.

[2] T. Csendes, L. Pál, J.-O. H. Sendín, and J. R. Banga. The GLOBAL optimization method revisited. *Optimization Letters*, 2(4):445–454, 2008.

[3] 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. Updated February 2010.

[4] N. Hansen, A. Auger, S. Finck, and R. Ros. Real-parameter black-box optimization benchmarking 2012: Experimental setup. Technical report, INRIA, 2012.

[5] N. Hansen, A. Auger, R. Ros, S. Finck, and P. Pošík. 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, 2010. ACM.

[6] 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. Updated February 2010.

[7] H. S. Hong and F. J. Hickernell. Algorithm 823: Implementing Scrambled Digital Sequences. *ACM Transactions on Mathematical Software*, 29:95–109, 2003.

[8] S. Kucherenko and Y. Sytsko. Application of Deterministic Low-Discrepancy Sequences in Global Optimization. *Computational Optimization and Applications*, 30:297–318, 2005.

[9] L. Pál, T. Csendes, M. C. Markót, and A. Neumaier. Black-box optimization benchmarking of the GLOBAL method. *Evolutionary Computation*, 20:609–639, 2012.

[10] P. Pošík and V. Klemš. JADE, an Adaptive Differential Evolution Algorithm, Benchmarked on the BBOB Noiseless Testbed. In *GECCO 2012: Genetic and Evolutionary Computation Conference Companion*, pages 197–204, New York, NY, USA, 2012. ACM.

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

[12] A. H. G. Rinnooy Kan and G. T. Timmer. Stochastic global optimization methods part II: Multi level methods. *Mathematical Programming*, 39:57–78, 1987.

[13] R. Storn and K. Price. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11:341–359, 1997.

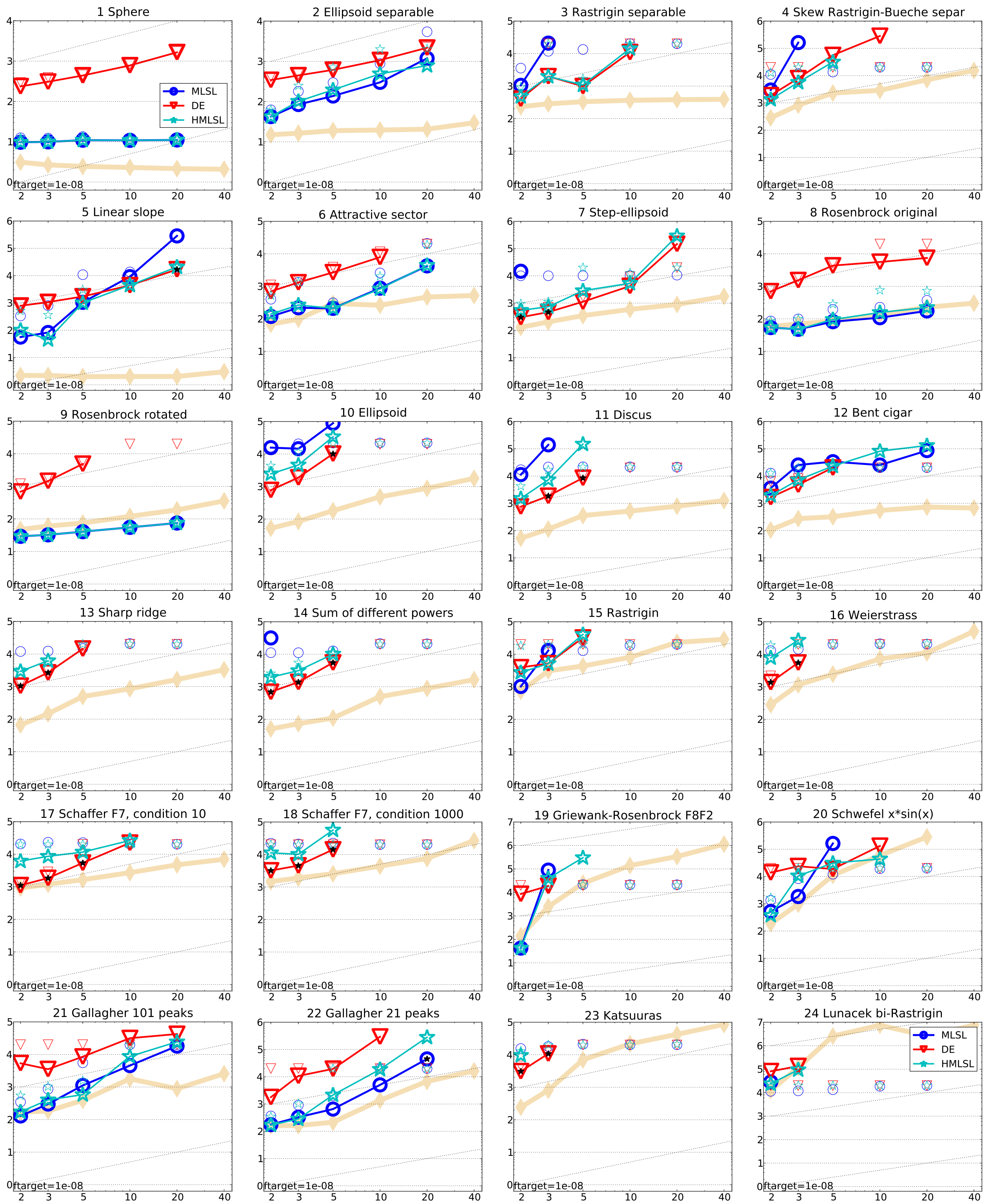

Figure 1: Expected running time (ERT in number of f-evaluations) divided by dimension for target function value 10^{-8} as $\log_{10}$ values versus dimension. Different symbols correspond to different algorithms given in the legend of f_1 and f_{24}. Light symbols give the maximum number of function evaluations from the longest trial divided by dimension. Horizontal lines give linear scaling, slanted dotted lines give quadratic scaling. Black stars indicate statistically better result compared to all other algorithms with $p < 0.01$ and Bonferroni correction number of dimensions (six). Legend: ○:MLSL, ▽:DE, ⋆:HMLSL

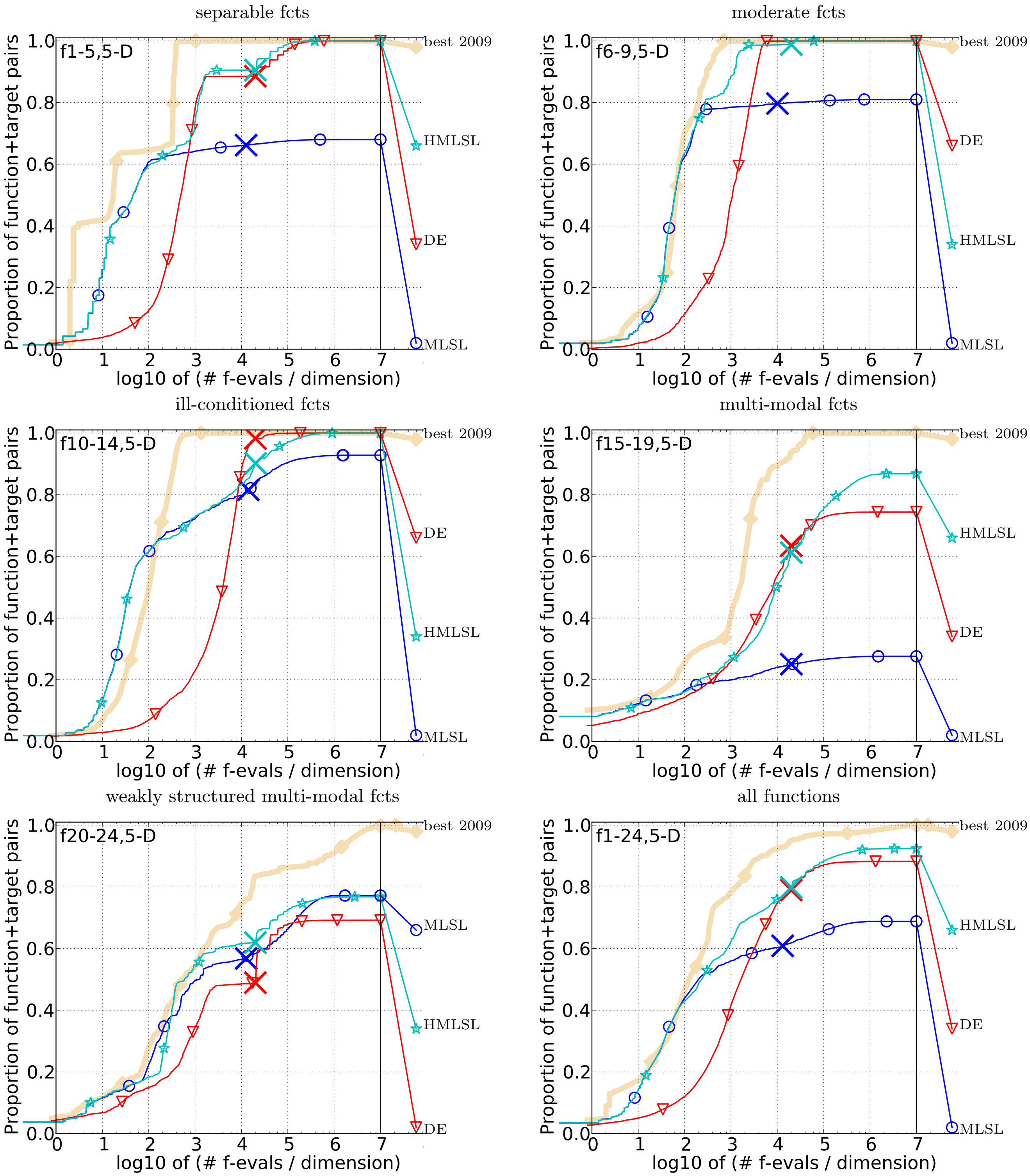

Figure 2: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/D) for 50 targets in $10^{[-8..2]}$ for all functions and subgroups in 5-D. The "best 2009" line corresponds to the best ERT observed during BBOB 2009 for each single target.

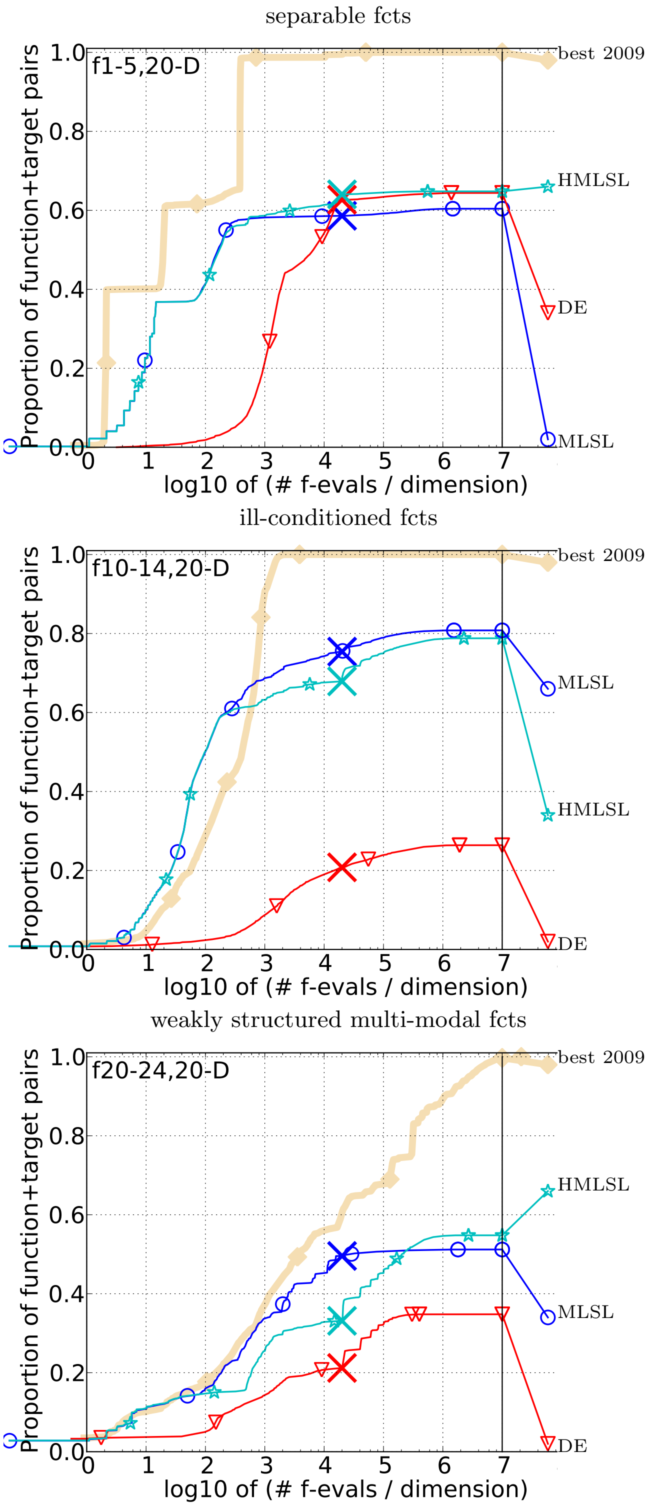

Figure 3: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/D) for 50 targets in $10^{[-8..2]}$ for all functions and subgroups in 20-D. The "best 2009" line corresponds to the best ERT observed during BBOB 2009 for each single target.

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f1	11	12	12	12	12	12	15/15
MLSL	**0.71**(0.3)	**1.4**(0.5)	**2.0**(0)	**2.6**(0.2)	**3.4**(0.2)	**3.9**(0.5)	15/15
DE	4.9(3)	20(7)	39(5)	79(7)	120(9)	157(8)	15/15
HMLSL	**0.71**(0.3)	**1.4**(0.5)	**2.0**(0)	**2.6**(0.2)	**3.4**(0.2)	**3.9**(0.5)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f2	83	87	88	90	92	94	15/15
MLSL	**1.9**(1)	**2.1**(1)	**2.3**(1)	**2.6**(1)	**3.7**(1)	**5.1**(5)	15/15
DE	11(1)	13(1)	15(1)	20(1)	25(1)	30(1)	15/15
HMLSL	**1.9**(1)	**2.1**(1)	**2.3**(1)	**2.6**(1)	3.9(1)	5.5(7)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f3	716	1622	1637	1646	1650	1654	15/15
MLSL	5.6(5)	279(315)	∞	∞	∞	∞ *6e4*	0/15
DE	**1.2**(0.4)	**1.7**(0.3)	**1.9**(0.3)$^{\star 2}$	**2.3**(0.3)$^{\star 2}$	**2.5**(0.3)*	**2.8**(0.4)	15/15
HMLSL	1.3(0.2)	2.2(1)	3.2(1.0)	3.2(1)	3.3(1)	3.3(1)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f4	809	1633	1688	1817	1886	1903	15/15
MLSL	22(24)	∞	∞	∞	∞	∞ *6e4*	0/15
DE	**1.3**(0.4)	**12**(31)*	165(208)	154(193)	149(186)	147(184)	4/15
HMLSL	1.5(0.7)	13(31)	**93**(119)	**87**(111)	**84**(106)	**83**(105)	6/15

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f5	10	10	10	10	10	10	15/15
MLSL	**2.5**(0)	**4.3**(0)	**5.5**(0)	**6.1**(0)	**6.7**(0)	**18**(16)	14/15
DE	29(14)	108(17)	193(31)	389(26)	572(37)	771(41)	15/15
HMLSL	**2.5**(0)	**4.3**(0)	**5.5**(0)	**6.1**(0)	**6.7**(0)	87(159)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f6	114	214	281	580	1038	1332	15/15
MLSL	**1.2**(0.6)	**1.1**(0.4)	**1.2**(0.5)	**0.85**(0.3)	**0.73**(0.2)	**0.71**(0.2)	15/15
DE	2.5(1)	5.1(2)	7.6(1)	8.9(2)	8.1(2)	8.9(3)	15/15
HMLSL	**1.2**(0.6)	**1.1**(0.4)	**1.2**(0.5)	**0.85**(0.3)	**0.73**(0.2)	**0.71**(0.2)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f7	24	324	1171	1572	1572	1597	15/15
MLSL	64(61)	1097(1314)	∞	∞	∞	∞ *5e4*	0/15
DE	**9.0**(9)	**2.8**(1)	**1.7**(1)	**2.8**(0.8)	**2.8**(0.8)	**3.0**(1)	15/15
HMLSL	16(3)	3.8(2)	8.7(1)	8.5(2)	8.5(2)	8.8(2)	14/15

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f8	73	273	336	391	410	422	15/15
MLSL	**1.0**(0.3)	**1.0**(1.0)	**0.99**(0.8)	**0.95**(0.7)	**0.97**(0.6)	**0.96**(0.6)	15/15
DE	8.0(2)	7.6(2)	17(3)	26(4)	36(6)	46(6)	15/15
HMLSL	**1.0**(0.3)	1.3(2)	1.2(2)	1.1(1)	1.1(1)	1.1(1)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f9	35	127	214	300	335	369	15/15
MLSL	**0.94**(0)	**0.64**(0.1)$_{\downarrow 4}$	**0.61**(0.1)$_{\downarrow 4}$	**0.57**(0.0)$_{\downarrow 4}$	**0.56**(0.0)$_{\downarrow 4}$	**0.54**(0.0)$_{\downarrow 4}$	15/15
DE	24(8)	24(9)	30(6)	38(10)	49(14)	57(16)	15/15
HMLSL	**0.94**(0)	**0.64**(0.1)$_{\downarrow 4}$	**0.61**(0.1)$_{\downarrow 4}$	**0.57**(0.0)$_{\downarrow 4}$	**0.56**(0.0)$_{\downarrow 4}$	**0.54**(0.0)$_{\downarrow 4}$	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f10	349	500	574	626	829	880	15/15
MLSL	**0.28**(0.1)$_{\downarrow 4}$	**0.23**(0.1)$_{\downarrow 4}$	**0.22**(0.1)$_{\downarrow 4}$	**0.60**(0.2)	**16**(15)	129(176)	3/15
DE	23(9)	26(6)	29(6)	42(5)	44(4)	**52**(6)	15/15
HMLSL	**0.28**(0.1)$_{\downarrow 4}$	**0.23**(0.1)$_{\downarrow 4}$	**0.22**(0.1)$_{\downarrow 4}$	**0.60**(0.2)	18(37)	104(114)	8/15

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f11	143	202	763	1177	1467	1673	15/15
MLSL	**0.24**(0.1)$_{\downarrow 4}$	**0.21**(0.1)$_{\downarrow 4}$	**0.07**(0.0)$_{\downarrow 4}$	1.4(2)	56(65)	868(986)	0/15
DE	23(13)	36(13)	16(5)	18(5)	**20**(6)	**23**(6)$^{\star 2}$	15/15
HMLSL	**0.24**(0.1)$_{\downarrow 4}$	**0.21**(0.1)$_{\downarrow 4}$	**0.07**(0.0)$_{\downarrow 4}$	**1.0**(2)	76(104)	441(475)	2/15

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f12	108	268	371	461	1303	1494	15/15
MLSL	**1.3**(0.4)	**0.94**(0.6)	**0.89**(0.7)	**0.93**(0.8)	**1.5**(2)	38(55)	5/15
DE	61(18)	50(20)	70(47)	94(58)	58(44)	64(41)	11/15
HMLSL	**1.3**(0.4)	**0.94**(0.6)	**0.89**(0.7)	**0.93**(0.8)	3.3(2)	**30**(35)	8/15

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f13	132	195	250	1310	1752	2255	15/15
MLSL	**0.74**(0.1)$_{\downarrow 2}$	**0.76**(0.1)$_{\downarrow 4}$	**0.80**(0.1)$_{\downarrow 4}$	4.2(0.5)	50(62)	∞ *7e4*	0/15
DE	11(4)	23(10)	43(15)	20(7)	26(7)	**28**(5)$^{\star 3}$	15/15
HMLSL	**0.74**(0.1)$_{\downarrow 2}$	**0.76**(0.1)$_{\downarrow 4}$	**0.80**(0.1)$_{\downarrow 4}$	**2.0**(3)	**21**(22)	164(157)	0/15

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f14	10	41	58	139	251	476	15/15
MLSL	**0.68**(0.4)	**0.53**(0.2)$_{\downarrow 3}$	**0.64**(0.2)$_{\downarrow 3}$	**0.67**(0.2)$_{\downarrow 2}$	**0.72**(0.1)$_{\downarrow}$	793(909)	0/15
DE	2.2(2)	6.2(3)	11(3)	16(7)	36(12)	**46**(9)*	15/15
HMLSL	**0.68**(0.4)	**0.53**(0.2)$_{\downarrow 3}$	**0.64**(0.2)$_{\downarrow 3}$	**0.67**(0.2)$_{\downarrow 2}$	**0.72**(0.1)$_{\downarrow}$	72(24)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f15	511	9310	19369	20073	20769	21359	14/15
MLSL	10(7)	∞	∞	∞	∞	∞ *6e4*	0/15
DE	4.5(3)	6.5(4)	**5.9**(5)	**7.9**(7)	**7.7**(6)	**7.5**(8)	8/15
HMLSL	**2.3**(1)*	**5.5**(6)	10(11)	10(10)	10(11)	10(10)	6/15

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f16	120	612	2663	10449	11644	12095	15/15
MLSL	4.7(3)	70(54)	∞	∞	∞	∞ *9e4*	0/15
DE	**3.3**(3)	**32**(28)	**183**(182)	∞	∞	∞ *1e5*	0/15
HMLSL	4.4(2)	45(30)	530(639)	∞	∞	∞ *1e5*	0/15

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f17	5.2	215	899	3669	6351	7934	15/15
MLSL	24(35)	125(137)	∞	∞	∞	∞ *1e5*	0/15
DE	**5.4**(5)	**2.6**(1)$^{\star 3}$	**2.3**(0.9)$^{\star 3}$	**1.7**(0.6)$^{\star 3}$	**3.1**(0.9)$^{\star 2}$	**3.2**(0.9)$^{\star 2}$	14/15
HMLSL	26(34)	11(8)	8.2(8)	5.3(2)	4.6(2)	6.9(6)	13/15

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f18	103	378	3968	9280	10905	12469	15/15
MLSL	20(36)	1206(1334)	∞	∞	∞	∞ *1e5*	0/15
DE	**1.9**(1.0)$^{\star 3}$	**4.9**(3)$^{\star 2}$	**1.5**(0.8)$^{\star 3}$	**2.3**(0.6)$^{\star 4}$	**4.1**(1)$^{\star 2}$	**5.3**(4)*	13/15
HMLSL	10(8)	10(7)	5.1(3)	5.2(1)	6.9(2)	14(12)	5/15

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f19	1	1	242	1.2e5	1.2e5	1.2e5	15/15
MLSL	**1**(0)	**1**(0)	**0.17**(0.0)$_{\downarrow 4}$	∞	∞	∞ *1e5*	0/15
DE	34(28)	3306(1883)	1087(1102)	∞	∞	∞ *1e5*	0/15
HMLSL	**1**(0)	**1**(0)	**0.17**(0.0)$_{\downarrow 4}$	**13**(14)	**12**(13)	**12**(14)	1/15

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f20	16	851	38111	54470	54861	55313	14/15
MLSL	**1.4**(0)	7.2(10)	21(25)	15(18)	15(16)	15(16)	1/15
DE	7.7(4)	**2.3**(2)	**2.4**(4)	**1.7**(3)	**1.7**(3)	**1.7**(2)	8/15
HMLSL	**1.4**(0)	2.3(1)	4.1(5)	2.9(4)	2.8(4)	2.8(4)	6/15

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f21	41	1157	1674	1705	1729	1757	14/15
MLSL	2.8(5)	**0.74**(0.7)	1.3(2)	1.3(2)	1.4(2)	1.4(2)	15/15
DE	2.5(2)	1.9(2)	23(31)	24(32)	24(31)	24(31)	11/15
HMLSL	**2.3**(3)	0.88(0.7)	**1.1**(0.6)	**1.1**(0.6)	**1.1**(0.6)	**1.1**(0.6)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f22	71	386	938	1008	1040	1068	14/15
MLSL	**2.7**(3)	**1.8**(3)	1.4(1)	1.3(1)	1.4(1)	**1.5**(1)	15/15
DE	3.5(2)	69(132)	96(108)	92(101)	91(143)	91(95)	8/15
HMLSL	3.6(3)	2.1(2)	**1.3**(0.8)	**1.3**(0.7)	**1.4**(0.7)	1.5(0.8)	14/15

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f23	3.0	518	14249	31654	33030	34256	15/15
MLSL	9.2(11)	**1.9**(2)	2.0(2)	**22**(23)	∞	∞ *1e5*	0/15
DE	**1.9**(2)	31(27)	∞	∞	∞	∞ *1e5*	0/15
HMLSL	9.2(11)	2.4(2)	**1.7**(2)	45(51)	∞	∞ *1e5*	0/15

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f24	1622	2.2e5	6.4e6	9.6e6	1.3e7	1.3e7	3/15
MLSL	3.4(2)	**4.2**(5)	∞	∞	∞	∞ *6e4*	0/15
DE	9.5(8)	∞	∞	∞	∞	∞ *1e5*	0/15
HMLSL	**1.8**(2)	∞	∞	∞	∞	∞ *1e5*	0/15

Table 1: Expected running time (ERT in number of function evaluations) divided by the respective best ERT measured during BBOB-2009 (given in the respective first row) for different Δf values in dimension 5. The central 80% range divided by two is given in braces. The median number of conducted function evaluations is additionally given in *italics*, if $\mathrm{ERT}(10^{-7}) = \infty$. #succ is the number of trials that reached the final target $f_{\mathrm{opt}} + 10^{-8}$. Best results are printed in bold.

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f1	43	43	43	43	43	43	15/15
MLSL	**0.77**(0.2)	**1.7**(0.5)	**1.9**(0.2)	**2.8**(0.2)	**3.7**(0.5)	**4.7**(0.5)	15/15
DE	62(11)	140(15)	220(15)	377(19)	530(20)	687(22)	15/15
HMLSL	**0.77**(0.2)	**1.7**(0.5)	**1.9**(0.2)	**2.8**(0.2)	**3.7**(0.5)	**4.7**(0.5)	15/15
f2	385	386	387	390	391	393	15/15
MLSL	**5.1**(2)	**5.7**(2)	**6.1**(2)	**7.2**(3)	**10**(3)	**13**(9)	15/15
DE	34(2)	43(2)	51(3)	68(2)	85(2)	102(3)	15/15
HMLSL	**5.1**(2)	**5.7**(2)	**6.1**(2)	**7.2**(3)	**10**(3)	15(15)	15/15
f3	5066	7626	7635	7643	7646	7651	15/15
MLSL	∞	$\infty^{\star 4}$	$\infty^{\star 4}$	$\infty^{\star 4}$	$\infty^{\star 4}$	∞ *4e5*$^{\star 4}$	0/15
DE	∞	∞	∞	∞	∞	∞ *4e5*	0/15
HMLSL	**219**(208)$^{\star 4}$	∞	∞	∞	∞	∞ *4e5*	0/15
f4	4722	7628	7666	7700	7758	1.4e5	9/15
MLSL	∞	$\infty^{\star 4}$	$\infty^{\star 4}$	$\infty^{\star 4}$	$\infty^{\star 4}$	∞ *4e5*$^{\star 4}$	0/15
DE	**620**(593)	∞	∞	∞	∞	∞ *4e5*	0/15
HMLSL	622(626)	∞	∞	∞	∞	∞ *4e5*	0/15
f5	41	41	41	41	41	41	15/15
MLSL	**3.6**(0)	**5.2**(0)	**5.7**(0)	**6.7**(0)	**7.3**(0)	**118**(93)$^{\star 3}$	1/15
DE	598(66)	1428(102)	2298(132)	3987(157)	5710(197)	7445(223)	15/15
HMLSL	**3.6**(0)	**5.2**(0)	**5.7**(0)	**6.7**(0)	**7.3**(0)	5412(4236)	14/15
f6	1296	2343	3413	5220	6728	8409	15/15
MLSL	**1.8**(1)	**1.5**(0.9)	**1.5**(0.9)	**1.8**(0.7)	**2.1**(0.6)	2.7(0.6)	15/15
DE	49(27)	63(18)	65(18)	114(82)	∞	∞ *4e5*	0/15
HMLSL	**1.8**(1)	**1.5**(0.9)	**1.5**(0.9)	**1.8**(0.7)	**2.1**(0.6)	**2.5**(0.6)	15/15
f7	1351	4274	9503	16524	16524	16969	15/15
MLSL	∞	∞	∞	∞	∞	∞ *2e5*	0/15
DE	21(5)	77(57)	67(63)	**118**(121)	**118**(111)	**115**(119)	2/15
HMLSL	**21**(5)	**72**(54)	**65**(49)	175(182)	175(176)	343(395)	1/15
f8	2039	3871	4040	4219	4371	4484	15/15
MLSL	**0.84**(0.2)	**0.78**(0.5)	**0.79**(0.5)	**0.79**(0.5)	**0.79**(0.5)	**0.78**(0.5)	15/15
DE	13(1)	27(52)	27(50)	29(48)	30(46)	32(45)	13/15
HMLSL	**0.84**(0.2)	1.0(1)	1.0(1)	1.0(1)	1.0(1)	0.99(1)	15/15
f9	1716	3102	3277	3455	3594	3727	15/15
MLSL	**0.17**(0.0)$_{\downarrow 4}$	**0.34**(0.0)$_{\downarrow 4}$	**0.38**(0.0)$_{\downarrow 4}$	**0.40**(0.0)$_{\downarrow 4}$	**0.40**(0.0)$_{\downarrow 4}$	**0.40**(0.0)$_{\downarrow 4}$	15/15
DE	∞	∞	∞	∞	∞	∞ *4e5*	0/15
HMLSL	**0.17**(0.0)$_{\downarrow 4}$	**0.34**(0.0)$_{\downarrow 4}$	**0.38**(0.0)$_{\downarrow 4}$	**0.40**(0.0)$_{\downarrow 4}$	**0.40**(0.0)$_{\downarrow 4}$	**0.40**(0.0)$_{\downarrow 4}$	15/15
f10	7413	8661	10735	14920	17073	17476	15/15
MLSL	**0.13**(0.0)$_{\downarrow 4}$	**0.12**(0.0)$_{\downarrow 4}$	**0.11**(0.0)$_{\downarrow 4}$	**0.12**(0.0)$_{\downarrow 4}$	41(50)	∞ *4e5*	0/15
DE	∞	∞	∞	∞	∞	∞ *4e5*	0/15
HMLSL	**0.13**(0.0)$_{\downarrow 4}$	**0.12**(0.0)$_{\downarrow 4}$	**0.11**(0.0)$_{\downarrow 4}$	**0.12**(0.0)$_{\downarrow 4}$	**41**(50)	∞ *4e5*	0/15
f11	1002	2228	6278	9762	12285	14831	15/15
MLSL	**0.17**(0.0)$_{\downarrow 4}$	**0.10**(0.0)$_{\downarrow 4}$	**0.04**(9e-3)$_{\downarrow 4}$	**0.86**(1)	∞	∞ *4e5*	0/15
DE	∞	∞	∞	∞	∞	∞ *4e5*	0/15
HMLSL	**0.17**(0.0)$_{\downarrow 4}$	**0.10**(0.0)$_{\downarrow 4}$	**0.04**(9e-3)$_{\downarrow 4}$	1.2(1)	∞	∞ *4e5*	0/15
f12	1042	1938	2740	4140	12407	13827	15/15
MLSL	**0.81**(0.6)	**0.89**(0.6)	**0.83**(0.5)	**0.72**(0.4)	**0.58**(0.5)	**11**(15)	3/15
DE	102(112)	235(310)	1028(1114)	∞	∞	∞ *4e5*	0/15
HMLSL	**0.81**(0.6)	**0.89**(0.6)	**0.83**(0.5)	**0.72**(0.4)	8.8(16)	37(47)	2/15
f13	652	2021	2751	18749	24455	30201	15/15
MLSL	**1.1**(0.2)	**0.68**(0.2)	**0.97**(1.0)	**0.59**(0.7)	∞	∞ *4e5*	0/15
DE	50(9)	103(110)	2144(2399)	∞	∞	∞ *4e5*	0/15
HMLSL	**1.1**(0.2)	0.82(0.2)	13(4)	43(64)	∞	∞ *4e5*	0/15
f14	75	239	304	932	1648	15661	15/15
MLSL	**0.74**(0.3)	**0.49**(0.1)$_{\downarrow 3}$	**0.65**(0.1)$_{\downarrow 2}$	**0.65**(0.1)$_{\downarrow 4}$	**0.67**(0.1)$_{\downarrow 4}$	∞ *4e5*	0/15
DE	22(7)	29(2)	42(4)	∞	∞	∞ *4e5*	0/15
HMLSL	**0.74**(0.3)	**0.49**(0.1)$_{\downarrow 3}$	**0.65**(0.1)$_{\downarrow 2}$	**0.65**(0.1)$_{\downarrow 4}$	**0.67**(0.1)$_{\downarrow 4}$	∞ *4e5*	0/15
f15	30378	1.5e5	3.1e5	3.2e5	4.5e5	4.6e5	15/15
MLSL	$\infty^{\star 3}$	$\infty^{\star 3}$	$\infty^{\star 3}$	$\infty^{\star 3}$	$\infty^{\star 3}$	∞ *4e5*$^{\star 3}$	0/15
DE	∞	∞	∞	∞	∞	∞ *4e5*	0/15
HMLSL	∞	∞	∞	∞	∞	∞ *4e5*	0/15
f16	1384	27265	77015	1.9e5	2.0e5	2.2e5	15/15
MLSL	1274(1415)	∞	∞	∞	∞	∞ *4e5*	0/15
DE	∞	∞	∞	∞	∞	∞ *4e5*	0/15
HMLSL	**478**(497)	∞	∞	∞	∞	∞ *4e5*	0/15
f17	63	1030	4005	30677	56288	80472	15/15
MLSL	22(30)	∞	∞	∞	∞	∞ *4e5*$^{\star 4}$	0/15
DE	**8.2**(4)	**17**(3)	**14**(2)	**10**(7)	**105**(117)	∞ *4e5*	0/15
HMLSL	24(41)	21(7)	18(4)	15(9)	∞	∞ *4e5*	0/15
f18	621	3972	19561	67569	1.3e5	1.5e5	15/15
MLSL	∞	∞	∞	$\infty^{\star 4}$	$\infty^{\star 4}$	∞ *4e5*$^{\star 4}$	0/15
DE	**13**(5)	**19**(5)	**19**(13)	∞	∞	∞ *4e5*	0/15
HMLSL	18(4)	22(6)	20(13)	∞	∞	∞ *4e5*	0/15
f19	1	1	3.4e5	6.2e6	6.7e6	6.7e6	15/15
MLSL	**1**(0)	**1**(0)	**7.3e-4**(0)$_{\downarrow 4}$	∞	∞	∞ *4e5*	0/15
DE	1337(554)	∞	∞	∞	∞	∞ *4e5*	0/15
HMLSL	**1**(0)	**1**(0)	**7.3e-4**(0)$_{\downarrow 4}$	∞	∞	∞ *4e5*	0/15
f20	82	46150	3.1e6	5.5e6	5.6e6	5.6e6	14/15
MLSL	**1.4**(0)	11(10)	∞	$\infty^{\star 4}$	$\infty^{\star 4}$	∞ *4e5*$^{\star 4}$	0/15
DE	36(4)	2.8(1)	∞	∞	∞	∞ *4e5*	0/15
HMLSL	**1.4**(0)	**1.6**(0.6)$^{\star 2}$	**1.9**(2)	∞	∞	∞ *4e5*	0/15
f21	561	6541	14103	14643	15567	17589	15/15
MLSL	**1.3**(2)	**1.0**(0.9)	**1.00**(2)	**0.98**(2)	**0.94**(2)	**0.91**(1)	8/15
DE	62(7)	94(122)	58(71)	56(68)	54(65)	48(57)	5/15
HMLSL	4.2(9)	11(31)	11(15)	11(15)	10(14)	9.3(12)	7/15
f22	467	5580	23491	24948	26847	1.3e5	12/15
MLSL	**3.2**(4)	**3.4**(4)	**4.3**(5)$^{\star 2}$	**4.1**(4)$^{\star 2}$	**3.8**(4)$^{\star 2}$	**0.82**(0.8)$^{\star 2}$	5/15
DE	584(859)	200(251)	∞	∞	∞	∞ *4e5*	0/15
HMLSL	140(429)	64(108)	111(128)	105(128)	97(112)	42(45)	1/15
f23	3.2	1614	67457	4.9e5	8.1e5	8.4e5	15/15
MLSL	11(14)	**3.4**(3)	**25**(30)	∞	∞	∞ *4e5*	0/15
DE	**1.9**(1)	∞	∞	∞	∞	∞ *4e5*	0/15
HMLSL	11(14)	6.4(5)	86(93)	∞	∞	∞ *4e5*	0/15
f24	1.3e6	7.5e6	5.2e7	5.2e7	5.2e7	5.2e7	3/15
MLSL	∞	∞	∞	∞	∞	∞ *4e5*	0/15
DE	∞	∞	∞	∞	∞	∞ *4e5*	0/15
HMLSL	∞	∞	∞	∞	∞	∞ *4e5*	0/15

Table 2: Expected running time (ERT in number of function evaluations) divided by the respective best ERT measured during BBOB-2009 (given in the respective first row) for different Δf values in dimension 20. The central 80% range divided by two is given in braces. The median number of conducted function evaluations is additionally given in *italics*, if $\mathrm{ERT}(10^{-7}) = \infty$. #succ is the number of trials that reached the final target $f_{\mathrm{opt}} + 10^{-8}$. Best results are printed in bold.