Evaluating the Population Size Adaptation Mechanism for CMA-ES on the BBOB Noiseless Testbed

Kouhei Nishida
Graduate School of Science and Engineering,
Shinshu University
15tm232a@shinshu-u.ac.jp

Youhei Akimoto
Faculty of Engineering,
Shinshu University
y_akimoto@shinshu-u.ac.jp

ABSTRACT

The population size adaptation mechanism for CMA-ES is evaluated on the BBOB noiseless testbed. The population size is adapted on the basis of the estimated accuracy of the update of the distribution parameters, i.e., the mean vector and the covariance matrix of the Gaussian distribution. The population size is adapted so that the estimated accuracy of the parameter update keeps a certain level. The CMA-ES with the population size adaptation mechanism could solve well-structured multimodal functions as efficiently as the best 2009 portfolio without a restart strategy that increases the population size every restart such as the IPOP strategy.

Keywords

Benchmarking, Black-Box Optimization, Covariance Matrix Adaptation, Population Size Adaptation

1. INTRODUCTION

The Covariance Matrix Adaptation Evolution Strategy (CMA-ES) is a stochastic and comparison-based search algorithm for continuous optimization. Thanks to the invariance to several transformations of the problem such as any strictly increasing transformation of the objective function and any positive definite affine transformation of the search space, the CMA-ES can efficiently solve difficult problems such as ill-conditioned and non-separable problems.

The CMA-ES is a quasi parameter free algorithm. That is, all the parameters used in the CMA-ES are not necessarily tuned for each problem. Their default values are predefined based on the number of variables to optimize, i.e., the search space dimension D. However, it is well-known that a large population size helps to find a better solution on relatively well-structured multimodal functions [8]. It is also important to set the initial step-size to a relatively small value compared to the search interval when solving weakly-structured multimodal functions [4]. The BIPOP restart strategy [4] tackles these difficulties by a restart strategy with two budgets of function evaluations—one is for incremental population size, the other is for relatively small population size and a relatively small step-size.

To tackle well-structured multimodal functions and noisy functions, where a large population size has an impact on the performance, the population size adaptation mechanism for the rank-μ update CMA-ES is proposed in [10]. It adapts the population size online based on the randomness of the parameter update in the parameter space, i.e., the product space of the domains of the mean vector and covariance matrix of the multivariate Gaussian distributions. It tries to keep the randomness of the parameter update in a certain level by increasing or decreasing the population size at each iteration.

In this paper, we evaluate the population size adaptation mechanism for the rank-μ update CMA-ES on the BBOB noiseless testbed and on the BBOB noisy testbed in a companion paper. Four algorithm variants with different parameter settings are compared.

GECCO'16 Companion, July 20 - 24, 2016, Denver, CO, USA

ISBN 978-1-4503-4323-7/16/07. . . $15.00

DOI: http://dx.doi.org/10.1145/2908961.2931700

2. ALGORITHM DESCRIPTION

The baseline algorithm for this paper is the so-called pure rank-μ update $(\mu/\mu_w, \lambda)$-CMA-ES. It is a variant of CMA-ES without step-size adaptation and rank-one covariance matrix update. It samples λ candidate solutions from the Gaussian distribution $\mathcal{N}(\boldsymbol{m}, \mathbf{C})$, evaluates them on the objective function, and updates the parameters $\boldsymbol{m}$ and $\mathbf{C}$ using the information of the candidate solutions and their fitness ranking. The weighted recombination and the rank-μ update are employed to update the mean vector and the covariance matrix. We introduce the learning rate $c_{\boldsymbol{m}}$ for the mean vector update.

The population size adaptation proposed in [10] is combined with the pure rank-μ update $(\mu/\mu_w, \lambda)$-CMA-ES. The idea of the population size adaptation is to estimate the signal-to-noise ratio of the parameter update and control the population size λ so that the signal-to-noise ratio is kept constant during optimization. The evolution path in the space of the parameters of the Gaussian distribution is introduced and its squared length is used to estimate the signal-to-noise ratio of the parameter update. The squared length of the evolution path is compared with a constant $\alpha > 1$ times the expected length of the evolution path under the random function. In [10], the rank-μ update CMA-ES is considered, where the expected length of the evolution path under the random function is analytically derived. Introducing the step-size adaptation and the rank-one covariance matrix update will make it rather complicated to mathematically analyze the expected length of the evolution path. This is an on-going future work and we restrict our attention to the rank-μ update CMA-ES in this paper.

The pseudocode of the rank-μ update CMA-ES with the population size adaptation is given below.

Algorithm 1 Rank-μ Update CMA-ES with λ-Adaptation

Require: $\boldsymbol{m}$, $\mathbf{C}$, $c_{\boldsymbol{m}}$, α, and initialize $\lambda = \lambda_{\min} = 4$, $c_\mu = c_{\boldsymbol{m}}/\sqrt{(D+1)/2}$, $\beta = \min(c_{\boldsymbol{m}}, 0.9)$.

repeat
  $\mu \leftarrow \lfloor \lambda/2 \rfloor$, $w_i \leftarrow (\ln((\lambda+1)/2) - \ln(i)) / \sum_{i=1}^{\mu}(\ln((\lambda+1)/2) - \ln(i))$ for $i = 1, \dots, \lambda$
  Sample $x_i \sim \mathcal{N}(\boldsymbol{m}, \mathbf{C})$ for $i = 1, \dots, \lambda$
  Evaluate $f(x_i)$ for $i = 1, \dots, \lambda$
  $x_{i:\lambda} \leftarrow$ the ith best among $\{x_i\}_{i=1,\dots,\lambda}$ w.r.t. $f(x_i)$
  $\boldsymbol{m}' \leftarrow \boldsymbol{m}$, $\mathbf{C}' \leftarrow \mathbf{C}$ ▷ keep old values
  $\mathbf{C} \leftarrow \mathbf{C} + c_\mu \sum_{i=1}^{\mu}[w_i(x_{i:\lambda} - \boldsymbol{m})(x_{i:\lambda} - \boldsymbol{m})^{\mathrm{T}} - \mathbf{C}]$
  $\boldsymbol{m} \leftarrow \boldsymbol{m} + c_{\boldsymbol{m}} \sum_{i=1}^{\mu} w_i(x_{i:\lambda} - \boldsymbol{m})$
  $\boldsymbol{p_m} \leftarrow (1-\beta)\boldsymbol{p_m} + \sqrt{\beta(2-\beta)}(\boldsymbol{m} - \boldsymbol{m}')$
  $\mathbf{P_C} \leftarrow (1-\beta)\mathbf{P_C} + \sqrt{\beta(2-\beta)}(\mathbf{C} - \mathbf{C}')$
  $\gamma \leftarrow (1-\beta)^2\gamma + \beta(2-\beta)(c_{\boldsymbol{m}}^2 D + c_\mu^2 D(D+1)/2) \sum_{i=1}^{\mu} w_i^2$
  $\Delta = \boldsymbol{p_m}^{\mathrm{T}}(\mathbf{C}')^{-1}\boldsymbol{p_m} + \mathrm{Tr}((\mathbf{P_C}(\mathbf{C}')^{-1})^2)/2$
  if $\Delta/\gamma < \alpha$ **then**
    $\lambda \leftarrow \lfloor \lambda \exp(\beta(\alpha - \Delta/\gamma)) \rfloor \vee \lambda + 1$
  else
    $\lambda \leftarrow \lfloor \lambda \exp(\beta(\alpha - \Delta/\gamma)) \rfloor \vee \lambda_{\min}$
until a termination condition is met.

Note that the mean vector update and the rank-μ covariance matrix update follow the estimation of the natural gradient [2,11]. It has been shown in [11] that the estimated natural gradient converges as $\lambda \to \infty$ if the objective function is deterministic. It implies that the accuracy of the parameter update can be arbitrarily high as we increase the population size. Indeed, $\Delta/\gamma \to \infty$ as $\lambda \to \infty$ since $\gamma \to 0$ as $\lambda \to \infty$. Therefore, there exists a stationary population size. An analogous argument holds for noisy optimization. See the companion paper.

2.1 Algorithm Variants

We compare four variants of the population size adaptation with different parameter settings. The modified parameters are the learning rate $c_{\boldsymbol{m}}$ for the mean vector and the threshold α to decide whether the parameter update is considered accurate or not. The combinations are listed below.

PSAaLmC $\alpha = \sqrt{2}$, $c_{\boldsymbol{m}} = 0.1$

PSAaLmD $\alpha = \sqrt{2}$, $c_{\boldsymbol{m}} = 1/D$

PSAaSmC $\alpha = 1.1$, $c_{\boldsymbol{m}} = 0.1$

PSAaSmD $\alpha = 1.1$, $c_{\boldsymbol{m}} = 1/D$

The greater α is, the greater the population size tends to be kept by the population size adaptation mechanism. The other parameters are set as described in [10].

2.2 Restart Strategy

For each (re-)start of the algorithm, we initialize the mean vector $\boldsymbol{m} \sim \mathcal{U}[-4,4)^D$ and $\mathbf{C} = 2^2\mathbf{I}$. The termination conditions for each (re-)start are as follows:

`tolf` `median(fiqr_hist)` $< 10^{-12}$`abs(median(fmin_hist))`, where `fiqr_hist` and `fmin_hist` are the histories of the interquartile ranges and the minimum values, respectively, of the function values for the last 20 iterations. If this is true, we consider that the objective function value differences are too small to sort them without being affected by numerical errors.

`tolx` `median(xiqr_hist`$_i$`)` $< 10^{-12}$`min(abs(xmed_hist`$_i$`))` for some $i = 1, \dots, D$, where `xiqr_hist`$_i$ and `xmed_hist`$_i$ are the histories of the interquartile ranges and the median values, respectively, of the ith coordinate of the candidate solutions for the last 20 iterations. If this is true, we consider that the coordinate value differences are too small to update parameters without being affected by numerical errors.

`maxcond` `cond(C)` $> 10^{-14}$, where `cond(C)` is the condition number, i.e., the ratio of the maximum and minimum eigenvalues of $\mathbf{C}$. If this is true, we consider the matrix operations using $\mathbf{C}$ are not reliable due to numerical errors.

`maxeval` $\#f$-call $\geqslant 5 \cdot 10^4 D$, while the maximum number of function calls per function instance is $10^5 D$.

Note that we do not implement a mechanism to shrink the initial step-size for restart. Therefore, we do not expect this restart strategy is effective on weakly-structured functions.

3. CPU TIMING

In order to evaluate the CPU timing of the algorithm, we have run PSAaLmD on the function f_8 with restarts until a maximum budget equal to $400(D+2)$ is reached. The code was run on a Mac Intel(R) Core(TM) i5-4288U CPU @ 2.60GHz with 1 processor and 4 cores. The time per function evaluation for dimensions 2, 3, 5, 10, 20 are 2.0, 4.8, 6.7, 7.1, 6.5 milliseconds, respectively. Note that all the variants tested have the same internal complexity per function evaluation.

4. RESULTS

Results from experiments according to [6] on the benchmark functions given in [3,7] are presented in Figures 1, 2 and 3 and in Tables 1 and 2. In Figures 1, 2 and 3, we present the results of the BIPOP-CMA-ES, taken from [4], as references. The results of the BIPOP-CMA-ES are removed from the tables due to the paper size limitation. The **averaged running time** (aRT), used in the figures and tables, depends on a given target function value, $f_{\mathrm{t}} = f_{\mathrm{opt}} + \Delta f$, 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} [6,12]. **Statistical significance** is tested with the rank-sum test for a given target Δf_{t} using, for each trial, either the number of needed function evaluations to reach Δf_{t} (inverted and multiplied by -1), or, if the target was not reached, the best Δf-value achieved, measured only up to the smallest number of overall function evaluations for any unsuccessful trial under consideration. The maximum function evaluations are set to $10^5 D$, and the experiments are done for $D = 2, 3, 5, 10, 20$.

5. DISCUSSION

Unimodal functions. Due to the lack of the step-size adaptation, we observe from Figure 1 that the convergence speed scales worse on Sphere function and the aRT is higher for most of the unimodal functions than the best 2009 portfolio. Especially for Rosenbrock functions, it fails to find the target function value within a given budget. An exception is Step-Ellipsoid function (f_7), where all of the algorithm variants scale similarly to the best 2009. It is because the function is locally flat and the convergence towards the optimum is less important than the adaptation of the covariance matrix. Then, the weak point of the algorithm variants—no step-size adaptation—is less emphasized. We expect that the performance of the algorithms will be improved if the step-size adaptation mechanism is introduced.

Well-structured multimodal functions. On f_{15}, f_{17}, f_{18}, and f_{19}, we observe that the tested algorithms scale similarly to the best 2009. It implies that the well-structured functions can be solved efficiently without a restart strategy using increasing population sizes.

Weakly-structured multimodal functions. As we expected, the population size adaptation is not very effective for weakly-structured functions. To tackle weakly-structured functions, we may need to restart the algorithm with varying the initial step-size. This should come with the step-size adaptation, which is the future work.

Comparing the variants. On noiseless functions, variants with $\alpha = 1.1$ tends to be better than ones with $\alpha = \sqrt{2}$, the former of which tends to keep the population size smaller than the latter. The difference due to c_m is visible when $D \geqslant 5$, where $c_m = 0.1$ is better than $1/D \geqslant 0.2$ on functions f_2, f_{10}, and f_{11}. The effect of a small learning rate for the mean update should be studied in details in the future work.

6. SUMMARY AND FUTURE WORK

We have evaluated the rank-μ update CMA-ES with the population size adaptation mechanism on the BBOB noiseless testbed. It has been revealed that the population size adaptation mechanism works well on well-structured multimodal functions, where the observed performance is similar to the best 2009 portfolio even without the step-size adaptation. On the other hand, the population size adaptation is not effective on weakly-structured multimodal functions, where a large population size does not help finding a global optimum, but a restart with a relatively small initial step-size is required. For such functions, we should employ a restart strategy that restarts the optimization with a random initialization of the mean vector and a varying initial step-size. With such a restart strategy, we expect that the performance of the algorithm improves on weakly-structured multimodal functions.

The future work is to incorporate the step-size adaptation mechanism such as the cumulative step-size adaptation (CSA) [9] and the two-point step-size adaptation [5] to make the algorithm robust and less sensitive to the initial step-size. This will improve the performance on many unimodal functions. It will accelerate the convergence, which is helpful also on multimodal functions since it results in restarting the run more often. We also need to compare this population size adaptation mechanism with the algorithm proposed in [1], which, however, requires a tuning of the maximum population size depending on problems. We may increase the maximum population size with restart, as the IPOP and the BIPOP restart strategy increase the population size.

Acknowledgements. This work is partially supported by JSPS KAKENHI Grant Number 15K16063 and the Kayamori Foundation of Informational Science Advancement.

7. REFERENCES

[1] A. Ahrari and M. Shariat-Panahi. An improved evolution strategy with adaptive population size. *Optimization*, 64(12):2567–2586, 2015.

[2] Y. Akimoto, Y. Nagata, I. Ono, and S. Kobayashi. Bidirectional relation between CMA evolution strategies and natural evolution strategies. In *Parallel Problem Solving from Nature – PPSN XI*, pages 154–163, Springer-Verlag, 2010.

[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. Benchmarking a BI-population CMA-ES on the BBOB-2009 function testbed. In *Workshop Proceedings of the Genetic and Evolutionary Computation Conference*, pages 2389–2395, ACM, 2009.

[5] N. Hansen, A. Atamna, and A. Auger. How to assess step-size adaptation mechanisms in randomised search. In *Parallel Problem Solving from Nature–PPSN XIII*, pages 60–69. Springer, 2014.

[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] N. Hansen and S. Kern. Evaluating the CMA evolution strategy on multimodal test functions. In *Parallel Problem Solving from Nature - PPSN VIII*, pages 282–291. Springer, 2004.

[9] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[10] K. Nishida and Y. Akimoto. Population size adaptation for the CMA-ES based on the estimation accuracy of the natural gradient. In *Proceedings of the Genetic and Evolutionary Computation Conference*, ACM, 2016. To appear.

[11] Y. Ollivier, L. Arnold, A. Auger, and N. Hansen. Information-geometric optimization algorithms: A unifying picture via invariance principles. *Preprint arXiv:1106.3708*.

[12] 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.

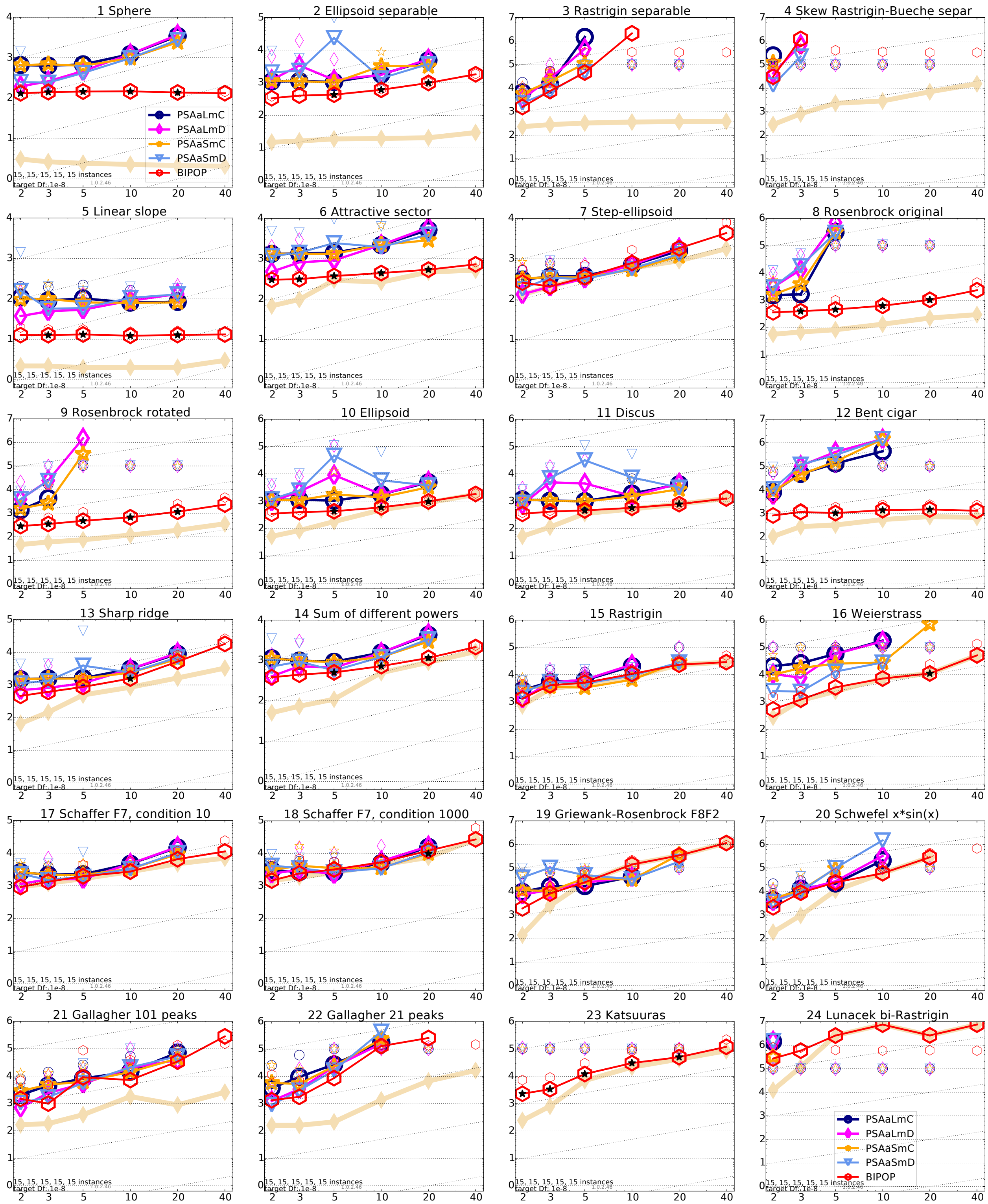

Figure 1: Average running time (aRT in number of f-evaluations as $\log_{10}$ value), divided by dimension for target function value 10^{-8} versus dimension. Slanted grid lines indicate quadratic scaling with the dimension. Different symbols correspond to different algorithms given in the legend of f_1 and f_{24}. Light symbols give the maximum number of function evaluations from the longest trial divided by dimension. Black stars indicate a statistically better result compared to all other algorithms with $p < 0.01$ and Bonferroni correction number of dimensions (six). Legend: ○: PSAaLmC, ◇: PSAaLmD, ⋆: PSAaSmC, ▽: PSAaSmD, ⬡: BIPOP

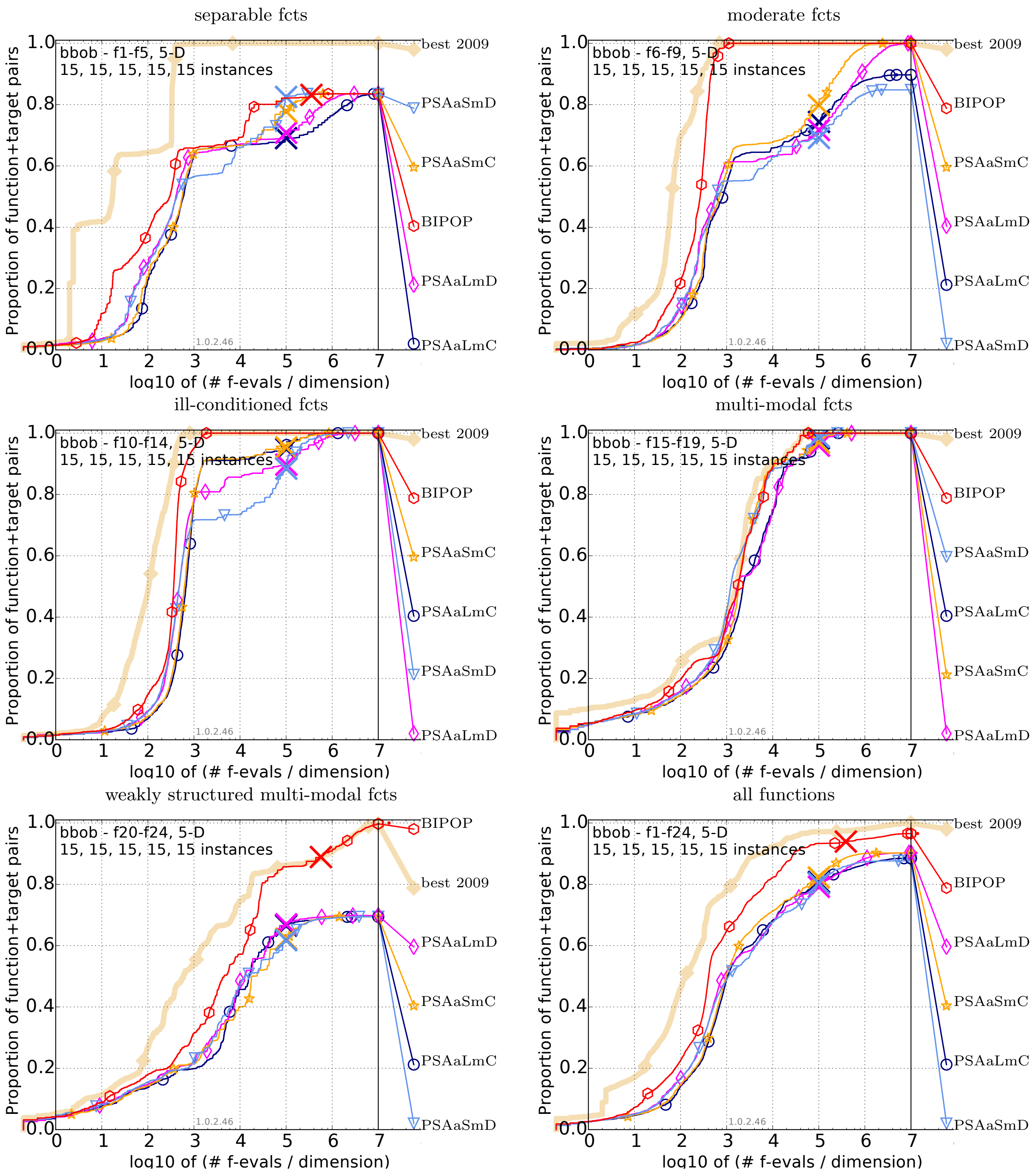

Figure 2: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for 51 targets with target precision in $10^{[-8..2]}$ for all functions and subgroups in 5-D. The "best 2009" line corresponds to the best aRT observed during BBOB 2009 for each selected target.

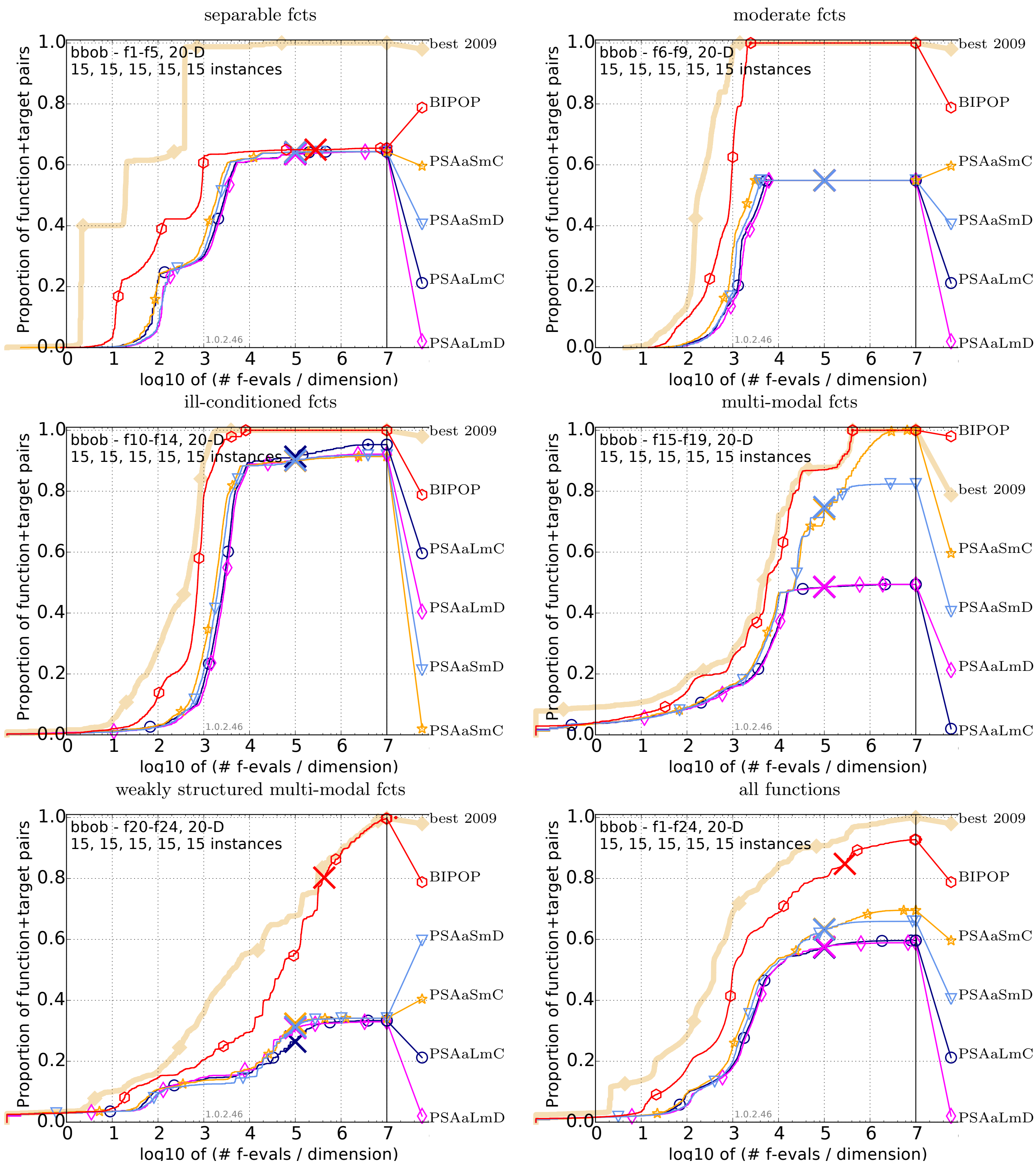

Figure 3: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for 51 targets with target precision in $10^{[-8..2]}$ for all functions and subgroups in 20-D. The "best 2009" line corresponds to the best aRT observed during BBOB 2009 for each selected target.

Table 1: Average running time (aRT in number of function evaluations) divided by the respective best aRT measured during BBOB-2009 in dimension 5. The aRT and in braces, as dispersion measure, the half difference between 10 and 90%-tile of bootstrapped run lengths appear for each algorithm and target, the corresponding best aRT in the first row. The different target Δf-values are shown in the top row. #succ is the number of trials that reached the (final) target $f_{\mathrm{opt}} + 10^{-8}$. The median number of conducted function evaluations is additionally given in *italics*, if the target in the last column was never reached. Entries, succeeded by a star, are statistically significantly better (according to the rank-sum test) when compared to all other algorithms of the table, with $p = 0.05$ or $p = 10^{-k}$ when the number k following the star is larger than 1, with Bonferroni correction by the number of instances. A ↓ indicates the same tested against the best algorithm of BBOB-2009.Best results are printed in bold.

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f1	11	12	12	12	12	12	12	15/15
aLmC	7.3(6)	25(6)	47(12)	67(11)	103(16)	178(9)	243(35)	15/15
aLmD	6.4(5)	**16**(5)	**26**(8)	**47**(8)	**68**(12)	118(9)	167(13)	15/15
aSmC	**6.3**(6)	23(9)	44(11)	71(13)	100(20)	166(15)	232(21)	15/15
aSmD	9.5(3)	19(14)	34(12)	50(7)	68(10)	**112**(6)	**149**(14)⋆2	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f2	83	87	88	89	90	92	94	15/15
aLmC	22(6)	25(5)	30(7)	33(6)	36(8)	44(5)	52(3)	15/15
aLmD	**13**(6)	**18**(7)	29(3)	41(127)	43(5)	49(70)	62(4)	15/15
aSmC	21(4)	25(4)	**29**(4)	**31**(3)	**34**(2)	**42**(3)	**50**(2)	15/15
aSmD	295(990)	836(2750)	866(1559)	975(870)	1278(866)	1274(510)	1249(1688)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f3	716	1622	1637	1642	1646	1650	1654	15/15
aLmC	2.6(3)	103(178)	2194(1146)	4555(4420)	4544(4105)	4533(4852)	4524(4388)	1/15
aLmD	4.8(2)	70(50)	1427(3309)	1423(1041)	1420(1215)	1417(2297)	1414(530)	3/15
aSmC	**2.2**(2)	56(53)	337(264)	336(393)	336(265)	335(441)	335(714)	9/15
aSmD	2.2(2)	**31**(28)	**146**(141)	**146**(241)	**146**(115)	**146**(94)	**146**(140)	14/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f4	809	1633	1688	1758	1817	1886	1903	15/15
aLmC	3.9(2)	∞	∞	∞	∞	∞	∞ *5e5*	0/15
aLmD	3.4(0.9)	∞	∞	∞	∞	∞	∞ *5e5*	0/15
aSmC	3.8(1)	∞	∞	∞	∞	∞	∞ *5e5*	0/15
aSmD	**3.1**(2)	∞	∞	∞	∞	∞	∞ *5e5*	0/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f5	10	10	10	10	10	10	10	15/15
aLmC	26(19)	47(34)	51(29)	51(17)	51(20)	51(41)	51(31)	15/15
aLmD	16(11)	**26**(14)	**27**(18)	**27**(17)	**27**(25)	**27**(24)	**27**(22)	15/15
aSmC	24(11)	38(24)	41(10)	41(22)	41(14)	41(25)	41(31)	15/15
aSmD	**15**(15)	28(18)	30(15)	30(21)	30(23)	30(21)	30(20)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f6	114	214	281	404	580	1038	1332	15/15
aLmC	5.3(3)	6.5(0.7)	7.3(0.6)	6.9(0.7)	6.0(0.9)	4.8(0.4)	4.8(0.4)	15/15
aLmD	**4.3**(1)	**4.4**(1)	**4.8**(1)	**4.5**(0.4)	**3.9**(0.2)	**3.1**(0.1)	**3.1**(0.2)	15/15
aSmC	5.7(3)	6.1(0.9)	7.1(2)	6.4(0.4)	5.6(0.3)	4.4(0.4)	4.5(0.3)	15/15
aSmD	4.4(1)	14(69)	17(27)	19(45)	16(38)	11(11)	8.8(9)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f7	24	324	1171	1451	1572	1572	1597	15/15
aLmC	7.5(4)	1.7(0.9)	0.92(0.3)	1.1(0.1)	1.1(0.1)	1.1(0.2)	1.1(0.1)	15/15
aLmD	8.5(5)	**1.5**(0.5)	**0.80**(0.6)	**0.87**(0.5)	**0.86**(0.6)	**0.86**(0.7)	**0.92**(0.3)	15/15
aSmC	9.4(11)	1.8(0.7)	0.81(0.3)	1.00(0.1)	1.0(0.2)	1.0(0.1)	1.1(0.1)	15/15
aSmD	**7.4**(3)	1.7(1)	0.87(0.8)	0.92(0.7)	0.91(0.2)	0.91(0.6)	0.95(0.1)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f8	73	273	336	372	391	410	422	15/15
aLmC	11(4)	81(234)	**229**(325)	**1538**(2769)	3063(3245)	**3015**(3047)	3782(2779)	4/15
aLmD	7.2(2)	84(167)	1180(1806)	2640(2655)	4236(3778)	5422(2972)	5270(5116)	3/15
aSmC	11(4)	**28**(66)	457(448)	1623(2118)	**2885**(2906)	3742(1738)	3637(6073)	4/15
aSmD	**6.3**(2)	135(85)	1434(1653)	3868(5127)	3800(3968)	3624(7064)	**3522**(4476)	4/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f9	35	127	214	263	300	335	369	15/15
aLmC	20(8)	127(182)	900(1905)	8587(9329)	1.2e4(2e4)	∞	∞ *5e5*	0/15
aLmD	15(3)	142(222)	1616(2032)	1.4e4(2e4)	1.2e4(1e4)	2.2e4(3e4)	2.0e4(2e4)	1/15
aSmC	18(8)	**78**(43)	**465**(596)	**1414**(1398)	**3147**(3538)	**2868**(7291)	**4391**(4456)	4/15
aSmD	**12**(5)	142(303)	4204(5317)	∞	∞	∞	∞ *5e5*	0/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f10	349	500	574	607	626	829	880	15/15
aLmC	5.3(0.8)	**4.4**(0.5)	**4.4**(0.4)	**4.8**(0.6)	**5.1**(0.4)	**4.8**(0.3)	**5.6**(0.2)	15/15
aLmD	36(240)	81(0.7)	71(474)	68(448)	70(226)	53(12)	51(618)	14/15
aSmC	**5.1**(2)	4.5(0.7)	5.0(4)	11(24)	11(0.3)	9.2(17)	10(33)	15/15
aSmD	108(223)	223(429)	256(621)	291(440)	284(626)	279(359)	263(692)	12/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f11	143	202	763	977	1177	1467	1673	15/15
aLmC	11(6)	11(5)	3.3(1)	2.8(1)	2.6(0.6)	2.6(0.5)	2.7(0.5)	15/15
aLmD	**7.4**(3)	26(69)	22(68)	19(53)	17(22)	15(12)	13(26)	15/15
aSmC	10(4)	**10**(0.9)	**3.1**(0.3)	**2.7**(0.4)	**2.5**(0.3)	**2.4**(0.1)	**2.6**(0.1)	15/15
aSmD	251(786)	342(888)	162(168)	159(125)	132(179)	107(115)	94(151)	13/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f12	108	268	371	413	461	1303	1494	15/15
aLmC	22(8)	100(78)	175(487)	253(343)	416(942)	396(681)	**438**(255)	7/15
aLmD	**14**(2)	192(3)	410(455)	832(1817)	1223(828)	1559(3479)	1359(2444)	3/15
aSmC	19(3)	**11**(3)	**35**(131)	**122**(146)	**383**(810)	**376**(321)	444(531)	7/15
aSmD	169(590)	229(583)	566(682)	690(1405)	969(1238)	908(565)	1000(661)	4/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f13	132	195	250	319	1310	1752	2255	15/15
aLmC	7.4(0.9)	9.0(1)	10(0.5)	10(0.7)	3.1(0.2)	3.2(0.1)	3.2(0.1)	15/15
aLmD	5.2(1)	6.4(1)	7.0(0.5)	7.2(0.5)	**2.2**(0.2)	**2.2**(0.1)	**2.3**(0.1)	15/15
aSmC	7.2(2)	8.6(2)	10(1)	10(1)	3.1(0.2)	3.2(0.1)	3.1(0.1)	15/15
aSmD	**5.1**(2)	**5.9**(2)	**6.5**(0.7)	**6.5**(2)	13(0.2)	10(32)	8.5(0.1)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f14	10	41	58	90	139	251	476	15/15
aLmC	3.9(9)	8.6(4)	11(3)	12(3)	12(1)	11(2)	8.7(0.8)	15/15
aLmD	3.4(4)	**4.5**(2)	**7.1**(2)	**7.3**(1)	8.1(0.9)	7.8(0.6)	6.0(0.8)	15/15
aSmC	**2.9**(4)	7.0(4)	10(4)	11(2)	11(2)	11(0.8)	8.1(1.0)	15/15
aSmD	4.1(5)	5.3(2)	7.2(0.8)	8.0(0.7)	**7.6**(2)	**7.0**(0.9)	**5.2**(0.5)⋆2	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f15	511	9310	19369	19743	20073	20769	21359	14/15
aLmC	3.9(2)	2.2(0.8)	1.5(1.0)	1.4(1)	1.4(0.4)	1.4(0.7)	1.4(0.8)	15/15
aLmD	**2.3**(1)	2.5(1)	1.6(0.6)	1.6(0.5)	1.6(0.5)	1.5(0.5)	1.5(0.6)	15/15
aSmC	3.7(2)	1.0(0.3)	**0.71**(0.5)	**0.72**(0.3)	**0.73**(0.2)	**0.75**(0.3)	**0.76**(0.3)	15/15
aSmD	3.0(4)	**0.88**(0.9)	1.1(1)	1.1(0.7)	1.1(2)	1.1(0.7)	1.1(0.7)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f16	120	612	2662	10163	10449	11644	12095	15/15
aLmC	**1.9**(2)	144(170)	104(69)	27(18)	27(22)	25(38)	24(16)	13/15
aLmD	3.0(3)	155(315)	105(66)	28(9)	27(26)	24(19)	24(24)	11/15
aSmC	3.1(4)	69(40)	36(54)	11(20)	12(18)	11(0.4)	11(18)	14/15
aSmD	4.2(4)	**49**(146)	**14**(39)	**5.5**(3)	**5.5**(13)	**5.5**(15)	**5.4**(3)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f17	5.2	215	899	2861	3669	6351	7934	15/15
aLmC	10(11)	2.9(0.9)	1.6(0.4)	1.0(0.3)	1.2(0.1)	1.2(0.1)	1.3(0.1)	15/15
aLmD	11(9)	2.1(0.7)	1.2(0.8)	0.76(0.3)	0.85(0.1)	0.85(0.0)	**0.94**(0.0)	15/15
aSmC	9.3(9)	2.3(1)	1.5(0.3)	0.90(0.1)	1.0(0.1)	1.0(0.1)	1.2(0.1)	15/15
aSmD	**6.1**(12)	**1.7**(0.8)	**1.1**(0.7)	**0.62**(0.1)⋆	**0.70**(0.1)⋆2	**0.68**(0.1)⋆2	1.1(2)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f18	103	378	3968	8451	9280	10905	12469	15/15
aLmC	2.3(2)	2.7(1)	**0.89**(1)	**0.59**(0.1)	**0.70**(0.8)	0.87(0.0)	0.98(0.6)	15/15
aLmD	**1.6**(1)	**1.9**(0.9)	1.1(3)	0.66(1)	0.71(0.0)	**0.81**(0.1)	**0.88**(0.8)	15/15
aSmC	3.1(1)	2.8(2)	2.3(6)	1.3(3)	1.3(2)	1.3(1)	1.4(0.9)	15/15
aSmD	2.2(2)	2.2(0.9)	1.7(3)	0.88(1)	0.89(0.1)	1.0(1)	1.0(1)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f19	1	1	242	1.0e5	1.2e5	1.2e5	1.2e5	15/15
aLmC	39(38)	1343(816)	220(102)	**0.62**(0.1)	**0.69**(1)	**0.69**(0.2)	**0.69**(0.6)	15/15
aLmD	25(27)	1277(1170)	263(104)	0.78(1)	0.84(0.6)	0.84(0.7)	0.97(1)	15/15
aSmC	24(22)	1736(1510)	**99**(40)	0.75(1)	1.5(4)	1.5(2)	1.5(1)	12/15
aSmD	**18**(11)	**1242**(773)	284(534)	1.1(0.9)	1.9(3)	1.9(3)	2.0(5)	13/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f20	16	851	38111	51362	54470	54861	55313	14/15
aLmC	6.8(5)	14(9)	**2.6**(2)	**2.1**(2)	**2.0**(2)	**2.0**(2)	**2.0**(2)	15/15
aLmD	**5.3**(9)	13(7)	3.4(4)	2.6(3)	2.5(3)	2.5(3)	2.4(2)	15/15
aSmC	7.6(6)	10(3)	12(11)	9.1(14)	8.6(5)	8.6(8)	8.5(12)	10/15
aSmD	6.0(4)	**6.7**(7)	13(29)	10(9)	9.0(13)	9.0(8)	8.9(9)	10/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f21	41	1157	1674	1692	1705	1729	1757	14/15
aLmC	4.2(2)	18(14)	23(37)	24(24)	23(18)	23(23)	23(22)	15/15
aLmD	**2.2**(1)	14(29)	**15**(12)	**15**(16)	**15**(26)	**15**(26)	**15**(23)	15/15
aSmC	3.3(3)	18(21)	18(27)	18(30)	18(11)	18(34)	18(34)	15/15
aSmD	2.2(2)	**13**(6)	16(30)	16(15)	16(25)	16(19)	16(8)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f22	71	386	938	980	1008	1040	1068	14/15
aLmC	13(3)	**36**(67)	125(243)	138(133)	134(46)	130(159)	127(176)	14/15
aLmD	6.9(17)	49(42)	115(98)	113(87)	110(185)	107(96)	105(168)	14/15
aSmC	**3.2**(4)	43(70)	124(216)	119(189)	115(378)	112(133)	110(195)	14/15
aSmD	9.4(2)	61(107)	**63**(122)	**72**(68)	**70**(46)	**68**(160)	**70**(21)	14/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f23	3.0	518	14249	27890	31654	33030	34256	15/15
aLmC	2.3(2)	43(63)	∞	∞	∞	∞	∞ *5e5*	0/15
aLmD	3.0(3)	75(52)	∞	∞	∞	∞	∞ *5e5*	0/15
aSmC	**1.7**(1)	54(46)	∞	∞	∞	∞	∞ *5e5*	0/15
aSmD	1.9(2)	**36**(49)	∞	∞	∞	∞	∞ *5e5*	0/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f24	1622	2.2e5	6.4e6	9.6e6	9.6e6	1.3e7	1.3e7	3/15
aLmC	5.8(4)	33(31)	∞	∞	∞	∞	∞ *5e5*	0/15
aLmD	**3.7**(4)	11(5)	∞	∞	∞	∞	∞ *5e5*	0/15
aSmC	4.1(2)	**10**(14)	∞	∞	∞	∞	∞ *5e5*	0/15
aSmD	4.7(3)	33(33)	∞	∞	∞	∞	∞ *5e5*	0/15

Table 2: Average running time (aRT in number of function evaluations) divided by the respective best aRT measured during BBOB-2009 in dimension 20. The aRT and in braces, as dispersion measure, the half difference between 10 and 90%-tile of bootstrapped run lengths appear for each algorithm and target, the corresponding best aRT in the first row. The different target Δf-values are shown in the top row. #succ is the number of trials that reached the (final) target $f_{\mathrm{opt}} + 10^{-8}$. The median number of conducted function evaluations is additionally given in *italics*, if the target in the last column was never reached. Entries, succeeded by a star, are statistically significantly better (according to the rank-sum test) when compared to all other algorithms of the table, with $p = 0.05$ or $p = 10^{-k}$ when the number k following the star is larger than 1, with Bonferroni correction by the number of instances. A ↓ indicates the same tested against the best algorithm of BBOB-2009.Best results are printed in bold.

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f1	43	43	43	43	43	43	43	15/15
aLmC	20(2)	51(12)	118(17)	306(37)	519(55)	951(60)	1375(51)	15/15
aLmD	29(4)	64(13)	142(15)	323(56)	563(33)	1018(30)	1471(54)	15/15
aSmC	**19**(6)	**47**(17)	**105**(11)	**218**(34)$^{\star 2}$	**376**(24)$^{\star 3}$	**675**(23)$^{\star 4}$	**981**(32)$^{\star 4}$	15/15
aSmD	25(6)	60(8)	138(31)	276(28)	455(36)	782(38)	1108(29)	15/15
f2	385	386	387	388	390	391	393	15/15
aLmC	66(7)	76(6)	89(4)	108(6)	132(8)	180(7)	226(7)	15/15
aLmD	74(7)	86(8)	98(6)	118(7)	144(4)	195(9)	243(5)	15/15
aSmC	**45**(4)$^{\star 4}$	**52**(6)$^{\star 3}$	**59**(5)$^{\star 3}$	**69**(7)$^{\star 4}$	**82**(8)$^{\star 3}$	**114**(19)$^{\star 3}$	**146**(3)$^{\star 4}$	15/15
aSmD	58(6)	66(6)	74(5)	85(4)	101(8)	138(11)	174(5)	15/15
f3	5066	7626	7635	7637	7643	7646	7651	15/15
aLmC	847(1037)	∞	∞	∞	∞	∞	∞ *2e6*	0/15
aLmD	635(1478)	∞	∞	∞	∞	∞	∞ *2e6*	0/15
aSmC	**95**(59)	∞	∞	∞	∞	∞	∞ *2e6*	0/15
aSmD	102(56)	∞	∞	∞	∞	∞	∞ *2e6*	0/15
f4	4722	7628	7666	7686	7700	7758	1.4e5	9/15
aLmC	∞	∞	∞	∞	∞	∞	∞ *2e6*	0/15
aLmD	∞	∞	∞	∞	∞	∞	∞ *2e6*	0/15
aSmC	∞	∞	∞	∞	∞	∞	∞ *2e6*	0/15
aSmD	∞	∞	∞	∞	∞	∞	∞ *2e6*	0/15
f5	41	41	41	41	41	41	41	15/15
aLmC	32(7)	41(8)	42(10)	42(10)	42(12)	42(10)	42(12)	15/15
aLmD	50(8)	62(18)	64(5)	65(20)	65(23)	65(21)	65(8)	15/15
aSmC	**30**(7)	**39**(11)	**40**(11)	**40**(11)	**40**(14)	**40**(15)	**40**(5)	15/15
aSmD	49(9)	62(20)	64(18)	64(19)	64(14)	64(22)	64(13)	15/15
f6	1296	2343	3413	4255	5220	6728	8409	15/15
aLmC	11(0.8)	10(0.5)	10(0.5)	10(0.6)	10(0.3)	11(0.3)	11(0.3)	15/15
aLmD	14(1)	12(0.9)	12(0.7)	12(0.4)	12(0.5)	13(0.4)	13(0.2)	15/15
aSmC	**7.0**(0.9)$^{\star 4}$	**6.1**(0.7)$^{\star 4}$	**5.7**(0.6)$^{\star 4}$	**5.7**(0.5)$^{\star 4}$	**5.7**(0.5)$^{\star 4}$	**6.1**(0.3)$^{\star 4}$	**6.2**(0.4)$^{\star 4}$	15/15
aSmD	10(0.7)	8.6(0.4)	8.0(0.6)	8.1(0.7)	8.1(0.6)	8.4(0.4)	8.5(0.4)	15/15
f7	1351	4274	9503	16523	16524	16524	16969	15/15
aLmC	3.0(0.8)	2.3(0.4)	1.6(0.3)	1.7(0.1)	1.7(0.2)	1.7(0.2)	1.7(0.1)	15/15
aLmD	3.7(1)	2.6(0.3)	1.7(0.2)	1.8(0.3)	1.9(0.2)	1.9(0.2)	1.9(0.4)	15/15
aSmC	**2.6**(0.7)	**1.7**(0.4)*	**1.2**(0.3)$^{\star 2}$	**1.1**(0.2)$^{\star 2}$	**1.1**(0.2)$^{\star 2}$	**1.1**(0.1)$^{\star 2}$	**1.2**(0.1)*	15/15
aSmD	3.6(1.0)	2.6(0.5)	1.6(0.2)	1.3(0.1)	1.3(0.1)	1.3(0.1)	1.4(0.1)	15/15
f8	2039	3871	4040	4148	4219	4371	4484	15/15
aLmC	∞	∞	∞	∞	∞	∞	∞ *2e6*	0/15
aLmD	∞	∞	∞	∞	∞	∞	∞ *2e6*	0/15
aSmC	∞	∞	∞	∞	∞	∞	∞ *2e6*	0/15
aSmD	∞	∞	∞	∞	∞	∞	∞ *2e6*	0/15
f9	1716	3102	3277	3379	3455	3594	3727	15/15
aLmC	∞*	∞*	∞*	∞*	∞*	∞*	∞ *2e6**	0/15
aLmD	∞	∞	∞	∞	∞	∞	∞ *2e6*	0/15
aSmC	∞	∞	∞	∞	∞	∞	∞ *2e6*	0/15
aSmD	∞	∞	∞	∞	∞	∞	∞ *2e6*	0/15
f10	7413	8661	10735	13641	14920	17073	17476	15/15
aLmC	3.3(0.4)	3.3(0.2)	3.1(0.2)	3.1(0.1)	3.4(0.2)	4.0(0.2)	5.1(0.1)	15/15
aLmD	3.9(0.2)	3.8(0.2)	3.6(0.3)	3.4(0.1)	3.8(0.1)	4.5(0.1)	5.5(0.1)	15/15
aSmC	**2.3**(0.3)$^{\star 4}$	**2.3**(0.4)$^{\star 4}$	**2.1**(0.2)$^{\star 4}$	**2.0**(0.1)$^{\star 4}$	**2.2**(0.2)$^{\star 3}$	**2.7**(0.1)$^{\star 4}$	**3.4**(0.1)$^{\star 4}$	15/15
aSmD	3.1(0.2)	3.0(0.1)	2.7(0.1)	2.5(0.2)	2.7(0.2)	3.2(0.1)	4.0(0.1)	15/15
f11	1002	2228	6278	8586	9762	12285	14831	15/15
aLmC	16(0.8)	8.4(0.2)	3.4(0.2)	3.2(0.2)	3.8(0.3)	4.5(0.2)	5.0(0.1)	15/15
aLmD	19(1.0)	9.4(0.5)	3.8(0.2)	3.6(0.2)	4.1(0.2)	4.9(0.2)	5.3(0.2)	15/15
aSmC	**10**(1.0)$^{\star 4}$	**5.4**(0.3)$^{\star 4}$	**2.2**(0.2)$^{\star 4}$	**2.0**(0.2)$^{\star 4}$	**2.3**(0.2)$^{\star 3}$	**2.8**(0.1)$^{\star 4}$	**3.2**(0.2)$^{\star 4}$	15/15
aSmD	13(1)	6.8(0.3)	2.8(0.1)	2.4(0.1)	2.7(0.2)	3.3(0.1)	3.8(0.2)	15/15
f12	1042	1938	2740	3156	4140	12407	13827	15/15
aLmC	39(2)	26(1)	27(2)	**47**(74)	**210**(527)	**1184**(1052)	∞ *2e6*	0/15
aLmD	43(2)	29(0.9)	55(100)	139(184)	456(248)	∞	∞ *2e6*	0/15
aSmC	**27**(1)$^{\star 4}$	**18**(1)$^{\star 4}$	**18**(2)*	53(50)	753(1602)	∞	∞ *2e6*	0/15
aSmD	33(3)	21(1)	22(6)	149(173)	656(691)	∞	∞ *2e6*	0/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f13	652	2021	2751	3507	18749	24455	30201	15/15
aLmC	24(3)	17(0.9)	19(0.9)	20(0.4)	4.7(0.1)	5.1(0.0)	5.3(0.1)	15/15
aLmD	28(1)	19(0.7)	21(0.7)	22(0.3)	5.1(0.0)	5.5(0.1)	5.7(0.1)	15/15
aSmC	**18**(3)$^{\star 3}$	**12**(1)$^{\star 3}$	**13**(0.6)$^{\star 4}$	**14**(0.4)$^{\star 4}$	**3.3**(0.1)$^{\star 4}$	**3.6**(0.1)$^{\star 4}$	**3.7**(0.1)$^{\star 4}$	15/15
aSmD	23(2)	14(0.8)	15(0.4)	16(0.3)	3.8(0.1)	4.0(0.1)	4.2(0.0)	15/15
f14	75	239	304	451	932	1648	15661	15/15
aLmC	14(11)	15(2)	22(4)	32(4)	28(3)	30(1)	4.8(0.2)	15/15
aLmD	17(7)	21(3)	26(2)	37(8)	31(2)	33(1)	5.1(0.2)	15/15
aSmC	**12**(5)	**12**(3)	**17**(3)$^{\star 3}$	**24**(3)$^{\star 3}$	**20**(2)$^{\star 2}$	**21**(1)$^{\star 4}$	**3.2**(0.1)$^{\star 4}$	15/15
aSmD	17(6)	17(1)	23(5)	31(4)	23(2)	24(1.0)	3.7(0.2)	15/15
f15	30378	1.5e5	3.1e5	3.2e5	3.2e5	4.5e5	4.6e5	15/15
aLmC	**7.5**(6)	∞	∞	∞	∞	∞	∞ *2e6*	0/15
aLmD	8.6(5)	∞	∞	∞	∞	∞	∞ *2e6*	0/15
aSmC	10(3)	3.6(0.4)	1.7(0.3)	1.7(0.3)	1.7(0.3)	1.3(0.2)	1.3(0.2)	15/15
aSmD	11(4)	**3.4**(0.2)	**1.6**(0.2)	**1.6**(0.2)	**1.6**(0.2)	**1.2**(0.1)	**1.2**(0.1)	15/15
f16	1384	27265	77015	1.4e5	1.9e5	2.0e5	2.2e5	15/15
aLmC	∞	∞	∞	∞	∞	∞	∞ *2e6*	0/15
aLmD	∞	∞	∞	∞	∞	∞	∞ *2e6*	0/15
aSmC	**1.0e4**(2e4)	**521**(503)	**185**(126)	**102**(116)	**76**(80)	**72**(127)	**65**(87)	2/15
aSmD	2.1e4(3e4)	∞	∞	∞	∞	∞	∞ *2e6*	0/15
f17	63	1030	4005	12242	30677	56288	80472	15/15
aLmC	**7.1**(3)	4.9(0.8)	4.9(1)	5.2(0.1)	3.1(0.2)	3.2(0.1)	3.3(0.1)	15/15
aLmD	12(4)	6.1(0.5)	5.5(1)	5.6(0.2)	3.3(0.1)	3.4(0.1)	3.5(0.1)	15/15
aSmC	8.2(4)	**4.0**(0.8)*	**3.9**(0.4)	**3.4**(0.2)$^{\star 3}$	**2.0**(0.1)$^{\star 3}$	**2.0**(0.1)$^{\star 4}$	**2.1**(0.0)$^{\star 4}$	15/15
aSmD	12(4)	5.3(0.5)	4.3(1)	3.8(0.2)	2.2(0.1)	2.3(0.1)	2.3(0.1)	15/15
f18	621	3972	19561	28555	67569	1.3e5	1.5e5	15/15
aLmC	4.4(1)	2.7(0.5)	2.6(0.3)	3.0(0.1)	1.8(0.0)	1.5(0.0)	2.0(0.0)	15/15
aLmD	5.4(1)	3.2(0.6)	2.8(0.2)	3.3(0.2)	1.9(0.0)	1.6(0.0)	2.1(0.0)	15/15
aSmC	**3.3**(1)	**1.9**(0.2)$^{\star 3}$	**1.7**(0.2)	**2.0**(0.1)$^{\star 2}$	**1.2**(0.1)$^{\star 3}$	**0.99**(0.0)$^{\star 4}$	**1.2**(0.0)$^{\star 4}$	15/15
aSmD	4.8(1)	2.6(0.3)	1.9(0.2)	2.2(0.1)	1.3(0.1)	1.1(0.0)	1.3(0.0)	15/15
f19	1	1	3.4e5	4.7e6	6.2e6	6.7e6	6.7e6	15/15
aLmC	408(132)	∞	∞	∞	∞	∞	∞ *2e6*	0/15
aLmD	518(150)	∞	∞	∞	∞	∞	∞ *2e6*	0/15
aSmC	**351**(136)	3.9e6(7e6)	13(14)	0.99(2)	0.74(0.7)	0.86(0.4)	1.1(0.9)	4/15
aSmD	455(151)	**1.6e6**(2e6)	**5.2**(5)	**0.51**(0.3)	**0.46**(0.2)	**0.48**(0.2)	**0.55**(0.6)	7/15
f20	82	46150	3.1e6	5.5e6	5.5e6	5.6e6	5.6e6	14/15
aLmC	15(5)	∞	∞	∞	∞	∞	∞ *2e6*	0/15
aLmD	22(9)	∞	∞	∞	∞	∞	∞ *2e6*	0/15
aSmC	**14**(3)	**8.9**(5)	∞	∞	∞	∞	∞ *2e6*	0/15
aSmD	20(12)	9.2(2)	∞	∞	∞	∞	∞ *2e6*	0/15
f21	561	6541	14103	14318	14643	15567	17589	15/15
aLmC	**3.7**(1.0)	188(153)	100(90)	99(85)	97(136)	92(93)	82(112)	11/15
aLmD	129(354)	**104**(150)	**53**(69)	**52**(103)	**51**(45)	**48**(70)	**43**(41)	14/15
aSmC	49(105)	120(137)	61(51)	60(62)	59(80)	55(71)	49(97)	14/15
aSmD	61(3)	126(270)	73(111)	72(90)	71(67)	67(90)	60(43)	13/15
f22	467	5580	23491	24163	24948	26847	1.3e5	12/15
aLmC	183(213)	171(200)	∞	∞	∞	∞	∞ *2e6*	0/15
aLmD	**24**(3)	285(255)	∞	∞	∞	∞	∞ *2e6*	0/15
aSmC	131(255)	**145**(76)	∞	∞	∞	∞	∞ *2e6*	0/15
aSmD	196(104)	177(317)	∞	∞	∞	∞	∞ *2e6*	0/15
f23	3.2	1614	67457	3.7e5	4.9e5	8.1e5	8.4e5	15/15
aLmC	4.0(7)	∞	∞	∞	∞	∞	∞ *2e6*	0/15
aLmD	2.9(1)	∞	∞	∞	∞	∞	∞ *2e6*	0/15
aSmC	**2.1**(2)	∞	∞	∞	∞	∞	∞ *2e6*	0/15
aSmD	3.3(2)	∞	∞	∞	∞	∞	∞ *2e6*	0/15
f24	1.3e6	7.5e6	5.2e7	5.2e7	5.2e7	5.2e7	5.2e7	3/15
aLmC	∞	∞	∞	∞	∞	∞	∞ *2e6*	0/15
aLmD	∞	∞	∞	∞	∞	∞	∞ *2e6*	0/15
aSmC	∞	∞	∞	∞	∞	∞	∞ *2e6*	0/15
aSmD	∞	∞	∞	∞	∞	∞	∞ *2e6*	0/15