MEMPSODE: An Empirical Assessment of Local Search Algorithm Impact on a Memetic Algorithm Using Noiseless Testbed

Costas Voglis*
Department of Computer Science
University of Ioannina
GR-45110 Ioannina, Greece
voglis@cs.uoi.gr

Grigoris S. Piperagkas
Department of Computer Science
University of Ioannina
GR-45110 Ioannina, Greece
gpiperag@cs.uoi.gr

Konstantinos E. Parsopoulos
Department of Computer Science
University of Ioannina
GR-45110 Ioannina, Greece
kostasp@cs.uoi.gr

Dimitris G. Papageorgiou
Department of Materials Science and Engineering
University of Ioannina
GR-45110 Ioannina, Greece
dpapageo@cc.uoi.gr

Isaac E. Lagaris
Department of Computer Science
University of Ioannina
GR-45110 Ioannina, Greece
lagaris@cs.uoi.gr

ABSTRACT

Memetic algorithms are hybrid schemes that usually integrate metaheuristics with classical local search techniques, in order to attain more balanced intensification/diversification trade–off in the search procedure. MEMPSODE is a recently published software that implements such memetic schemes, based on the Particle Swarm Optimization and Differential Evolution algorithms, as well as on the Merlin optimization environment that offers a variety of local search methods. The present study aims at investigating the impact of the selected local search algorithm in the memetic schemes produced by MEMPSODE. Our interest was focused on gradient–free local search methods. We applied the derived memetic schemes on the noiseless testbed of the Black–Box Optimization Benchmarking 2012 workshop. The obtained results can offer significant insight to optimization practitioners with respect to the most promising approaches.

Categories and Subject Descriptors

G.1.6 [**Numerical Analysis**]: Optimization—*global optimization*; G.4 [**Mathematical Software**]

Keywords

Global optimization, memetic algorithms, hybrid algorithms, Black–box optimization, local search

*Corresponding author.

GECCO'12 Companion, July 7–11, 2012, Philadelphia, PA, USA.

1. INTRODUCTION

Memetic Algorithms (MAs) are the outcome of integrating metaheuristics with local search (LS) procedures. The combination of the two approaches is accompanied by increased effectiveness and accuracy for locating solutions of global optimization problems [5, 11].

The most common MA scheme assumes an evolving population and periodically applies LS to some or all its members. The dynamic of the MA is dictated by the choices of the user with respect to the following issues, also referred to as the *fundamental memetic questions* [11]:

(I) *Where* will the LS procedures be applied. The user shall define the individuals that will constitute initial points for the LS.

(II) *When* will the LS procedures be applied. The application frequency of LS shall be determined in order to attain a balanced exploitation of the available computational budget.

(III) *How much* of the computational budget will be consumed at each application of LS. The user must specify the fraction of the available computational budget that will be devoted to LS.

These issues were studied in [12], under a memetic strategy based on the Particle Swarm Optimization (PSO) algorithm.

MEMPSODE (MEMetic Particle Swarm Optimization and Differential Evolution) is a recently introduced [16] software package that implements closely the PSO–based memetic approaches of [12] and extends them also to the Differential Evolution (DE) framework [14]. More specifically, MEMPSODE contains implementations of the Unified PSO (UPSO) approach [10], which generalizes plain PSO by harnessing the strengths of its standard local and global variants, as well as the five basic DE operators. Also, MEMPSODE borrows LS methods from the Merlin optimization environment [9], which offers some of the most fundamental gradient–based and gradient–free LS methods.

Evidently, the choice of the LS scheme in an MA is expected to have an impact on its performance. Attempts to quantify this impact in hybrid evolutionary algorithms, were made in previous works. For instance, in [6] a hill–climbing method [13] was compared to the nonlinear simplex method [7] and CMA–ES [4]. Moreover, in [2] a hybrid PSO was studied under the influence of the nonlinear simplex method [7] and Powell's direction set method [8].

The present paper constitutes a first attempt to investigate the impact of the LS method on the MAs implemented in MEMPSODE. For this reason, we fixed the parameters of all algorithms to specific values and alternated the LS procedures combined with PSO and DE. The resulting memetic schemes were tested on the noiseless testbed of the Black–Box Optimization Benchmarking 2012 (BBOB'12) workshop.

The rest of the paper is organized as follows: the employed algorithms are given in Section 2, while Section 3 describes the experimental setting. The obtained experimental results are reported in Section 5, and the paper concludes in Section 6.

2. EMPLOYED ALGORITHMS

The design of the MA schemes presented in [12], were based on PSO. In this context, the following three schemes were proposed:

Scheme 1: LS is applied only on the overall best position, p_g, of the swarm.

Scheme 2: LS is applied on each locally best position, p_i, $i = 1, 2, \ldots, N$, with a prescribed fixed probability, $\rho \in (0, 1]$.

Scheme 3: LS is applied both on the best position, p_g, as well as on some randomly selected locally best positions, p_i, $i \in \{1, 2, \ldots, N\}$.

These schemes can be applied either at each iteration or whenever a specific number of consecutive iterations has been completed.

In MEMPSODE [16] we extended this strategy to the DE [14] framework and incorporated the Merlin optimization environment for the LS procedures. A detailed presentation of the MEMPSODE software is provided in [16] as well as in the accompanying manual of the software distribution.

A short description of the LS algorithms employed in the present work, are given in the following paragraphs. The notation used for the algorithms' names, follows the corresponding MEMPSODE (Merlin) standard.

BFGS Method

BFGS belongs to the category of quasi–Newton methods with line search [8]. At the start of k–th iteration a point $\mathbf{x}^{(k)}$, the gradient (or an finite–difference approximation) $\mathbf{g}^{(k)}$ and an approximation $\mathbf{B}^{(k)}$ of the Hessian matrix are available. The method continues by performing a line search to a direction $\mathbf{s}^{(k)}$ defined as $\mathbf{B}^{(k)}\mathbf{s}^{(k)} = -\mathbf{g}^{(k)}$, and an update from $\mathbf{B}^{(k)}$ to $\mathbf{B}^{(k+1)}$, using the BFGS update formula [1]. A careful selection of line search, in addition to the powerful local properties of the Hessian approximation, results in one of the most robust and effective optimization algorithms.

TRUST Method

The TRUST method is a quasi–Newton algorithm that uses also BFGS updates to maintain an approximation of the Hessian, but for the step of the line search is properly selected to guarantee convergence, following the trust region strategy [8]. In this strategy, a quadratic model is being build around the current point using the approximation $\mathbf{B}^{(k)}$ and the gradient vector $\mathbf{g}^{(k)}$. The algorithm *trusts* only the quadratic model in a restricted region around the current iterate. The restriction is usually imposed by a radius that defines a hypersphere.

SIMPLEX Method

This method belongs to the class of direct search methods for nonlinear optimization. It was designed by Nelder and Mead [7]. The algorithm is based on the concept of *simplex* (or polytope) in $\mathbb{R}^n$, which is a volume element defined by $(n + 1)$ vertices. The input of the algorithm is an initial simplex. The SIMPLEX algorithm then moves this initial simplex towards the minimum by adapting its geometry and finally shrinks it to a small volume element around the minimizer. The procedure is derivative–free and proceeds towards the minimum using a set of $n + 1$ points. Thus, it is expected to be tolerant to ill–conditioned cases. The transformation rules include *contraction*, *expansion* and *shrinkage* of the initial simplex.

ROLL Method

This method belongs to the class of pattern search methods. It proceeds by taking proper steps along each coordinate direction, in turn. Then, the method performs an one–dimensional search on a properly formed direction, in order to tackle possible correlations among the variables. Its input consists of the initial point and a user–defined exploration factor.

AUTO Method

AUTO is a procedure that tries to automatically select the best LS algorithm. The methods BFGS, ROLL, SIMPLEX and TRUST are invoked one after the other. For each one, a rate is calculated by dividing the relative achieved reduction of the function's value by the number of function calls spent. The method with the highest rate is then invoked again and the procedure is repeated. If all rates assume vanishing values, then all the method tolerances are set to zero and the methods are applied in the following order: `ROLL, TRUST, BFGS, SIMPLEX`.

3. EXPERIMENTAL SETUP

We used the default restart mechanism provided by the noiseless testbed for a maximum number of $10^5 \times n$ function evaluations. The memetic Scheme 3 was used with LS probability $\rho_i = 0.05$. We selected the default DE operator as the main metaheuristic of the MA, with population size $N = 25$ and parameter values $F = 0.5$ and $CR = 0.7$.

Each LS application was restricted to 2000 function evaluations. Whenever first order derivatives were needed (e.g., in BFGS) we applied an $O(h)$ finite–difference formula where h is an adaptable step size [15]. The experiments were conducted on an Intel I7–2600 3.4 GHz processor machine with 8GB RAM.

	Dimension					
Algorithm	2	3	5	10	20	40
BFGS	3.1^{-4}	2.0^{-4}	1.2^{-4}	5.0^{-5}	2.7^{-5}	2.2^{-5}
ROLL	4.6^{-5}	5.4^{-5}	5.7^{-5}	5.6^{-5}	4.5^{-5}	3.8^{-5}
SIMPLEX	2.1^{-4}	1.5^{-4}	9.2^{-5}	5.1^{-5}	5.5^{-5}	5.7^{-5}
AUTO	1.5^{-4}	1.1^{-4}	6.5^{-5}	1.5^{-5}	1.3^{-5}	1.1^{-5}

Table 1: Summary of timing results in seconds per function evaluation.

4. CPU TIMING EXPERIMENT

According to [3], for the timing experiment the experimental setting described above was run on f8 with at most 10^3 function evaluations for each call of MEMPSODE, and restarted until at least 30 seconds had passed. The timing experiment was performed on the same platform as the experimental procedure. The timing results using the four LS algorithms are reported in Table 1.

5. EXPERIMENTAL RESULTS

We compared the performance of the memetic DE scheme with the four LS strategies implemented in MEMPSODE. The results are reported in Tables 2 and 3 and graphically illustrated in Figs. 1–3. The AUTO method proved to be superior in robustness, solving in total 1480 of 2160 experiments up to the desired accuracy, 10^{-8}. BFGS was capable of approaching the minimizer using a small number of function evaluations, although it was successful in a smaller number of experiments (1307 out of 2160).

Figures 2 and 3 reveal that in the 5–dimensional experiments, especially for the separable, ill–conditioned and moderate functions, BFGS and AUTO had the best performance in terms of function evaluations. The same holds also for the 20–dimensional case, favoring the aforementioned methods. Considering the SIMPLEX method, it exhibited improved performance only for the small–dimensional cases.

A closer examination of Fig. 1 and Tables 2 and 3, with respect to the ERT values, suggests that the ROLL method performs nicely in the separable functions, although it cannot dominate in all cases. The BFGS performance is improved in f8–f12 while, in the rest of the cases, the results are not conclusive in terms of ERT. Overall, the results suggest that the algorithms implemented in MEMPSODE can be very competitive.

6. CONCLUSION

We presented a comparison among the LS schemes implemented in the MEMPSODE software, within a memetic DE framework. Among the four tested LS methods, BFGS proved to be more efficient in terms of accuracy and function evaluations. On the other hand the AUTO versatile strategy was very robust in solving most of the moderate and ill–conditioned problems to the prescribed accuracy. Since AUTO is a combination of local search strategies (including BFGS), we may conclude that a portfolio of LS approaches may be a valuable addition to hybrid memetic schemes. Further research will consider also the PSO metaheuristic offered by MEMPSODE, as well as different parameter settings.

7. REFERENCES

[1] R. Fletcher. A new approach to variable metric algorithms. *The Computer Journal*, 13(3):317–322, 1970.

[2] J. Gimmler, T. Stützle, and T. Exner. Hybrid particle swarm optimization: An examination of the influence of iterative improvement algorithms on performance. *Ant Colony Optimization and Swarm Intelligence*, pages 436–443, 2006.

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

[4] N. Hansen and A. Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*, pages 312–317. IEEE, 1996.

[5] J. Kennedy and R. C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann Publishers, 2001.

[6] D. Molina, M. Lozano, C. García-Martínez, and F. Herrera. Memetic algorithms for continuous optimisation based on local search chains. *Evolutionary Computation*, 18(1):27–63, 2010.

[7] J. Nelder and R. Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.

[8] J. Nocedal and S. Wright. *Numerical optimization*. Springer verlag, 1999.

[9] D. Papageorgiou, I. Demetropoulos, and I. Lagaris. MERLIN-3.1. 1. A new version of the Merlin optimization environment. *Computer Physics Communications*, 159(1):70–71, 2004.

[10] K. E. Parsopoulos and M. N. Vrahatis. Parameter selection and adaptation in unified particle swarm optimization. *Mathematical and Computer Modelling*, 46(1–2):198–213, 2007.

[11] K. E. Parsopoulos and M. N. Vrahatis. *Particle Swarm Optimization and Intelligence: Advances and Applications*. Information Science Publishing (IGI Global), 2010.

[12] Y. G. Petalas, K. E. Parsopoulos, and M. N. Vrahatis. Memetic particle swarm optimization. *Annals of Operations Research*, 156(1):99–127, 2007.

[13] F. Solis. Minimization by random search techniques. *Mathematics of operations research*, pages 19–30, 1981.

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

[15] C. Voglis, P. Hadjidoukas, I. Lagaris, and D. Papageorgiou. A numerical differentiation library exploiting parallel architectures. *Computer Physics Communications*, 180(8):1404–1415, 2009.

[16] C. Voglis, K. Parsopoulos, D. Papageorgiou, I. Lagaris, and M. Vrahatis. Mempsode: A global optimization software based on hybridization of population-based algorithms and local searches. *Computer Physics Communications*, 183(5):1139–1154, 2012.

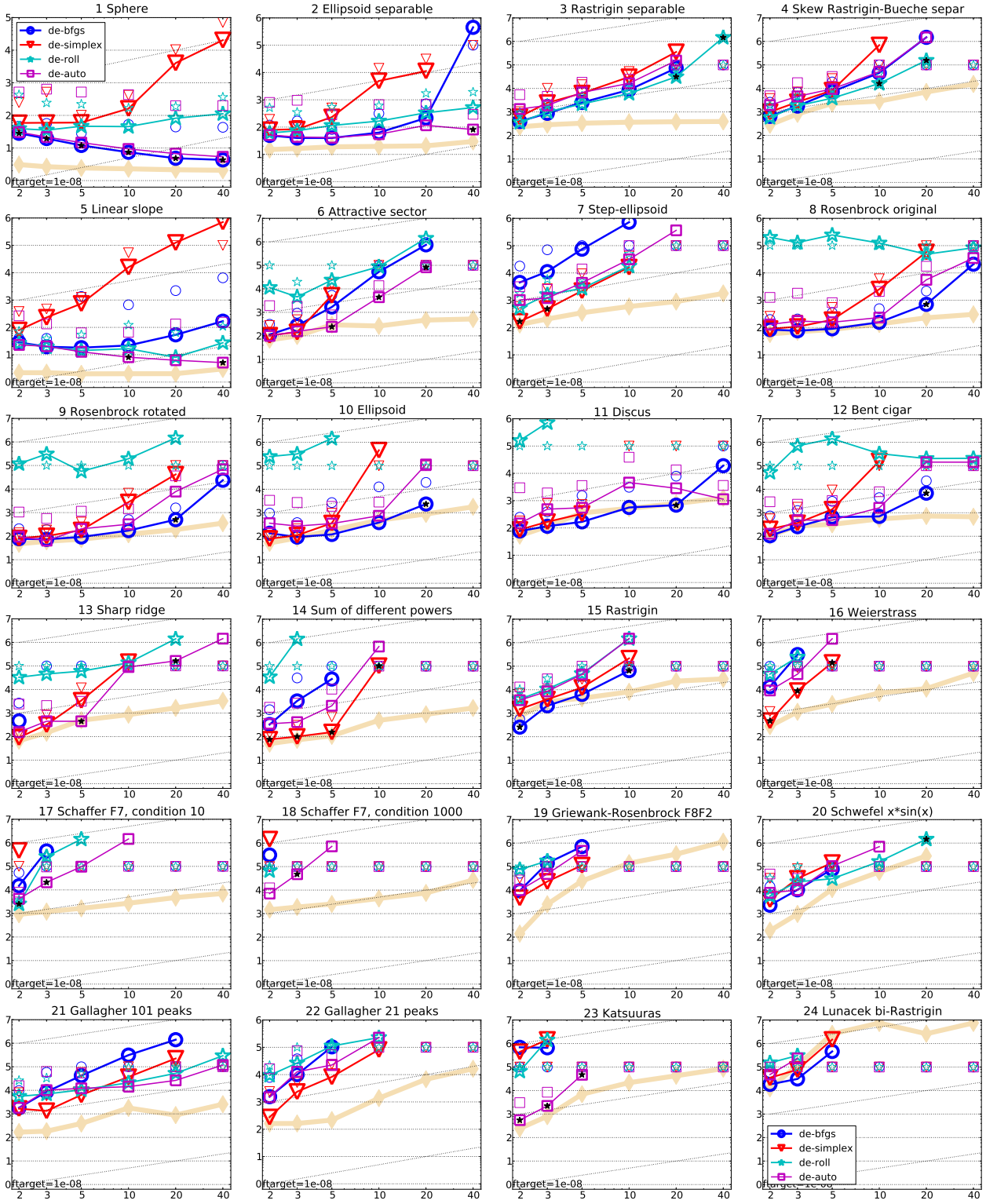

Figure 1: Expected running time (ERT in number of f-evaluations) divided by dimension for target function value 10^{-8} as $\log_{10}$ values versus dimension. Different symbols correspond to different algorithms given in the legend of f_1 and f_{24}. Light symbols give the maximum number of function 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: ○: bfgs, ▽: simplex, ⋆: roll, □: auto.

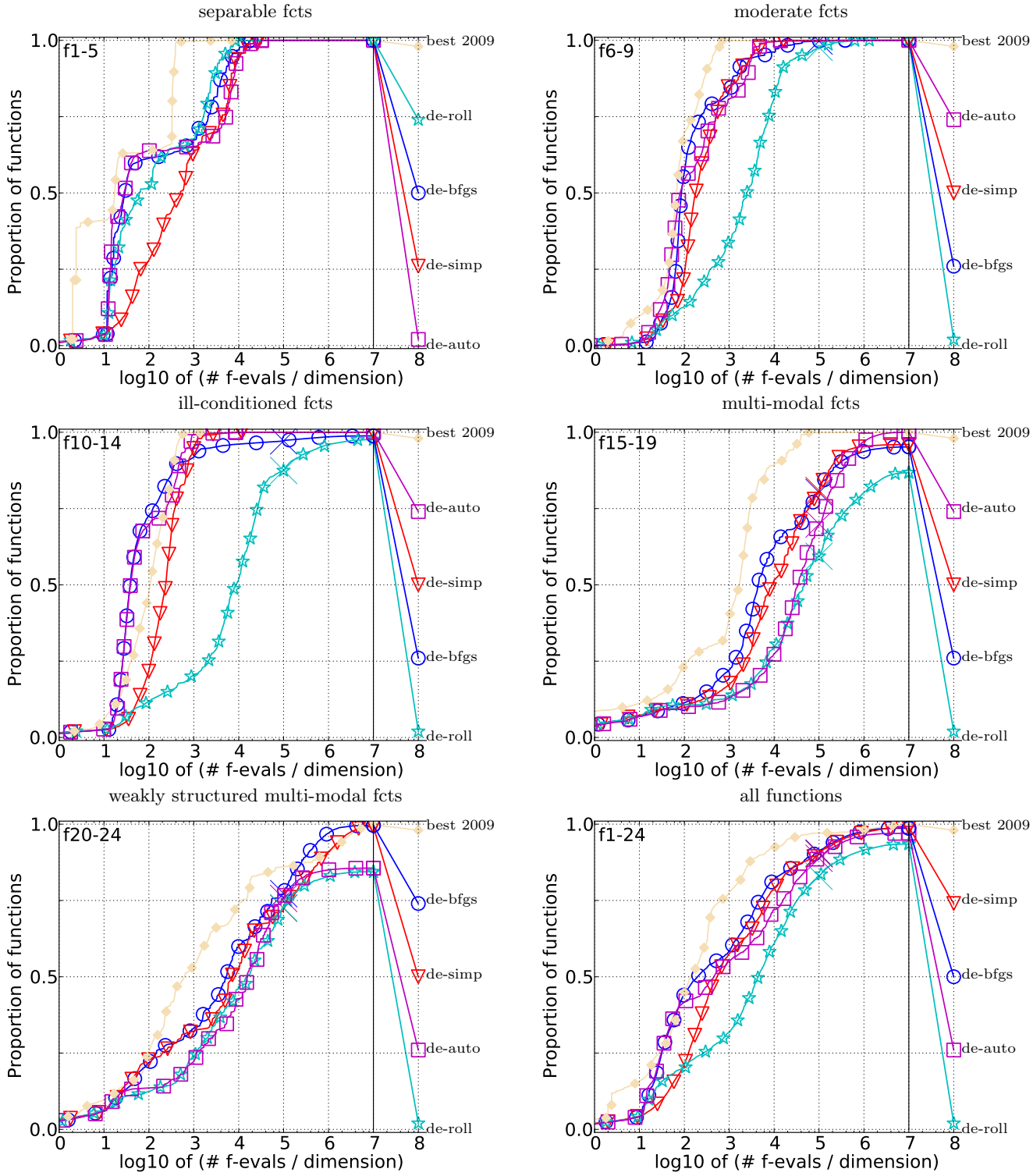

Figure 2: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/D) for 50 targets in $10^{[-8..2]}$ for all functions and subgroups in 5-D. The "best 2009" line corresponds to the best ERT observed during BBOB 2009 for each single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

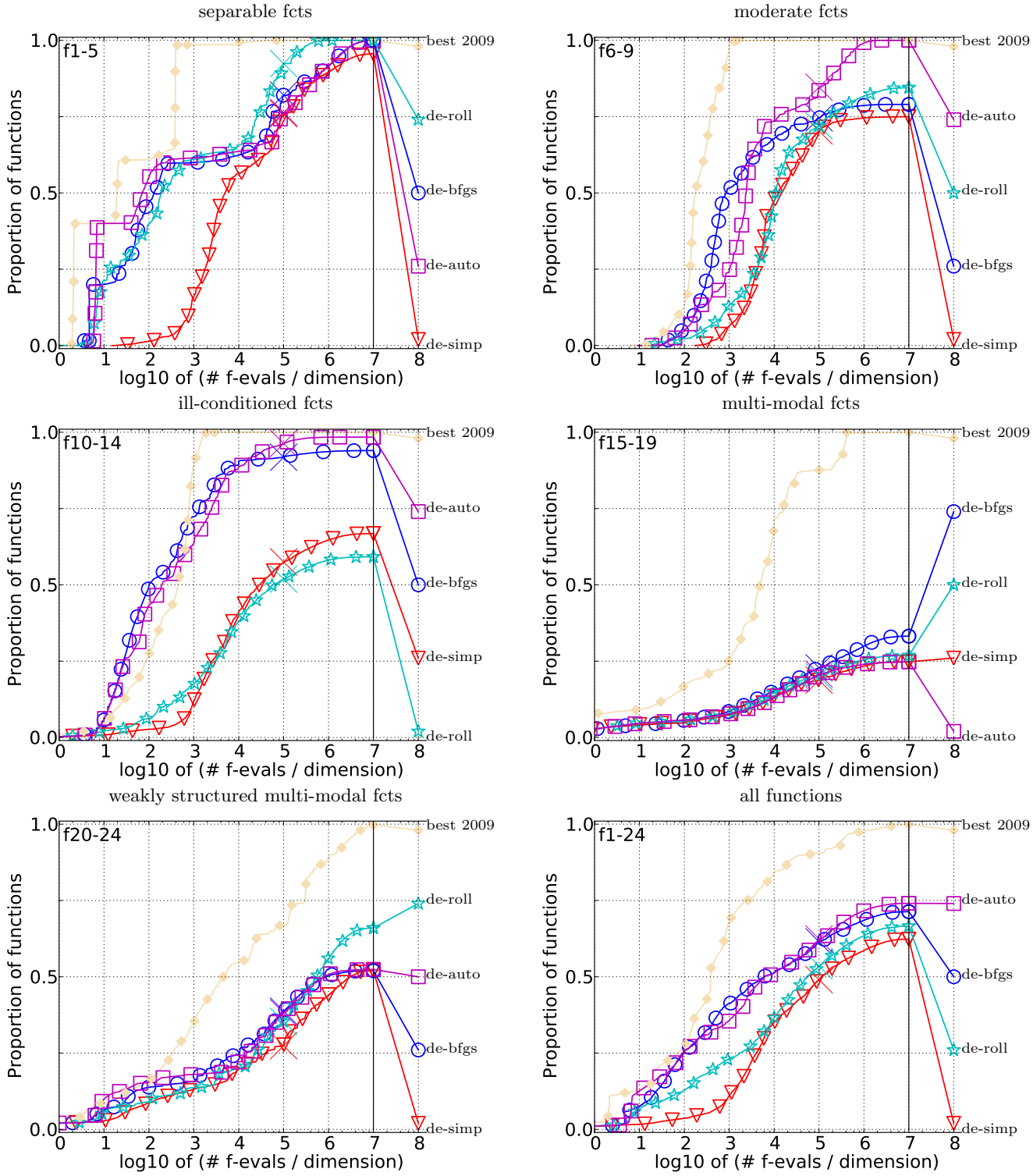

Figure 3: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/D) for 50 targets in $10^{[-8..2]}$ for all functions and subgroups in 20-D. The "best 2009" line corresponds to the best ERT observed during BBOB 2009 for each single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

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f1	11	12	12	12	12	12	15/15
bfgs	3.3(2)	**4.9**(0.2)*	**4.9**(0.2)$^{\star 3}$	**4.9**(0.2)$^{\star 4}$	**4.9**(0.2)$^{\star 4}$	**4.9**(0.2)$^{\star 4}$	15/15
simp	**2.8**(3)	6.7(2)	10(5)	15(3)	19(2)	23(3)	15/15
roll	3.3(2)	17(21)	18(21)	19(21)	19(21)	19(21)	15/15
auto	3.3(2)	5.8(0.7)	5.9(0.3)	6.0(0.2)	6.0(0.2)	6.0(0.2)	15/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f2	83	87	88	90	92	94	15/15
bfgs	**1.4**(0.3)	**1.4**(0.3)	**1.5**(0.3)	**1.7**(0.3)	**1.9**(0.3)	**2.0**(0.3)	15/15
simp	6.1(4)	10(7)	11(7)	12(7)	12(6)	12(6)	15/15
roll	5.1(6)	5.3(6)	5.4(6)	5.5(6)	5.8(6)	5.9(6)	15/15
auto	1.4(0.2)	1.4(0.2)	1.5(0.4)	1.8(0.4)	1.9(0.5)	2.1(0.5)	15/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f3	716	1622	1637	1646	1650	1654	15/15
bfgs	2.0(1)	**3.7**(3)	7.7(5)	7.6(5)	7.6(5)	7.6(5)	15/15
simp	4.1(3)	10(8)	19(13)	19(14)	19(14)	19(13)	15/15
roll	2.2(1)	4.4(2)	**6.6**(4)	**6.6**(4)	**6.6**(4)	**6.6**(4)	15/15
auto	**1.3**(1)	11(5)	21(9)	20(9)	20(9)	20(9)	15/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f4	809	1633	1688	1817	1886	1903	15/15
bfgs	3.8(2)	9.0(6)	20(17)	19(15)	18(15)	18(15)	15/15
simp	6.8(3)	14(7)	24(16)	23(14)	22(14)	22(14)	15/15
roll	**1.6**(2)	**4.7**(3)	**11**(10)	**10**(9)	**10**(9)	**10**(8)	15/15
auto	1.8(2)	16(9)	28(22)	26(21)	25(20)	25(20)	15/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f5	10	10	10	10	10	10	15/15
bfgs	6.5(2)	8.8(3)	9.0(2)	9.0(2)	9.0(2)	9.0(2)	15/15
simp	29(13)	122(119)	266(171)	324(99)	336(123)	369(138)	15/15
roll	5.9(0.3)	6.4(0.4)	6.5(0.4)	6.7(0.5)	6.9(0.5)	7.1(0.5)	15/15
auto	**5.8**(0.4)	**6.3**(0.4)	**6.4**(0.3)	**6.4**(0.3)	**6.4**(0.3)	**6.4**(0.3)	15/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f6	114	214	281	580	1038	1332	15/15
bfgs	5.1(2)	3.1(1)	2.5(0.9)	1.5(0.6)	1.3(2)	2.8(3)	15/15
simp	5.5(9)	5.6(5)	6.2(4)	5.6(5)	5.9(6)	12(12)	15/15
roll	13(19)	13(19)	19(27)	18(23)	13(20)	53(66)	14/15
auto	**1.3**(0.7)	**1.0**(0.4)	**1.00**(0.3)	**0.79**(0.4)	**0.73**(0.4)	**0.79**(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
bfgs	**8.9**(6)	**3.4**(5)	10(22)	9.2(17)	9.2(17)	41(64)	10/15
simp	11(11)	8.0(7)	**5.1**(7)	**4.7**(5)	**4.7**(5)	**4.9**(5)	15/15
roll	29(39)	11(11)	8.4(7)	8.3(5)	8.3(5)	8.5(5)	15/15
auto	40(64)	13(10)	6.9(3)	9.1(3)	9.1(3)	14(9)	15/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f8	73	273	336	391	410	422	15/15
bfgs	2.8(2)	**1.2**(0.6)	**1.1**(0.5)	**1.1**(0.4)	**1.1**(0.4)	**1.1**(0.4)	15/15
simp	1.9(0.8)	2.2(3)	2.2(2)	2.1(2)	2.2(2)	2.2(2)	15/15
roll	4.6(6)	13(10)	40(33)	83(51)	120(38)	1134(1186)	5/15
auto	**1.6**(0.4)	2.4(4)	2.1(4)	1.9(3)	1.9(3)	1.8(3)	15/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f9	35	127	214	300	335	369	15/15
bfgs	4.2(0.7)	**2.7**(2)	**1.9**(0.9)	**1.5**(0.7)	**1.3**(0.6)	**1.2**(0.5)	15/15
simp	5.8(4)	4.4(4)	3.2(2)	2.6(2)	2.5(1)	2.4(1)	15/15
roll	8.1(6)	17(10)	71(43)	124(58)	165(67)	361(334)	13/15
auto	**3.7**(1)	7.1(6)	4.5(4)	3.3(2)	3.0(2)	2.8(2)	15/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f10	349	500	574	626	829	880	15/15
bfgs	**0.75**(0.7)	**0.54**(0.4)	**0.49**(0.4)	**0.48**(0.4)	**0.39**(0.3)	**0.50**(0.9)	15/15
simp	2.9(3)	2.7(2)	2.5(1)	2.4(1)	1.9(1.0)	2.0(2)	15/15
roll	93(102)	121(86)	129(82)	176(66)	178(43)	1865(1957)	1/15
auto	2.6(4)	1.9(3)	1.6(3)	1.5(2)	1.2(2)	1.1(2)	15/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f11	143	202	763	1177	1467	1673	15/15
bfgs	**0.70**(0.2)$_{\downarrow 2}$	**0.53**(0.1)$_{\downarrow 3}$	**0.15**(0.0)$_{\downarrow 4}$	**0.11**(0.0)$_{\downarrow 4}$	**0.11**(0.0)$_{\downarrow 4}$	**0.12**(0.1)$_{\downarrow 4}$	15/15
simp	4.5(4)	5.3(3)	1.7(0.5)	1.3(0.6)	1.1(0.5)	0.97(0.4)	15/15
roll	76(76)	163(111)	63(26)	81(14)	92(14)	2107(2324)	0/15
auto	2.6(0.1)	1.9(0.1)	0.51(0.0)	0.35(0.0)	0.29(0.0)	0.61(1)	15/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f12	108	268	371	461	1303	1494	15/15
bfgs	2.8(1)	**2.0**(1)	**1.9**(1)	**1.9**(1)	**0.80**(0.5)	**1.1**(0.7)	15/15
simp	4.1(4)	3.1(2)	3.4(2)	3.4(2)	1.6(1)	2.0(2)	15/15
roll	21(19)	112(37)	148(105)	211(256)	118(148)	485(567)	1/15
auto	**2.8**(3)	2.2(2)	2.2(2)	2.2(2)	0.93(0.8)	1.2(1)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f13	132	195	250	1310	1752	2255	15/15
bfgs	1.0(0.2)	0.91(0.1)	0.87(0.1)	0.23(0.0)$_{\downarrow 4}$	2.6(1.0)	529(621)	0/15
simp	4.8(4)	6.2(5)	8.4(6)	2.3(1)	2.2(1)	2.7(2)	15/15
roll	20(18)	43(39)	53(29)	16(7)	16(4)	79(118)	12/15
auto	**0.99**(0.1)	**0.87**(0.1)$_{\downarrow 2}$	**0.83**(0.1)$_{\downarrow 4}$	**0.22**(0.0)$_{\downarrow 4}$	**1.1**(0.5)$^{\star 2}$	**0.96**(0.1)$^{\star 2}$	15/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f14	10	41	58	139	251	476	15/15
bfgs	1.0(1)	**1.8**(0.2)	**1.5**(0.3)	**0.95**(0.1)	**0.71**(0.1)	6.4(10)	13/15
simp	**0.95**(2)	5.3(4)	4.9(3)	3.1(0.9)	2.2(0.5)	**1.5**(0.3)	15/15
roll	1.4(2)	4.7(6)	4.9(4)	10(10)	112(33)	1824(2233)	0/15
auto	1.7(1)	2.0(0.3)	1.7(0.2)	0.99(0.1)	0.73(0.1)	5.1(5)	15/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f15	511	9310	19369	20073	20769	21359	14/15
bfgs	**3.8**(3)$^{\star 2}$	**1.1**(1)*	**1.6**(1)	**1.6**(1)	**1.5**(1)	**1.5**(1)	15/15
simp	10(5)	3.1(3)	3.2(3)	3.0(3)	2.9(3)	2.9(3)	15/15
roll	24(14)	7.0(6)	12(13)	12(15)	12(14)	11(12)	13/15
auto	15(9)	6.1(7)	11(14)	11(13)	11(13)	10(13)	13/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f16	120	612	2662	10449	11644	12095	15/15
bfgs	8.0(8)	17(12)	21(9)	33(48)	30(43)	585(705)	0/15
simp	**2.4**(2)	**7.6**(7)	**19**(25)	**10**(9)	**10**(8)	**37**(47)	7/15
roll	8.3(9)	69(41)	62(95)	73(81)	110(120)	616(684)	0/15
auto	6.6(15)	115(105)	99(95)	93(103)	86(87)	605(686)	1/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f17	5.2	215	899	3669	6351	7934	15/15
bfgs	3.0(3)	**7.1**(7)$^{\star 2}$	**35**(110)*	**10**(27)$^{\star 2}$	**6.9**(16)	145(165)	0/15
simp	**2.5**(2)	23(16)	55(35)	24(34)	15(20)	887(1042)	0/15
roll	2.7(3)	47(41)	125(261)	54(72)	45(42)	898(1042)	1/15
auto	3.8(3)	163(53)	120(212)	47(57)	35(32)	**41**(35)	10/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f18	103	378	3968	9280	10905	12469	15/15
bfgs	**5.7**(7)	**18**(10)$^{\star 2}$	**28**(63)	**19**(27)	**20**(24)	587(625)	0/15
simp	23(23)	155(285)	62(77)	65(69)	74(74)	575(603)	0/15
roll	49(69)	112(57)	65(66)	88(97)	∞	∞ *5e5*	0/15
auto	48(107)	162(60)	60(72)	54(58)	60(51)	**84**(81)	2/15
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
bfgs	40(38)	**2166**(1133)	71(65)	**4.3**(4)	**4.3**(3)	11(10)	2/15
simp	62(16)	2807(2948)	**68**(95)	4.7(5)	4.7(6)	**4.7**(5)	9/15
roll	**38**(26)	5195(8159)	502(489)	61(67)	61(68)	∞ *5e5*	0/15
auto	323(28)	1.3e4(1e4)	361(226)	19(19)	19(21)	19(22)	3/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f20	16	851	38111	54470	54861	55313	14/15
bfgs	4.1(2)	**2.8**(1)	10(13)	7.1(9)	7.0(7)	7.0(8)	10/15
simp	6.4(2)	4.7(5)	19(22)	13(16)	13(14)	13(14)	7/15
roll	15(19)	3.1(3)	**4.0**(6)	**2.8**(4)	**2.8**(4)	**2.8**(4)	14/15
auto	**4.0**(1)	7.2(6)	14(17)	10(11)	10(11)	10(12)	9/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f21	41	1157	1674	1705	1729	1757	14/15
bfgs	**2.2**(2)	**1.6**(2)	**2.7**(3)	**2.7**(3)	**2.7**(3)	**16**(15)	12/15
simp	3.2(2)	19(23)	19(17)	19(17)	18(17)	18(17)	15/15
roll	10(21)	33(51)	31(39)	31(39)	30(38)	30(38)	15/15
auto	10(24)	26(47)	36(72)	36(71)	35(70)	35(69)	15/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f22	71	386	938	1008	1040	1068	14/15
bfgs	6.7(15)	**13**(27)	**23**(24)	**21**(22)	**21**(22)	131(245)	8/15
simp	**6.5**(8)	29(74)	44(47)	41(44)	40(42)	**39**(41)	15/15
roll	16(24)	144(290)	150(233)	143(217)	144(215)	208(256)	8/15
auto	8.1(15)	84(101)	102(128)	95(119)	92(115)	96(103)	15/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f23	3.0	518	14249	31654	33030	34256	15/15
bfgs	**2.2**(2)	2.5(3)	2.0(2)	1.3(0.9)	1.4(0.9)	96(109)	0/15
simp	2.3(2)	**0.92**(0.9)	**0.88**(0.7)	**0.91**(0.8)	**1.1**(0.8)	99(116)	0/15
roll	2.4(2)	4.4(5)	18(22)	14(16)	16(15)	104(117)	0/15
auto	2.9(3)	4.6(4)	3.5(2)	4.4(2)	4.8(2)	**6.1**(3)	14/15
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
bfgs	**1.3**(1)*	**4.3**(5)	**0.19**(0.2)	**0.24**(0.3)	**0.18**(0.2)	**0.18**(0.2)	3/15
simp	3.5(3)	10(12)	0.56(0.6)	0.76(0.8)	0.57(0.6)	0.57(0.6)	1/15
roll	13(11)	16(17)	∞	∞	∞	∞ *5e5*	0/15
auto	10(11)	10(11)	∞	∞	∞	∞ *5e5*	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 5. The inter-80%tile 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 $f_{\text{opt}} + 10^{-8}$. 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
bfgs	**2.0**(0.2)$^{\star 4}$	**2.2**(0)$^{\star 4}$	**2.2**(0)$^{\star 4}$	**2.2**(0)$^{\star 4}$	**2.2**(0)$^{\star 4}$	**2.2**(0)$^{\star 4}$	15/15
simp	71(41)	181(83)	237(92)	389(99)	461(117)	840(800)	15/15
roll	6.4(7)	36(35)	37(36)	38(36)	38(36)	38(36)	15/15
auto	3.1(0.1)	3.1(0.1)	3.1(0.1)	3.1(0.1)	3.1(0.1)	3.1(0.1)	15/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f2	385	386	387	390	391	393	15/15
bfgs	5.7(4)	6.1(4)	6.2(4)	6.5(4)	7.1(5)	10(5)	15/15
simp	90(20)	104(16)	120(23)	135(24)	149(23)	287(246)	15/15
roll	15(12)	16(12)	16(12)	16(12)	17(12)	17(12)	15/15
auto	**3.5**(2)	**3.9**(2)	**4.3**(2)	**4.7**(2)	**5.1**(2)	**5.7**(3)	15/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f3	5066	7626	7635	7643	7646	7651	15/15
bfgs	56(24)	132(68)	181(150)	181(151)	181(150)	191(148)	13/15
simp	119(44)	212(150)	355(283)	355(283)	355(290)	484(424)	4/15
roll	**15**(11)*	**55**(30)$^{\star 2}$	**82**(65)*	**82**(65)*	**82**(65)*	**82**(65)*	15/15
auto	47(49)	224(181)	401(388)	401(309)	401(309)	401(352)	8/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f4	4722	7628	7666	7700	7758	1.4e5	9/15
bfgs	143(83)	589(540)	3893(4049)	3876(4291)	3847(4259)	212(217)	1/15
simp	172(76)	388(277)	3910(4048)	3893(3900)	3866(3871)	∞ *2e6*	0/15
roll	**26**(19)	**165**(86)*	**407**(389)*	**406**(387)*	**403**(354)*	**22**(18)*	8/15
auto	67(50)	1265(1188)	3887(4179)	3870(4290)	3841(4258)	211(241)	1/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f5	41	41	41	41	41	41	15/15
bfgs	14(8)	22(14)	24(16)	24(16)	26(20)	26(20)	15/15
simp	578(126)	869(181)	1437(334)	2145(454)	2753(766)	2.9e4(3e4)	8/15
roll	3.8(0.7)	4.0(0.9)	4.0(0.9)	4.0(0.9)	4.0(0.9)	4.0(0.9)	15/15
auto	**3.1**(0.1)	**3.1**(0.1)	**3.1**(0.1)	**3.1**(0.1)	**3.1**(0.1)	**3.1**(0.1)	15/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f6	1296	2343	3413	5220	6728	8409	15/15
bfgs	6.8(5)	8.5(6)	17(16)	32(36)	96(154)	295(334)	2/15
simp	34(13)	34(10)	42(23)	102(61)	329(184)	∞ *2e6*	0/15
roll	21(7)	23(35)	23(24)	69(35)	209(246)	1025(1190)	1/15
auto	**1.9**(0.7)$^{\star 2}$	**2.2**(2)$^{\star 2}$	**4.3**(3)*	**12**(3)*	**16**(3)$^{\star 2}$	**71**(120)	10/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f7	1351	4274	9503	16524	16524	16969	15/15
bfgs	502(766)	∞	∞	∞	∞	∞ *2e6*	0/15
simp	125(107)	1278(1235)	∞	∞	∞	∞ *2e6*	0/15
roll	**26**(34)	**259**(249)	548(571)	∞	∞	∞ *2e6*	0/15
auto	28(11)	577(588)	**522**(571)	**439**(414)	**439**(414)	**427**(452)	4/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f8	2039	3871	4040	4219	4371	4484	15/15
bfgs	**1.5**(1)	**2.8**(4)	**2.8**(4)	**3.2**(3)*	**3.2**(3)$^{\star 2}$	**3.2**(3)$^{\star 3}$	15/15
simp	12(8)	13(8)	15(8)	19(8)	24(10)	81(56)	14/15
roll	6.4(8)	15(16)	20(19)	28(21)	33(20)	129(117)	13/15
auto	1.5(0.4)	4.2(3)	5.2(5)	8.2(5)	9.3(4)	12(5)	15/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f9	1716	3102	3277	3455	3594	3727	15/15
bfgs	**1.8**(1)	**2.6**(2)*	**2.8**(1)	**2.7**(1)$^{\star 3}$	**2.7**(1)$^{\star 3}$	**2.7**(1)$^{\star 3}$	15/15
simp	24(8)	149(212)	144(198)	140(188)	142(179)	207(287)	13/15
roll	25(16)	44(18)	54(26)	63(23)	83(44)	1299(1589)	1/15
auto	2.0(0.9)	6.9(5)	8.8(6)	11(5)	11(4)	20(30)	15/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f10	7413	8661	10735	14920	17073	17476	15/15
bfgs	6.1(6)	5.3(5)	4.3(4)	3.1(3)	**2.7**(3)*	**2.6**(3)$^{\star 3}$	15/15
simp	138(105)	817(809)	∞	∞	∞	∞ *2e6*	0/15
roll	∞	∞	∞	∞	∞	∞ *2e6*	0/15
auto	**1.5**(1)	**1.5**(1)	**1.8**(2)	**2.5**(2)	4.6(2)	17(11)	9/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f11	1002	2228	6278	9762	12285	14831	15/15
bfgs	**0.20**(0.1)$_{\downarrow 4}$	**0.11**(0.1)$_{\downarrow 4}$	**0.05**(0.0)$_{\downarrow 4}$	**0.04**(0.0)$_{\downarrow 4}$	**0.04**(9e-3)$_{\downarrow 4}$	**0.05**(0.0)$_{\downarrow 4}$	15/15
simp	90(38)	162(67)	142(53)	1512(1620)	∞	∞ *2e6*	0/15
roll	510(225)	6543(6688)	∞	∞	∞	∞ *2e6*	0/15
auto	0.38(0.5)	0.19(0.2)$_{\downarrow 3}$	0.07(0.1)$_{\downarrow 4}$	0.06(0.1)$_{\downarrow 4}$	0.05(0.0)$_{\downarrow 4}$	0.10(0.0)$_{\downarrow 4}$	15/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f12	1042	1938	2740	4140	12407	13827	15/15
bfgs	**1.4**(0.8)	**1.4**(1)	**1.6**(1)	**1.7**(1)*	**1.1**(0.6)$^{\star 3}$	**4.6**(4)$^{\star 2}$	15/15
simp	34(31)	30(23)	28(17)	31(17)	21(17)	621(670)	0/15
roll	21(35)	18(24)	18(17)	27(20)	25(20)	100(103)	6/15
auto	2.2(2)	2.3(3)	2.9(3)	4.4(4)	4.5(2)	39(49)	7/15

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f13	652	2021	2751	18749	24455	30201	15/15
bfgs	**1.3**(0.1)$^{\star 2}$	**0.59**(0.1)$^{\star 2}_{\downarrow 4}$	**0.53**(0.0)$^{\star 2}_{\downarrow 4}$	**6.18**(0.2)$^{\star 2}_{\downarrow 4}$	92(123)	∞ *2e6*	0/15
simp	28(17)	15(7)	16(6)	9.0(6)	195(213)	∞ *2e6*	0/15
roll	24(15)	18(20)	17(15)	5.8(4)	24(16)	289(299)	1/15
auto	1.8(0.5)	0.91(0.4)	0.95(0.3)	0.70(0.6)	**4.5**(3)	**32**(36)$^{\star 2}$	7/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f14	75	239	304	932	1648	15661	15/15
bfgs	**1.7**(0.4)	**0.84**(0.1)	**0.90**(0.2)	0.62(0.1)$_{\downarrow 4}$	**0.56**(0.1)$^{\star 2}_{\downarrow 4}$	25(141)	0/15
simp	12(10)	40(32)	53(17)	30(9)	122(52)	∞ *2e6*	0/15
roll	7.4(14)	18(21)	17(16)	33(7)	3481(3085)	∞ *2e6*	0/15
auto	2.0(0.3)	0.92(0.1)	0.93(0.1)	**0.58**(0.0)$_{\downarrow 4}$	0.91(0.0)$_{\downarrow 3}$	**62**(67)	0/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f15	30378	1.5e5	3.1e5	3.2e5	4.5e5	4.6e5	15/15
bfgs	28(15)	∞	∞	∞	∞	∞ *2e6*	0/15
simp	**25**(11)	∞	∞	∞	∞	∞ *2e6*	0/15
roll	32(19)	∞	∞	∞	∞	∞ *2e6*	0/15
auto	59(46)	∞	∞	∞	∞	∞ *2e6*	0/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f16	1384	27265	77015	1.9e5	2.0e5	2.2e5	15/15
bfgs	595(508)	**242**(257)	∞	∞	∞	∞ *2e6*	0/15
simp	54(37)	∞	∞	∞	∞	∞ *2e6*	0/15
roll	**20**(18)	1093(1138)	∞	∞	∞	∞ *2e6*	0/15
auto	40(46)	∞	∞	∞	∞	∞ *2e6*	0/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f17	63	1030	4005	30677	56288	80472	15/15
bfgs	17(19)	**117**(48)	**327**(297)	**952**(1060)	∞	∞ *2e6*	0/15
simp	34(40)	381(238)	2127(2524)	∞	∞	∞ *2e6*	0/15
roll	**15**(17)	358(276)	535(600)	∞	∞	∞ *2e6*	0/15
auto	48(68)	335(185)	529(334)	∞	∞	∞ *2e6*	0/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f18	621	3972	19561	67569	1.3e5	1.5e5	15/15
bfgs	**69**(61)	**350**(344)	1503(1689)	∞	∞	∞ *2e6*	0/15
simp	115(81)	1124(1266)	**1462**(1586)	∞	∞	∞ *2e6*	0/15
roll	87(28)	518(750)	1470(1740)	∞	∞	∞ *2e6*	0/15
auto	131(69)	416(376)	∞	∞	∞	∞ *2e6*	0/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f19	1	1	3.4e5	6.2e6	6.7e6	6.7e6	15/15
bfgs	3206(3658)	**3.8e4**(2e4)$^{\star 2}$	**5.4**(6)$^{\star 3}$	∞	∞	∞ *2e6*	0/15
simp	8348(6218)	1.5e5(5e4)	∞	∞	∞	∞ *2e6*	0/15
roll	3941(5048)	2.2e5(3e5)	∞	∞	∞	∞ *2e6*	0/15
auto	**2751**(2816)	1.1e5(8e4)	87(96)	∞	∞	∞ *2e6*	0/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f20	82	46150	3.1e6	5.5e6	5.6e6	5.6e6	14/15
bfgs	6.3(4)	1.0(1.0)	∞	∞	∞	∞ *2e6*	0/15
simp	21(27)	3.6(3)	∞	∞	∞	∞ *2e6*	0/15
roll	21(18)	**0.84**(0.8)	**4.4**(5)*	**5.3**(5)*	**5.3**(6)*	**5.2**(6)*	1/15
auto	**2.1**(0.5)*	3.4(2)	∞	∞	∞	∞ *2e6*	0/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f21	561	6541	14103	14643	15567	17589	15/15
bfgs	**10**(11)	**33**(56)	**17**(26)	**16**(25)	**15**(24)	205(233)	1/15
simp	89(131)	208(212)	159(212)	154(207)	145(159)	196(234)	5/15
roll	84(121)	66(79)	66(87)	64(84)	60(79)	53(67)	12/15
auto	50(111)	52(72)	37(48)	36(47)	34(44)	**30**(39)	14/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f22	467	5580	23491	24948	26847	1.3e5	12/15
bfgs	**5.7**(13)	**135**(189)	**151**(143)	**142**(139)	**132**(125)	∞ *2e6*	0/15
simp	38(81)	338(473)	1194(1300)	1125(1244)	1046(1193)	**208**(238)	0/15
roll	68(139)	199(199)	396(416)	374(401)	348(373)	221(238)	0/15
auto	128(88)	234(273)	281(298)	265(288)	246(266)	216(226)	0/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f23	3.2	1614	67457	4.9e5	8.1e5	8.4e5	15/15
bfgs	2.1(2)	7.0(7)	54(59)	∞	∞	∞ *2e6*	0/15
simp	**1.6**(1)	**1.3**(0.9)$^{\star 2}$	**36**(43)	∞	∞	∞ *2e6*	0/15
roll	1.9(2)	15(14)	∞	∞	∞	∞ *2e6*	0/15
auto	2.2(2)	3.8(3)	36(45)	∞	∞	∞ *2e6*	0/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f24	1.3e6	7.5e6	5.2e7	5.2e7	5.2e7	5.2e7	3/15
bfgs	**4.3**(5)	∞	∞	∞	∞	∞ *2e6*	0/15
simp	22(25)	∞	∞	∞	∞	∞ *2e6*	0/15
roll	4.7(5)	∞	∞	∞	∞	∞ *2e6*	0/15
auto	∞	∞	∞	∞	∞	∞ *2e6*	0/15

Table 3: 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 inter-80%tile range divided by two is given in braces. The median number of conducted function evaluations is additionally given in *italics*, if $\text{ERT}(10^{-7}) = \infty$. #succ is the number of trials that reached the final target $f_{\text{opt}} + 10^{-8}$. Best results are printed in bold.