Benchmarking Cellular Genetic Algorithms on the BBOB Noiseless Testbed

Neal Holtschulte
Dept. of Computer Science
University of New Mexico
Albuquerque, NM 87131
+1 (505) 277-8432
neal.holts@cs.unm.edu

Melanie Moses
Dept. of Computer Science
University of New Mexico
Albuquerque, NM 87131
+1 (505) 277-9140
melaniem@cs.unm.edu

ABSTRACT

In this paper we evaluate 2 cellular genetic algorithms (CGAs), a single-population genetic algorithm, and a hill-climber on the Black Box Optimization Benchmarking testbed. CGAs are fine grain parallel genetic algorithms with a spatial structure imposed by embedding individuals in a connected graph. They are popular for their diversity-preserving properties and efficient implementations on parallel architectures. We find that a CGA with a uni-directional ring topology outperforms the canonical CGA that uses a bi-directional grid topology in nearly all cases. Our results also highlight the importance of carefully chosen genetic operators for finding precise solutions to optimization problems.

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

1. INTRODUCTION

Parallel genetic algorithms (PGAs) are genetic algorithms in which the population is divided into semi-isolated subpopulations. PGAs take advantage of the speed afforded by parallel or multicore architectures. The isolation of individuals in different subpopulations has been shown to be advantageous even when running on a single CPU [3, 10, 8].

Coarse grain PGAs divide their population amongst few subpopulations. Fine grain PGAs divide it amongst many. Fine grain PGAs maintain diversity better than coarse grain PGAs, but pay a steep communication cost when they are scaled up to a large number of subpopulations [10].

Cellular genetic algorithms (CGAs), in which each individual is its own subpopulation, are the most fine of the fine grain PGAs. A spatial structure is imposed on CGAs by locating each individual on a sparse, connected graph. Individuals crossover with others from a small, connected neighborhood. The sparsity of the graph and lack of global communication allows CGAs to scale more efficiently than other fine grain PGAs.

Mühlenbein and Gorges-Schleuter introduced one of the earliest cellular genetic algorithms in 1989 with an asynchronous parallel genetic algorithm called ASPARAGOS. The algorithm uses a ladder structure where each individual has 3 neighbors at one Manhattan distance away from itself. ASPARAGOS was shown to be effective at solving traveling salesman and quadratic assignment problems. [5, 9]

The most common graph structure for cellular GAs is a two-dimensional grid with wrapping edges such that each individual has 4 neighbors, one in each cardinal direction. This structure mimics the topology of interconnected processors common to many parallel systems [10].

In addition to scalability and efficiency on GPUs, CGAs preserve diversity and avoid premature convergence because individuals in a CGA are isolated by distance and the best solutions in the population spread gradually from neighborhood to neighborhood. CGAs emphasize the exploration side of the exploration/exploitation tradeoff. [1, 5]

We benchmark and compare 2 CGA variants, a single-population GA, and a hill-climbing algorithm. Comparisons to a single-population GA benchmarked by Tran and Jin [12] are also discussed. Data and source code from these experiments can be found on the GECCO Black-Box Optimization Benchmarking (BBOB) 2013 webpage.

2. ALGORITHMS

The canonical CGA (*grid*) is implemented as described by Alba [1]. Individuals are laid out on a two-dimensional, toroidal grid. Each individual has 4 neighbors. We use the North-East-West-South, or "NEWS", neighborhood. With 90% probability, crossover occurs between an individual and another individual selected from its neighborhood by rank selection. The resulting children are mutated with some probability and then the best individual out of the first parent and two children replaces the first parent. Pseudocode for the canonical CGA is shown in Figure 1.

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

```
GenerateInitialPopulation(cga.pop);
Evaluation(cga.pop);
while ! StopCondition() do
  for individual ← to cga.popSize do
    neighbors ← CalculateNeighborhood(cga, position(Individual));
    parents ← Selection(neighbors);
    offspring ← Recombination(cga.Pc, parents);
    offspring ← Mutation(cga.Pm, offspring);
    Evaluation(offspring);
    Replacement(position(individual), auxiliary_pop, offspring);
  end for
  cga.pop ← auxiliary_pop;
end while
```

Figure 1: The above pseudocode for the canonical genetic algorithm is duplicated from [1].

The second CGA evaluated on the benchmarks differs from the canonical CGA only in its neighborhood. We implement a one-directional, ring CGA (*ring*) in which each individual has one neighbor. The selection of a mate is deterministic since there is only one other individual in each neighborhood.

A generational, single-population genetic algorithm using rank selection (*ga*) is implemented to test whether CGAs are superior to single-population GAs.

A hill-climber (*hill*) is also benchmarked for comparison. Hill-climbers have a population of one and take steps along the fitness landscape. Our hill-climber uses the same mutation operator as the GA and CGAs for its step function. Our hill-climber does not restart if it reaches a local optimum.

3. EXPERIMENTAL DESIGN

Both CGAs update synchronously. The CGAs, GA, and hill-climber use a per-gene mutation rate of 1/dimensionality such that one mutation occurs per individual per generation on average. Two point crossover with a crossover rate of 90% is used for the GA and CGAs.

All the algorithms we implement use a Gaussian mutation operator. The Gaussian mutation operator replaces a value, x, in an individual with a value selected from a Gaussian distribution with mean x and variance 2. 2 is 20% of the range of a gene since genes range from -5 to 5. A smaller variance would result in more localized search. Algorithms benchmarked with a uniform mutation operator are included in the source code and data associated with this paper, which is available on the BBOB website, but are not included in this paper due to their poor performance.

We benchmark each of the CGAs with three different population sizes: 100, 49, and 16. These values are used because the individuals can be laid out in a square grid. These values and the neighborhood differences between ring and grid CGAs are the only experimentally varied parameters. The results for population size 49 are omitted from the paper, but included in the associated data. The GA is benchmarked with a population size of 100.

None of our algorithms restart. The number of function evaluations is limited to $50,000 * D$ where D is the number of dimensions. This limit is the same as the limit used by other researchers on this benchmark set [11, 12].

4. RESULTS

Results from experiments according to [6] on the benchmark functions given in [4, 7] are presented in Figures 3, 4 and 5 and in Tables 1 and 2.

Uni-directional, one-dimensional "ring" CGAs (*ring*) outperform the canonical bi-directional, two-dimensional CGA (*grid*) with very few exceptions, such as the f8 and f19 benchmarks, for which *grid* with population size 16 is competitive with *ring*. Furthermore, the population 16 *grid* outperforms *grid* with population 49 and 100. Since the only difference between *ring* and *grid* is the spatial structure of the populations, these results suggest that the canonical CGA struggles to diffuse superior solutions through the population. Such diffusion occurs faster with a smaller population. Since the canonical CGA uses neighborhoods with size greater than one and rank selection to choose which neighbor to crossover, inferior neighbors can be selected, further slowing the diffusion of superior solutions. Future work could test the hypothesis that slow diffusion of superior solutions hampers the canonical CGA by using an elitist selection scheme.

Population size has less of an impact on the ring CGA than it has on the grid CGA. The population size 100 ring algorithm (*ring*100) outperforms all others on the weakly-structured multi-modal functions in Figure 5, but is outperformed by *ring*16 on all multi-modal functions in 5 dimensions (Figure 4). In all other cases, the difference between *ring*100 and *ring*16 is small.

The genetic algorithm with population size 100, *ga*100, is superior to or competitive with the grid CGAs. *Ga*100 is inferior to or competitive with the ring CGAs. This is a surprising result since CGAs are generally considered to be superior to single-population GAs.

Figure 3 shows our algorithms reaching the maximum function evaluation limit before finding a solution within 10^{-3} of most benchmark problems. The algorithms scale quadratically with respect to dimensionality on all benchmarks except f2 through f5, on which they scale linearly.

Tables 1 and 2 along with Figures 4 and 5 suggest that while separable problems are amenable to hill-climbing, the *hill* algorithm has difficulty getting within 10^{-7} of the final solution. We suspect that the unchanging variance of the Gaussian mutation operator made it difficult for the hill-climber (and the CGAs as well) to close the distance to the optimal solution for these benchmarks.

The testbed format permits easy comparison of algorithms. We compare our algorithms to Tran and Jin's Real-Coded GA (*rcga*) [12], but do not include their results in this paper due to space constraints.

Rcga outperforms *hill*, *ga*, and the CGAs we implement on most of the benchmarks, some notable exceptions being the weakly-structured, multi-modal functions f20, f21, and f22, on which the CGAs outperform *rcga*. It may be that the diversity-preserving properties of the CGAs improve search by emphasizing exploration over exploitation on these difficult landscapes that exhibit weak global structure and have many local optima. However, the superior performance of *rcga* on most other functions suggests that the non-uniform mutation operator and arithmetical crossover operator *rcga* uses are superior to the operators our algorithms use for many benchmarks. Non-uniform mutation uses a variable step size such that the magnitude of mutation tends to decay over time. This results in increasingly local search as

Dimensions:	2	3	5	10	20	40
grid100	0.99	0.99	1.0	1.0	1.0	1.1
ring100	0.75	0.75	0.76	0.77	0.80	0.84
ga100	1.6	1.6	1.6	1.6	1.6	1.7
hill	0.56	0.56	0.57	0.57	0.60	0.63

Figure 2: The average CPU time per function evaluation for 4 algorithms are shown. All values are 10^{-4} seconds. Results are obtained from running the algorithms on the f8 benchmark until at least 30 seconds have elapsed or a maximum number of function evaluations is reached. Timing experiments were run on an Intel Xeon W3550 processor running at 3.07GHz under Ubuntu 12.04.2 LTS.

time progresses, allowing algorithms using such an operator to close in on optimal values [2]. Future work can test whether using arithmetic crossover and non-uniform mutation, as *rcga* does, in a ring CGA, can further improve the performance of the ring CGA.

5. CONCLUSION

Cellular GAs are a popular solution to the scaling problems faced by Fine Grain PGAs. The toroidal grid structure of the canonical CGA reflects underlying architectures such as GPUs. However, CGAs with uni-directional ring topologies demonstrate faster convergence and a superior final solution compared to the canonical CGA on all benchmark functions. The canonical CGA with a population size of 16 was superior to, or competitive with, both its larger population counterparts, but population size has less effect on the ring CGA. We posit that rank selection should be replaced with a more elitist selection scheme to improve the performance of the canonical CGA by facilitating more rapid spread of high quality solutions through the population.

Additionally, we find that hill-climbing algorithms are robust and effective for solving some simple benchmark functions provided that the right step operator is chosen. The hill-climber exhibits rapid convergence and competitive final solutions for separable functions, even in higher dimensions.

A standard, single-population GA implementation is surprisingly competitive with the CGAs, though it typically has slightly worse performance than the ring CGA. This suggests that the superior performance of parallel GAs, when run on sequential CPUs, may be overstated in the literature.

Though none of the algorithms presented were competitive with the best 2009 optimization algorithm, non-uniform mutation and arithmetic crossover could greatly improve CGA performance. Our results also show that ring CGAs perform better than the more common grid CGAs on these benchmarks.

6. ACKNOWLEDGEMENT

The authors would like to thank the Black Box Optimization Benchmarking team and the GECCO Workshop for Real-Parameter Optimization organizers for providing the benchmark suite and analysis tools that made this paper possible. This work is supported by DARPA CRASH P-1070-113237 and NSF EF 1038682.

7. REFERENCES

[1] E. Alba and B. Dorronsoro. *Cellular genetic algorithms*, volume 42. Springer, 2008.

[2] K. J. Austin and P. A. Jacobs. An adaptive range mutation operator for real-coded genetic algorithms. *Evolutionary Computation*, 9, 2001.

[3] E. Cantu-Paz. A summary of research on parallel genetic algorithms, 1995.

[4] 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.

[5] M. Gorges-Schleuter. Asparagos a parallel genetic algorithm and population genetics. In J. Becker, I. Eisele, and F. Mündemann, editors, *Parallelism, Learning, Evolution*, volume 565 of *Lecture Notes in Computer Science*, pages 407–418. Springer Berlin Heidelberg, 1991.

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

[7] 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.

[8] F. Herrera, M. Lozano, and C. Moraga. Hybrid distributed real-coded genetic algorithms. In A. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature âĂŤ PPSN V*, volume 1498 of *Lecture Notes in Computer Science*, pages 603–612. Springer Berlin Heidelberg, 1998.

[9] H. Mühlenbein. Parallel genetic algorithms, population genetics and combinatorial optimization. In J. Becker, I. Eisele, and F. Mündemann, editors, *Parallelism, Learning, Evolution*, volume 565 of *Lecture Notes in Computer Science*, pages 398–406. Springer Berlin Heidelberg, 1991.

[10] M. Nowostawski and R. Poli. Parallel genetic algorithm taxonomy. In *Knowledge-Based Intelligent Information Engineering Systems, 1999. Third International Conference*, pages 88–92, 1999.

[11] P. Pošík and V. Klemš. Jade, an adaptive differential evolution algorithm, benchmarked on the bbob noiseless testbed. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference companion*, GECCO Companion '12, pages 197–204, New York, NY, USA, 2012. ACM.

[12] T.-D. Tran and G.-G. Jin. Real-coded genetic algorithm benchmarked on noiseless black-box optimization testbed. In *Proceedings of the 12th annual conference companion on Genetic and evolutionary computation*, GECCO '10, pages 1731–1738, New York, NY, USA, 2010. ACM.

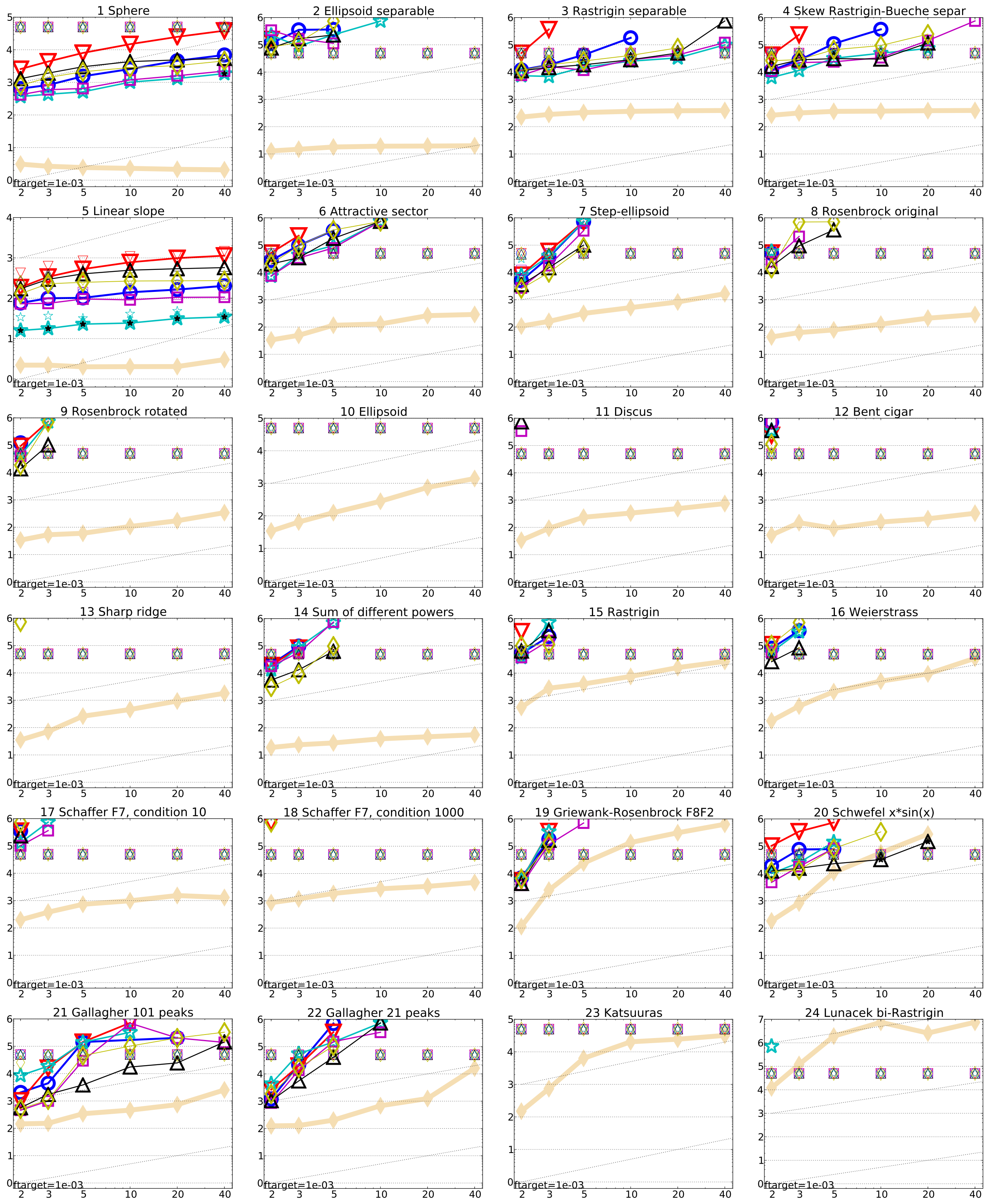

Figure 3: Expected running time (ERT in number of f-evaluations) divided by dimension for target function value 10^{-3} as $\log_{10}$ values versus dimension. Different symbols correspond to different algorithms given in the legend of f_1 and f_{24}. Light symbols give the maximum number of function 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: ○:grid16, ▽:grid100, ⋆:hill, □:ring16, △:ring100, ◇:ga100

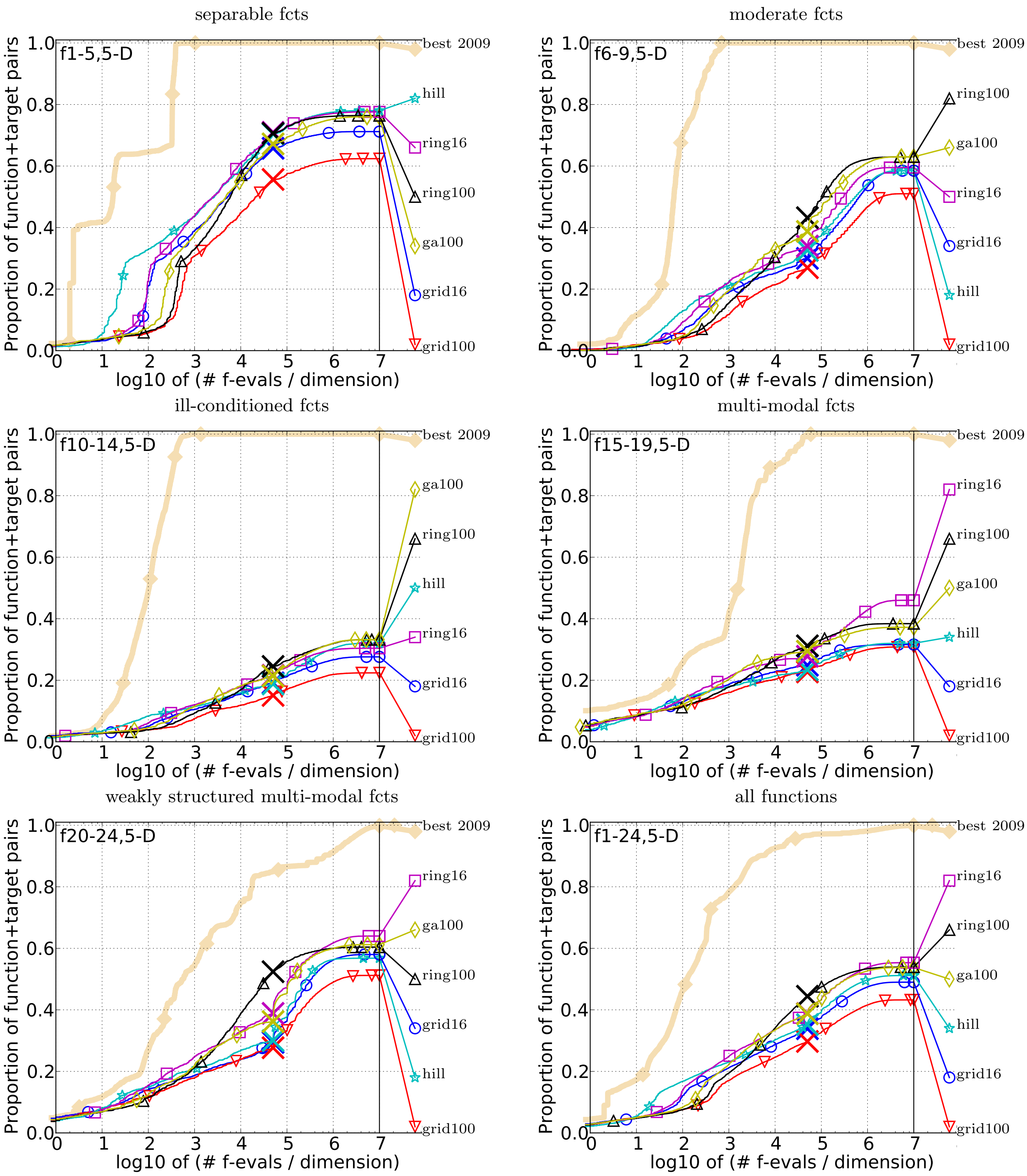

Figure 4: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/D) for 50 targets in $10^{[-8..2]}$ for all functions and subgroups in 5-D. The "best 2009" line corresponds to the best ERT observed during BBOB 2009 for each single target.

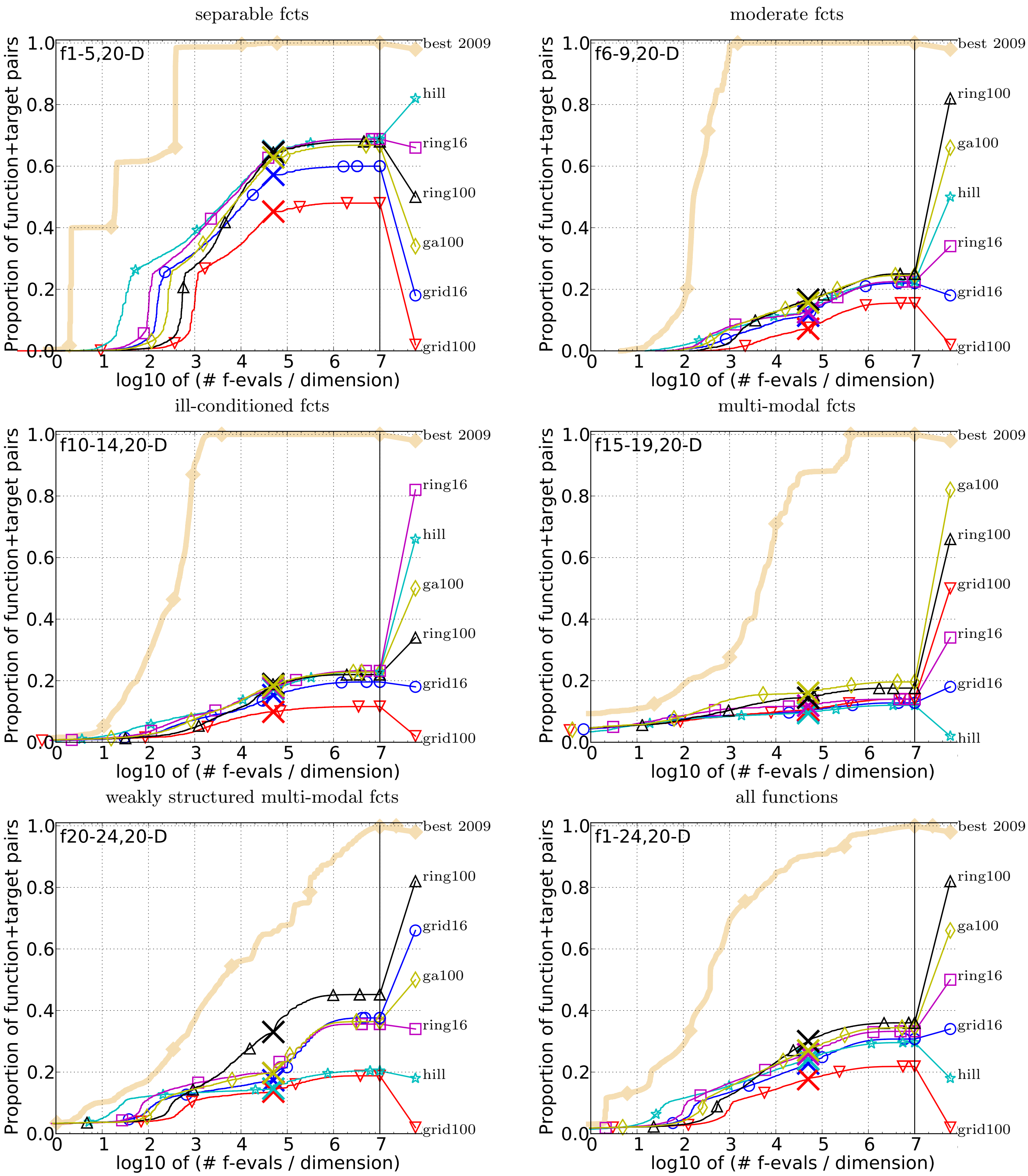

Figure 5: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/D) for 50 targets in $10^{[-8..2]}$ for all functions and subgroups in 20-D. The "best 2009" line corresponds to the best ERT observed during BBOB 2009 for each single target.

$\Delta f_{\rm opt}$	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f1	11	12	12	12	12	12	15/15
grid16	11(12)	37(26)	98(45)	631(295)	5962(2179)	3.0e5(3e5)	0/15
grid100	16(27)	152(35)	451(163)	3279(2154)	7.3e4(7e4)	∞ *2e5*	0/15
hill	**4.3**(3)	**8.3**(6)$^{\star 2}$	**22**(8)$^{\star 3}$	**210**(108)	2623(1297)	**3.8e4**(3e4)	0/15
ring16	6.7(7)	27(11)	66(16)	266(118)	**2051**(1062)	4.6e4(4e4)	0/15
ring100	12(13)	108(46)	315(77)	1235(513)	4221(940)	4.7e4(4e4)	0/15
ga100	6.1(4)	70(32)	173(67)	868(246)	5195(1902)	1.5e5(2e5)	0/15

$\Delta f_{\rm opt}$	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f2	83	87	88	90	92	94	15/15
grid16	127(144)	347(338)	943(867)	2.0e4(2e4)	∞	∞ *2e5*	0/15
grid100	320(259)	1245(1243)	4897(4892)	∞	∞	∞ *2e5*	0/15
hill	94(73)	301(211)	1204(1444)	1.3e4(1e4)	∞	∞ *2e5*	0/15
ring16	**68**(39)	250(171)	**713**(703)	**6309**(6237)	∞	∞ *2e5*	0/15
ring100	137(70)	**240**(109)	1453(1546)	1.3e4(1e4)	∞	∞ *2e5*	0/15
ga100	88(71)	293(213)	2069(2850)	4.1e4(4e4)	∞	∞ *2e5*	0/15

$\Delta f_{\rm opt}$	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f3	716	1622	1637	1646	1650	1654	15/15
grid16	1.1(0.5)	4.2(2)	11(5)	129(113)	∞	∞ *2e5*	0/15
grid100	5.3(4)	19(7)	56(21)	∞	∞	∞ *2e5*	0/15
hill	**0.36**(0.2)$^{\star 3}$	**1.4**(1.0)	**4.6**(3)	46(19)	∞	∞ *2e5*	0/15
ring16	0.98(0.3)	1.9(0.7)	5.7(4)	**36**(22)	∞	∞ *2e5*	0/15
ring100	4.4(1)	7.4(2)	16(4)	56(21)	∞	∞ *2e5*	0/15
ga100	2.3(0.6)	4.1(2)	11(5)	77(25)	**2257**(2348)	∞ *2e5*	0/15

$\Delta f_{\rm opt}$	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f4	809	1633	1688	1817	1886	1903	15/15
grid16	1.6(0.6)	5.4(3)	17(7)	302(284)	∞	∞ *2e5*	0/15
grid100	7.2(3)	24(11)	58(37)	∞	∞	∞ *2e5*	0/15
hill	**0.48**(0.2)$^{\star 3}$	**2.1**(2)	7.8(4)	88(75)	∞	∞ *2e5*	0/15
ring16	1.1(0.4)	2.7(1)	**7.0**(5)	**66**(47)	∞	∞ *2e5*	0/15
ring100	5.0(1)	10(2)	19(5)	86(42)	∞	∞ *2e5*	0/15
ga100	2.5(0.7)	5.9(3)	15(5)	181(167)	∞	∞ *2e5*	0/15

$\Delta f_{\rm opt}$	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f5	10	10	10	10	10	10	15/15
grid16	28(11)	46(14)	49(12)	51(14)	51(14)	51(14)	15/15
grid100	116(69)	247(102)	264(76)	264(76)	264(76)	264(76)	15/15
hill	**7.6**(4)$^{\star 3}$	**11**(5)$^{\star 4}$	**11**(4)$^{\star 4}$	**11**(4)$^{\star 4}$	**11**(4)$^{\star 4}$	**11**(4)$^{\star 4}$	15/15
ring16	20(10)	41(14)	47(12)	48(12)	48(12)	48(12)	15/15
ring100	94(31)	171(26)	190(42)	201(42)	201(42)	201(42)	15/15
ga100	55(19)	100(12)	121(27)	124(27)	124(27)	124(27)	15/15

$\Delta f_{\rm opt}$	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f6	114	214	281	580	1038	1332	15/15
grid16	8.3(7)	47(44)	400(475)	2955(3225)	∞	∞ *2e5*	0/15
grid100	29(22)	253(190)	964(889)	∞	∞	∞ *2e5*	0/15
hill	**2.8**(1)	13(11)	192(348)	831(966)	∞	∞ *2e5*	0/15
ring16	4.7(4)	**11**(8)	**46**(45)	**682**(851)	∞	∞ *2e5*	0/15
ring100	17(10)	53(24)	136(76)	1526(1520)	∞	∞ *2e5*	0/15
ga100	11(7)	31(17)	104(86)	3094(3323)	∞	∞ *2e5*	0/15

$\Delta f_{\rm opt}$	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f7	24	324	1171	1572	1572	1597	15/15
grid16	13(10)	97(110)	401(470)	2321(2624)	2321(2385)	2299(2270)	1/15
grid100	40(39)	74(83)	266(294)	2287(2544)	2287(2544)	2252(2466)	1/15
hill	20(26)	117(200)	300(371)	2330(2703)	2330(2624)	2294(2427)	1/15
ring16	**13**(8)	104(276)	436(539)	1075(1272)	1075(1269)	1059(1172)	2/15
ring100	28(27)	19(11)	**42**(40)	319(318)	319(372)	322(334)	5/15
ga100	21(20)	**11**(10)	58(108)	**271**(329)	**271**(321)	**269**(323)	6/15

$\Delta f_{\rm opt}$	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f8	73	273	336	391	410	422	15/15
grid16	33(20)	631(917)	2092(2619)	∞	∞	∞ *2e5*	0/15
grid100	113(86)	1140(1405)	∞	∞	∞	∞ *2e5*	0/15
hill	**8.6**(6)*	721(920)	1515(1862)	∞	∞	∞ *2e5*	0/15
ring16	17(8)	623(917)	3002(3709)	∞	∞	∞ *2e5*	0/15
ring100	58(11)	**141**(66)	**559**(731)	**4631**(4795)	∞	∞ *2e5*	0/15
ga100	34(20)	487(915)	4864(5945)	9180(1e4)	∞	∞ *2e5*	0/15

$\Delta f_{\rm opt}$	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f9	35	127	214	300	335	369	15/15
grid16	57(42)	1.3e4(1e4)	1.6e4(2e4)	∞	∞	∞ *2e5*	0/15
grid100	315(218)	1.3e4(1e4)	1.7e4(2e4)	∞	∞	∞ *2e5*	0/15
hill	**13**(11)$^{\star 2}$	4153(4291)	∞	∞	∞	∞ *2e5*	0/15
ring16	30(11)	2428(2611)	∞	∞	∞	∞ *2e5*	0/15
ring100	138(47)	**1261**(1979)	**4949**(5841)	∞	∞	∞ *2e5*	0/15
ga100	58(23)	2015(2076)	1.6e4(2e4)	∞	∞	∞ *2e5*	0/15

$\Delta f_{\rm opt}$	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f10	349	500	574	626	829	880	15/15
grid16	∞	∞	∞	∞	∞	∞ *2e5*	0/15
grid100	∞	∞	∞	∞	∞	∞ *2e5*	0/15
hill	1.0e4(1e4)	∞	∞	∞	∞	∞ *2e5*	0/15
ring16	∞	∞	∞	∞	∞	∞ *2e5*	0/15
ring100	1.0e4(1e4)	∞	∞	∞	∞	∞ *2e5*	0/15
ga100	**3067**(3575)	∞	∞	∞	∞	∞ *2e5*	0/15

$\Delta f_{\rm opt}$	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f11	143	202	763	1177	1467	1673	15/15
grid16	752(561)	4204(4161)	∞	∞	∞	∞ *2e5*	0/15
grid100	2799(3111)	∞	∞	∞	∞	∞ *2e5*	0/15
hill	197(192)	**845**(865)	4823(5654)	∞	∞	∞ *2e5*	0/15
ring16	295(408)	1715(1754)	∞	∞	∞	∞ *2e5*	0/15
ring100	**188**(220)	3942(4394)	**4713**(5080)	∞	∞	∞ *2e5*	0/15
ga100	202(93)	1652(1639)	∞	∞	∞	∞ *2e5*	0/15

$\Delta f_{\rm opt}$	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f12	108	268	371	461	1303	1494	15/15
grid16	834(1193)	6623(7417)	∞	∞	∞	∞ *2e5*	0/15
grid100	3812(3626)	∞	∞	∞	∞	∞ *2e5*	0/15
hill	4969(5942)	6174(7468)	9654(1e4)	∞	∞	∞ *2e5*	0/15
ring16	**603**(1164)	**1172**(1226)	**4723**(5277)	∞	∞	∞ *2e5*	0/15
ring100	1018(676)	1999(1913)	∞	∞	∞	∞ *2e5*	0/15
ga100	1059(1335)	2353(2736)	9857(1e4)	∞	∞	∞ *2e5*	0/15

$\Delta f_{\rm opt}$	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f13	132	195	250	1310	1752	2255	15/15
grid16	1004(1432)	1.8e4(2e4)	∞	∞	∞	∞ *2e5*	0/15
grid100	8027(9950)	9305(9960)	∞	∞	∞	∞ *2e5*	0/15
hill	2189(2842)	5715(6423)	∞	∞	∞	∞ *2e5*	0/15
ring16	729(965)	5313(6422)	∞	∞	∞	∞ *2e5*	0/15
ring100	**152**(52)	**756**(367)	∞	∞	∞	∞ *2e5*	0/15
ga100	1720(2839)	8953(9914)	∞	∞	∞	∞ *2e5*	0/15

$\Delta f_{\rm opt}$	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f14	10	41	58	139	251	476	15/15
grid16	**0.81**(0.7)	12(7)	24(13)	∞	∞	∞ *2e5*	0/15
grid100	1.9(2)	54(39)	117(54)	∞	∞	∞ *2e5*	0/15
hill	1.9(2)	**2.2**(1)$^{\star 2}$	**7.3**(6)$^{\star 2}$	2.5e4(3e4)	∞	∞ *2e5*	0/15
ring16	1.8(3)	8.3(3)	16(7)	2.7e4(3e4)	∞	∞ *2e5*	0/15
ring100	1.7(1)	27(12)	72(24)	**2277**(2723)	∞	∞ *2e5*	0/15
ga100	1.6(1)	14(9)	37(11)	3523(4138)	∞	∞ *2e5*	0/15

$\Delta f_{\rm opt}$	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f15	511	9310	19369	20073	20769	21359	14/15
grid16	131(248)	∞	∞	∞	∞	∞ *2e5*	0/15
grid100	335(383)	∞	∞	∞	∞	∞ *2e5*	0/15
hill	142(245)	∞	∞	∞	∞	∞ *2e5*	0/15
ring16	44(74)	383(450)	∞	∞	∞	∞ *2e5*	0/15
ring100	19(5)	198(201)	∞	∞	∞	∞ *2e5*	0/15
ga100	**9.2**(5)	**122**(137)	∞	∞	∞	∞ *2e5*	0/15

$\Delta f_{\rm opt}$	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f16	120	612	2662	10449	11644	12095	15/15
grid16	2.1(3)	98(207)	387(469)	∞	∞	∞ *2e5*	0/15
grid100	2.6(2)	185(242)	641(703)	∞	∞	∞ *2e5*	0/15
hill	2.9(2)	373(612)	∞	∞	∞	∞ *2e5*	0/15
ring16	**1.5**(1.0)	**6.4**(3)	117(147)	∞	∞	∞ *2e5*	0/15
ring100	3.1(4)	23(19)	**65**(67)	∞	∞	∞ *2e5*	0/15
ga100	3.2(3)	75(206)	119(146)	∞	∞	∞ *2e5*	0/15

$\Delta f_{\rm opt}$	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f17	5.2	215	899	3669	6351	7934	15/15
grid16	4.2(6)	101(69)	167(214)	∞	∞	∞ *2e5*	0/15
grid100	3.7(3)	29(18)	836(964)	∞	∞	∞ *2e5*	0/15
hill	39(19)	411(584)	1148(1390)	∞	∞	∞ *2e5*	0/15
ring16	5.9(8)	**3.7**(2)	117(142)	∞	∞	∞ *2e5*	0/15
ring100	3.8(6)	16(3)	37(22)	∞	∞	∞ *2e5*	0/15
ga100	**3.5**(6)	6.9(2)	**32**(9)	∞	∞	∞ *2e5*	0/15

$\Delta f_{\rm opt}$	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f18	103	378	3968	9280	10905	12469	15/15
grid16	8.5(12)	462(664)	∞	∞	∞	∞ *2e5*	0/15
grid100	16(14)	979(1089)	921(1008)	∞	∞	∞ *2e5*	0/15
hill	36(19)	454(661)	415(504)	∞	∞	∞ *2e5*	0/15
ring16	**3.6**(3)	453(666)	273(311)	∞	∞	∞ *2e5*	0/15
ring100	7.9(4)	32(15)	**78**(78)	∞	∞	∞ *2e5*	0/15
ga100	5.4(4)	**14**(9)	116(127)	∞	∞	∞ *2e5*	0/15

$\Delta f_{\rm opt}$	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f19	1	1	242	1.2e5	1.2e5	1.2e5	15/15
grid16	50(50)	1.6e4(2e4)	4684(5454)	∞	∞	∞ *2e5*	0/15
grid100	39(34)	4.8e4(1e5)	1.5e4(2e4)	∞	∞	∞ *2e5*	0/15
hill	42(32)	1.1e4(1e4)	6883(7950)	∞	∞	∞ *2e5*	0/15
ring16	**31**(24)	5321(5646)	3361(3947)	**29**(33)	**29**(35)	∞ *2e5*	0/15
ring100	49(55)	6741(5378)	3410(3595)	∞	∞	∞ *2e5*	0/15
ga100	41(37)	**3511**(2624)	**1121**(1103)	∞	∞	∞ *2e5*	0/15

$\Delta f_{\rm opt}$	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f20	16	851	38111	54470	54861	55313	14/15
grid16	13(8)	8.1(3)	8.7(11)	7.2(7)	∞	∞ *2e5*	0/15
grid100	41(44)	14(7)	14(17)	68(71)	∞	∞ *2e5*	0/15
hill	**5.7**(4)	4.4(8)	18(23)	13(16)	**16**(16)	∞ *2e5*	0/15
ring16	8.5(6)	**3.3**(0.5)	10(13)	7.7(10)	21(21)	∞ *2e5*	0/15
ring100	32(17)	7.4(2)	**1.4**(2)	**2.1**(1)	22(21)	∞ *2e5*	0/15
ga100	18(12)	4.1(1)	10(13)	7.6(9)	66(75)	∞ *2e5*	0/15

$\Delta f_{\rm opt}$	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f21	41	1157	1674	1705	1729	1757	14/15
grid16	**2.6**(3)	267(327)	414(522)	420(514)	436(469)	963(1138)	2/15
grid100	3.2(3)	170(224)	244(299)	421(513)	647(698)	2119(2348)	0/15
hill	6.8(6)	345(432)	416(523)	410(513)	411(507)	428(494)	4/15
ring16	2.7(2)	95(123)	86(149)	87(147)	95(146)	119(143)	9/15
ring100	5.3(5)	**2.6**(1)	**3.5**(2)	**11**(5)	**29**(17)	**81**(48)	8/15
ga100	3.8(4)	65(109)	133(187)	133(149)	139(151)	163(209)	8/15

$\Delta f_{\rm opt}$	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f22	71	386	938	1008	1040	1068	14/15
grid16	5.1(7)	685(970)	663(776)	3483(4092)	3457(3965)	∞ *2e5*	0/15
grid100	5.5(6)	128(326)	544(673)	1680(1860)	∞	∞ *2e5*	0/15
hill	6.7(10)	455(649)	321(444)	731(868)	∞	∞ *2e5*	0/15
ring16	**2.5**(2)	93(307)	320(413)	753(868)	3373(3965)	**3323**(3630)	0/15
ring100	5.5(3)	**11**(11)	**36**(33)	**201**(248)	829(834)	∞ *2e5*	0/15
ga100	5.9(5)	241(327)	439(537)	525(638)	**816**(886)	∞ *2e5*	0/15

$\Delta f_{\rm opt}$	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f23	3.0	518	14249	31654	33030	34256	15/15
grid16	2.2(3)	40(30)	∞	∞	∞	∞ *2e5*	0/15
grid100	3.0(2)	60(74)	∞	∞	∞	∞ *2e5*	0/15
hill	2.5(3)	49(45)	∞	∞	∞	∞ *2e5*	0/15
ring16	3.0(2)	26(29)	∞	∞	∞	∞ *2e5*	0/15
ring100	**1.8**(2)	**23**(27)	∞	∞	∞	∞ *2e5*	0/15
ga100	2.3(2)	45(46)	∞	∞	∞	∞ *2e5*	0/15

$\Delta f_{\rm opt}$	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
grid16	106(154)	∞	∞	∞	∞	∞ *2e5*	0/15
grid100	93(98)	∞	∞	∞	∞	∞ *2e5*	0/15
hill	38(72)	∞	∞	∞	∞	∞ *2e5*	0/15
ring16	**13**(12)	∞	∞	∞	∞	∞ *2e5*	0/15
ring100	19(17)	∞	∞	∞	∞	∞ *2e5*	0/15
ga100	15(16)	∞	∞	∞	∞	∞ *2e5*	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 $f_{\rm opt} + 10^{-8}$. 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
grid16	39(11)	105(13)	279(67)	2128(425)	2.0e4(4402)	∞ *1e6*	0/15
grid100	223(85)	574(100)	1496(267)	1.2e4(3766)	∞	∞ *1e6*	0/15
hill	**7.2**(2)^{*4}	**21**(6)^{*4}	**62**(14)^{*4}	**612**(183)	**5461**(1401)	∞ *1e6*	0/15
ring16	25(4)	67(8)	145(19)	743(185)	6011(1402)	∞ *1e6*	0/15
ring100	129(20)	333(67)	698(61)	2271(203)	8144(2008)	∞ *1e6*	0/15
ga100	56(13)	152(19)	359(47)	1518(285)	9101(1908)	∞ *1e6*	0/15

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f2	385	386	387	390	391	393	15/15
grid16	312(163)	1174(531)	3.8e4(4e4)	∞	∞	∞ *1e6*	0/15
grid100	1764(663)	∞	∞	∞	∞	∞ *1e6*	0/15
hill	174(114)	**487**(255)	1996(1598)	∞	∞	∞ *1e6*	0/15
ring16	**173**(72)	571(460)	2014(1879)	∞	∞	∞ *1e6*	0/15
ring100	205(44)	510(179)	**1622**(780)	∞	∞	∞ *1e6*	0/15
ga100	219(124)	643(533)	2437(1831)	∞	∞	∞ *1e6*	0/15

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f3	5066	7626	7635	7643	7646	7651	15/15
grid16	3.4(1.0)	9.1(2)	31(6)	∞	∞	∞ *1e6*	0/15
grid100	19(3)	54(12)	637(640)	∞	∞	∞ *1e6*	0/15
hill	**0.83**(0.4)^{*3}	**3.1**(2)	10(2)	**87**(25)	∞	∞ *1e6*	0/15
ring16	1.5(0.3)	3.4(0.9)	**10**(3)	111(77)	∞	∞ *1e6*	0/15
ring100	7.2(0.8)	10(0.9)	18(2)	131(88)	∞	∞ *1e6*	0/15
ga100	3.5(0.5)	6.2(2)	18(3)	203(135)	∞	∞ *1e6*	0/15

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f4	4722	7628	7666	7700	7758	1.4e5	9/15
grid16	4.9(1)	13(4)	37(12)	∞	∞	∞ *1e6*	0/15
grid100	29(6)	74(12)	∞	∞	∞	∞ *1e6*	0/15
hill	**1.3**(0.3)^{*3}	**3.6**(1)	**13**(6)	**191**(148)	∞	∞ *1e6*	0/15
ring16	2.3(0.7)	4.8(2)	16(6)	373(325)	∞	∞ *1e6*	0/15
ring100	9.2(0.6)	12(2)	23(4)	302(274)	∞	∞ *1e6*	0/15
ga100	4.7(0.8)	8.7(2)	23(7)	641(652)	∞	∞ *1e6*	0/15

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f5	41	41	41	41	41	41	15/15
grid16	61(20)	74(15)	80(14)	81(17)	81(17)	81(17)	15/15
grid100	347(70)	462(69)	482(95)	486(90)	486(90)	486(90)	15/15
hill	**11**(3)^{*4}	**14**(5)^{*4}	**15**(5)^{*4}	**15**(5)^{*4}	**15**(5)^{*4}	**15**(5)^{*4}	15/15
ring16	36(5)	47(5)	50(5)	52(7)	52(7)	52(7)	15/15
ring100	187(15)	242(26)	263(27)	265(28)	265(28)	265(28)	15/15
ga100	90(12)	118(13)	130(17)	133(18)	133(18)	133(18)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f6	1296	2343	3413	5220	6728	8409	15/15
grid16	443(639)	1259(1344)	4177(4835)	∞	∞	∞ *1e6*	0/15
grid100	1211(1369)	∞	∞	∞	∞	∞ *1e6*	0/15
hill	271(407)	1852(2180)	4258(4982)	∞	∞	∞ *1e6*	0/15
ring16	29(21)	367(460)	1278(1457)	∞	∞	∞ *1e6*	0/15
ring100	53(13)	98(36)	**257**(208)	∞	∞	∞ *1e6*	0/15
ga100	**21**(9)	**78**(62)	331(382)	∞	∞	∞ *1e6*	0/15

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f7	1351	4274	9503	16524	16524	16969	15/15
grid16	2371(2732)	∞	∞	∞	∞	∞ *1e6*	0/15
grid100	5054(5872)	∞	∞	∞	∞	∞ *1e6*	0/15
hill	2235(2591)	∞	∞	∞	∞	∞ *1e6*	0/15
ring16	842(908)	∞	∞	∞	∞	∞ *1e6*	0/15
ring100	**93**(121)	∞	∞	∞	∞	∞ *1e6*	0/15
ga100	175(224)	∞	∞	∞	∞	∞ *1e6*	0/15

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f8	2039	3871	4040	4219	4371	4484	15/15
grid16	611(765)	1113(1275)	3657(3960)	∞	∞	∞ *1e6*	0/15
grid100	2245(2321)	∞	∞	∞	∞	∞ *1e6*	0/15
hill	**260**(491)	**243**(388)	**1713**(1947)	∞	∞	∞ *1e6*	0/15
ring16	3367(3723)	3820(4134)	∞	∞	∞	∞ *1e6*	0/15
ring100	3250(3679)	3671(4134)	3702(4208)	∞	∞	∞ *1e6*	0/15
ga100	1080(1237)	1830(1999)	∞	∞	∞	∞ *1e6*	0/15

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f9	1716	3102	3277	3455	3594	3727	15/15
grid16	∞	∞	∞	∞	∞	∞ *1e6*	0/15
grid100	∞	∞	∞	∞	∞	∞ *1e6*	0/15
hill	∞	∞	∞	∞	∞	∞ *1e6*	0/15
ring16	∞	∞	∞	∞	∞	∞ *1e6*	0/15
ring100	∞	∞	∞	∞	∞	∞ *1e6*	0/15
ga100	∞	∞	∞	∞	∞	∞ *1e6*	0/15

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f10	7413	8661	10735	14920	17073	17476	15/15
grid16	∞	∞	∞	∞	∞	∞ *1e6*	0/15
grid100	∞	∞	∞	∞	∞	∞ *1e6*	0/15
hill	∞	∞	∞	∞	∞	∞ *1e6*	0/15
ring16	∞	∞	∞	∞	∞	∞ *1e6*	0/15
ring100	∞	∞	∞	∞	∞	∞ *1e6*	0/15
ga100	∞	∞	∞	∞	∞	∞ *1e6*	0/15

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f11	1002	2228	6278	9762	12285	14831	15/15
grid16	2753(2744)	∞	∞	∞	∞	∞ *1e6*	0/15
grid100	∞	∞	∞	∞	∞	∞ *1e6*	0/15
hill	577(255)	∞	∞	∞	∞	∞ *1e6*	0/15
ring16	470(240)	∞	∞	∞	∞	∞ *1e6*	0/15
ring100	808(199)	∞	∞	∞	∞	∞ *1e6*	0/15
ga100	**333**(121)	∞	∞	∞	∞	∞ *1e6*	0/15

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f12	1042	1938	2740	4140	12407	13827	15/15
grid16	2612(2440)	∞	∞	∞	∞	∞ *1e6*	0/15
grid100	∞	∞	∞	∞	∞	∞ *1e6*	0/15
hill	685(630)	**2451**(2615)	∞	∞	∞	∞ *1e6*	0/15
ring16	**571**(548)	7559(8258)	∞	∞	∞	∞ *1e6*	0/15
ring100	605(147)	2564(2332)	∞	∞	∞	∞ *1e6*	0/15
ga100	721(628)	7713(9032)	∞	∞	∞	∞ *1e6*	0/15

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f13	652	2021	2751	18749	24455	30201	15/15
grid16	1143(1534)	7247(7669)	∞	∞	∞	∞ *1e6*	0/15
grid100	5140(5336)	∞	∞	∞	∞	∞ *1e6*	0/15
hill	1221(1554)	7252(7916)	∞	∞	∞	∞ *1e6*	0/15
ring16	960(1534)	1613(1831)	∞	∞	∞	∞ *1e6*	0/15
ring100	**275**(67)	**662**(586)	∞	∞	∞	∞ *1e6*	0/15
ga100	519(781)	1266(1272)	∞	∞	∞	∞ *1e6*	0/15

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f14	75	239	304	932	1648	15661	15/15
grid16	22(10)	20(4)	43(8)	∞	∞	∞ *1e6*	0/15
grid100	100(46)	112(21)	257(72)	∞	∞	∞ *1e6*	0/15
hill	**3.5**(1)^{*4}	**3.6**(0.9)^{*4}	**10**(3)^{*4}	∞	∞	∞ *1e6*	0/15
ring16	11(5)	12(2)	23(5)	∞	∞	∞ *1e6*	0/15
ring100	50(11)	61(6)	121(16)	∞	∞	∞ *1e6*	0/15
ga100	19(8)	24(6)	47(11)	∞	∞	∞ *1e6*	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
grid16	∞	∞	∞	∞	∞	∞ *1e6*	0/15
grid100	∞	∞	∞	∞	∞	∞ *1e6*	0/15
hill	∞	∞	∞	∞	∞	∞ *1e6*	0/15
ring16	∞	∞	∞	∞	∞	∞ *1e6*	0/15
ring100	∞	∞	∞	∞	∞	∞ *1e6*	0/15
ga100	∞	∞	∞	∞	∞	∞ *1e6*	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
grid16	276(362)	∞	∞	∞	∞	∞ *1e6*	0/15
grid100	294(387)	∞	∞	∞	∞	∞ *1e6*	0/15
hill	322(382)	∞	∞	∞	∞	∞ *1e6*	0/15
ring16	106(298)	∞	∞	∞	∞	∞ *1e6*	0/15
ring100	**16**(6)	∞	∞	∞	∞	∞ *1e6*	0/15
ga100	59(2)	∞	∞	∞	∞	∞ *1e6*	0/15

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f17	63	1030	4005	30677	56288	80472	15/15
grid16	21(21)	∞	∞	∞	∞	∞ *1e6*	0/15
grid100	25(28)	∞	∞	∞	∞	∞ *1e6*	0/15
hill	6895(7962)	∞	∞	∞	∞	∞ *1e6*	0/15
ring16	**4.6**(2)	∞	∞	∞	∞	∞ *1e6*	0/15
ring100	12(10)	1587(1961)	∞	∞	∞	∞ *1e6*	0/15
ga100	7.4(3)	**8.9**(3)^{*4}	∞	∞	∞	∞ *1e6*	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
grid16	2.4e4(3e4)	∞	∞	∞	∞	∞ *1e6*	0/15
grid100	1.1e4(1e4)	∞	∞	∞	∞	∞ *1e6*	0/15
hill	∞	∞	∞	∞	∞	∞ *1e6*	0/15
ring16	815(1611)	∞	∞	∞	∞	∞ *1e6*	0/15
ring100	**54**(49)	∞	∞	∞	∞	∞ *1e6*	0/15
ga100	80(2)	∞	∞	∞	∞	∞ *1e6*	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
grid16	1200(899)	∞	∞	∞	∞	∞ *1e6*	0/15
grid100	4258(2178)	∞	∞	∞	∞	∞ *1e6*	0/15
hill	1100(837)	∞	∞	∞	∞	∞ *1e6*	0/15
ring16	**596**(184)	∞	∞	∞	∞	∞ *1e6*	0/15
ring100	2654(824)	∞	∞	∞	∞	∞ *1e6*	0/15
ga100	1023(368)	∞	∞	∞	∞	∞ *1e6*	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
grid16	27(7)	0.50(0.2)$_{\downarrow4}$	∞	∞	∞	∞ *1e6*	0/15
grid100	153(61)	3.2(2)	∞	∞	∞	∞ *1e6*	0/15
hill	**5.6**(2)^{*4}	**0.13**(0.1)$_{\downarrow4}$	4.6(5)	∞	∞	∞ *1e6*	0/15
ring16	17(3)	0.17(0.1)$_{\downarrow4}$	∞	∞	∞	∞ *1e6*	0/15
ring100	75(14)	0.65(0.1)$_{\downarrow4}$	**0.68**(0.8)^{*2}	**0.54**(0.6)^{*2}	**1.3**(1)^{*2}	∞ *1e6*	0/15
ga100	35(7)	0.39(0.1)$_{\downarrow4}$	∞	∞	∞	∞ *1e6*	0/15

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f21	561	6541	14103	14643	15567	17589	15/15
grid16	654(895)	612(764)	284(354)	277(342)	271(322)	263(310)	0/15
grid100	1275(1807)	∞	∞	∞	∞	∞ *1e6*	0/15
hill	2036(2672)	∞	∞	∞	∞	∞ *1e6*	0/15
ring16	279(891)	230(307)	284(355)	275(308)	262(290)	239(284)	3/15
ring100	14(3)	**30**(77)	**29**(36)	**34**(37)	**42**(37)	**54**(36)	8/15
ga100	**7.1**(2)	612(688)	285(354)	276(342)	265(305)	248(284)	3/15

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f22	467	5580	23491	24948	26847	1.3e5	12/15
grid16	2457(3216)	1171(1428)	**622**(660)	∞	∞	∞ *1e6*	0/15
grid100	3247(4295)	1170(1389)	∞	∞	∞	∞ *1e6*	0/15
hill	780(1071)	1165(1344)	∞	∞	∞	∞ *1e6*	0/15
ring16	333(1071)	494(629)	∞	∞	∞	∞ *1e6*	0/15
ring100	**17**(6)	**125**(180)	∞	∞	∞	∞ *1e6*	0/15
ga100	787(1073)	1166(1389)	∞	∞	∞	∞ *1e6*	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
grid16	2.5(3)	459(490)	∞	∞	∞	∞ *1e6*	0/15
grid100	2.2(2)	2274(2122)	∞	∞	∞	∞ *1e6*	0/15
hill	2.1(1)	1120(1264)	∞	∞	∞	∞ *1e6*	0/15
ring16	**1.4**(2)	**264**(345)	∞	∞	∞	∞ *1e6*	0/15
ring100	2.1(3)	274(327)	∞	∞	∞	∞ *1e6*	0/15
ga100	1.6(2)	1486(1693)	∞	∞	∞	∞ *1e6*	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
grid16	∞	∞	∞	∞	∞	∞ *1e6*	0/15
grid100	∞	∞	∞	∞	∞	∞ *1e6*	0/15
hill	∞	∞	∞	∞	∞	∞ *1e6*	0/15
ring16	∞	∞	∞	∞	∞	∞ *1e6*	0/15
ring100	∞	∞	∞	∞	∞	∞ *1e6*	0/15
ga100	∞	∞	∞	∞	∞	∞ *1e6*	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 $f_{opt} + 10^{-8}$. Best results are printed in bold.