Self-adaptive Search Equation-Based Artificial Bee Colony Algorithm with CMA-ES on the Noiseless BBOB Testbed*

Doğan Aydın
Dumlupinar University
Computer Engineering Dept. 43000
Kütahya, Turkey
dogan.aydin@dpu.edu.tr

Gürcan Yavuz
Dumlupinar University
Computer Engineering Dept. 43000
Kütahya, Turkey
gurcan.yavuz@dpu.edu.tr

ABSTRACT

Self-Adaptive Search Equation based Artificial Bee Colony (SSEABC) is a recent variant of Artificial Bee Colony (ABC) algorithm. SSEABC proposed three enhancements on the canonical ABC algorithm. These are the self-adaptive search equation selection strategy, hybridization with a local search procedure and incremental population size strategy. The performance of SSEABC is tested on CEC 2015 benchmark suite and ranked third within all participants of competition. In this paper, we benchmark SSEABC using the noise-free BBOB function testbed. We also compare SSEABC performance to PSO, ABC and GA algorithms.

CCS CONCEPTS

•Mathematics of computing → Probabilistic algorithms; •Theory of computation → Mathematical optimization; Random search heuristics; Theory of randomized search heuristics; •Computing methodologies → Search methodologies;

KEYWORDS

Artificial Bee Colony, Continuous Domains, Self-Adaptation, Benchmarking

ACM Reference format:
Doğan Aydın and Gürcan Yavuz. 2017. Self-adaptive Search Equation-Based Artificial Bee Colony Algorithm with CMA-ES on the Noiseless BBOB Testbed. In *Proceedings of GECCO '17 Companion, Berlin, Germany, July 15-19, 2017,* 8 pages.
DOI: http://dx.doi.org/10.1145/3067695.3084204

1 INTRODUCTION

Ever since the Artificial Bee Colony (ABC) algorithm came into existence [11], it has been used in solving continuous optimization problems. However, failure to produce successful results in some types of problems has led to the emergence of many improved ABC variants in recent years. Many of these algorithms have suggested enhancements over one or more of the steps of the ABC algorithm [1, 3]. A recent research [1] has shown that the best improvements can be made with changes to the employed bees and onlooker bees steps or with new extensions to the canonical ABC algorithm.

A recent ABC variant, Self-adaptive Search Equation based Artificial Bee Colony (SSEABC) [14], is focused on these remediation methods. The SSEABC algorithm solves the problem of finding the appropriate search equation in the employed bees and onlooker bees steps in a self-adaptive way. On the other hand, the algorithm has been improved with iteratively increasing the number of populations and using local search procedures.

SSEABC algorithm performance has been compared with ABC and many contemporary algorithms on CEC 2016 benchmark functions suite and it has been observed that we have obtained successful results [14]. In this paper, the performance of SSEABC algorithm on the BBOB functions testbed has been tested.

2 ALGORITHM PRESENTATION

SSEABC proposes three modifications on the original ABC algorithm to improve performance. These strategies are based on the self-adaptive search equation selection, hybridization with a local search procedure and increasing population size during execution.The pseudo-code of SSEABC is presented in Algorithm 1.

Self-adaptive search equation selection: In solving numerical optimization problems, the most important factor affecting the performance of ABC algorithm is the search equations that take place in the steps of employed bees and onlooker bees. In addition to the search equation, the number of dimensions considered to be changed is another important factor affecting the performance of the algorithm. When considering the structure of the problem and that is supposed to be solved; determining the appropriate search equation becomes a difficult task. Thus, in this study, a mechanism has been developed that determines the appropriate search equation among the various candidates. To do this, SSEABC has proposed a search equation pool which is filled with randomly generated search equations. The general template of candidate search equation is as seen in Algorithm 2.

The candidate search equations take the form of four terms with alternatives in Table 1 with M values. At initialization step of the algorithm, the pool, S, is filled by randomly generated search equation using Algorithm 2 and Table 1.Then, at each iteration a candidate search equation from the pool is used in the employed bees and onlooker bees steps. This process repeats until all candidates used in the pool. Throughout these steps, the number of success rates of the search equations that provide the update of the solution is increased. After all the candidate search equations in the pool are

*Submission deadline: March 31st.

GECCO '17 Companion, Berlin, Germany

DOI: http://dx.doi.org/10.1145/3067695.3084204

Algorithm 1 The Pseudo-code for the SSEABC algorithm

function SSEABCMain
  Initialization
  fill search equations pool, S, with randomly generated search equations
  $itr = 0$
  $isCompetitionNeeded = TRUE$
  while termination condition is not met **do**
    EmployedBeesStep($S[itr \pmod{ps}]$) ▷ it also counts number of successful updates with $S[itr \pmod{ps}]$
    OnlookerBeesStep($S[itr \pmod{ps}]$) ▷ it also counts number of successful updates with $S[itr \pmod{ps}]$
    ScoutBeesStep()
    X_{gbest} = getBestFoodSource()
    if $currentFES \geq tr * MAXFES$ & $isLocalSearchCallNeeded == TRUE$ **then**
      applyLocalSearch(X_{gbest})
    if $SN < SN_{max}$ and $itr \pmod{g} == 0$ **then** ▷ Incemental population size strategy
      add new solution to the current population by using Equation 2
    if all search equations are used **then** ▷ A part of the self-adaptive search equation determination strategy
      shrink the search equations pool size ,ps, with Equation 1
    $itr = itr + 1$
  return X_{gbest} as the best solution

Algorithm 2 The general form of the search equation

for $m = 1$ to M **do**
  select random dimension j ($1 \leq j \leq D$)
  $x_{i,j} = term_1 + term_2 + term_3 + term_4$

used in the employed bees and onlooker steps, ps, which is the size of the pool, is scaled down by the equation 1:

$$ps = \frac{ps^2}{itr_{MAX}} \tag{1}$$

where $MAXFES$ is the maximum number of function evaluations for one execution and $2 \times SN$ is the number of function evaluations at each iteration and where itr_{MAX} is the approximated value of the maximum number of iterations. itr_{MAX} is the approximated value because the incremental population size strategy in which SN is changing over time. Finally, when the algorithm finishes its execution, very few search equations, which are the appropriate ones, remain in the pool only.

hybridization with a local search procedure. : In SSEABC algorithm, bees move using the SSEABC rules and by the invocation of a local search procedure. Specifically, best-so-far solution is used as the initial solution a local search procedure is called from. The final solution found through local search becomes the new best-so-far solution if it is better than the initial solution. In SSEABC, the local search procedure is not called every iteration. The local search procedure is called only when it is expected that its invocation will result in an improvement of the-best-so-far solution. In previous implementation of SSEABC [14], competitive local search selection procedure was used. However, for the BBOB testbed, we used CMA-ES algorithm [12] as the local search procedure because competitive local search selection provides a wasteful use of function evaluations.

Table 1: Alternative options for each component in the generalized search equation. $x_{G,j}$, $x_{GD,j}$, $x_{SC,j}$, $x_{MD,j}$, $x_{WO,j}$ and $x_{AVE,j}$ are best-so-far, best-distance, second best, median, worst foods sources at dimension j, respectively. On the other hand, X_{r1} and X_{r2} are two randomly selected food source and $x_{AVE,j}$ refers to average positions of the food source at dimension j. ϕ_N can take two possible ranges: $[-1, -1]$ and $[-SF, SF]$ where SF is randomly selected positive real value. These ranges are decided randomly while creating each component of randomly generated search equation.

M	$term1$	$term2 \parallel terms3 \parallel terms4$
1	$x_{i,j}$	$\phi N(x_{i,j} - x_{G,j})$
k $(1 \leq k \leq D)$	$x_{G,j}$	$\phi N(x_{i,j} - x_{r1,j})$
$[t, k]$ $(1 \leq t < k \leq D)$	$x_{r1,j}$	$\phi N(x_{G,j} - x_{r1,j})$
		$\phi N(x_{r1,j} - x_{r2,j})$
		$\phi N(x_{i,j} - x_{GD,j})$
		$\phi N(x_{i,j} - x_{SC,j})$
		$\phi N(x_{i,j} - x_{MD,j})$
		$\phi N(x_{i,j} - x_{WO,j})$
		$\phi N(x_{SC,j} - x_{MD,j})$
		$\phi N(x_{MD,j} - x_{WO,j})$
		$\phi N(x_{G,j} - x_{WO,j})$
		$\phi N(x_{r1,j} - x_{MD,j})$
		$\phi N(x_{G,j} - x_{MD,j})$
		$\phi N(x_{r1,j} - x_{WO,j})$
		$\phi N(x_{SC,j} - x_{r1,j})$
		$\phi N(x_{i,j} - x_{AVE,j})$
		$\phi N(x_{r1,j} - x_{AVE,j})$
		$\phi N(x_{G,j} - x_{AVE,j})$
		$DoNotUse$

Incremental population size strategy: This strategy is very similar to the incremental social learning (ISL) framework [2, 4]. According to this strategy, SSEABC starts to work with a small population. During the algorithm execution, a new solution influenced by the best-so-far solution is added to the population after a certain number of iterations called growth period, g. This addition process continues until the maximum population value is reached. The solution to be newly added to the population is created using the equation 2:

$$\acute{x}_{new,j} = x_{new,j} + \varphi_{i,j}(x_{G,j} - x_{new,j}) \quad (2)$$

where $\acute{x}_{new,j}$ is the new solution to be added.

3 EXPERIMENTAL PROCEDURE

We have used the default parameter values for SSEABC and CMAES algorithms which were given in [14] and [12] respectively. A maximum of $10^4 D$ function evaluations was used. Every periodic $2500D$ function evaluations SSEABC restarts without forgetting the best-so-far solution.

4 CPU TIMING

In order to evaluate the CPU timing of the algorithm, we have run the SSEABC on the $f8$ without restarts 30 seconds and until a maximum budget equal to $1000D$ is reached. The C++ code was run on a Intel Xeon E5410 quadcore CPUs running at 2.33 GHz with 2 x 6 MB L2 cache and 8 GB RAM. The time per function evaluation for dimensions 2, 3, 5, 10, 20, 40 equals 0.0041, 0.0082, 0.0146, 0.627, 1.421, and 2, 751 seconds respectively.

5 RESULTS

Results from experiments according to [10] and [6] on the benchmark functions given in [5, 9] are presented in Figures 1, 2 and 3 and in Tables 2 and 3. The experiments were performed with COCO [8], version 2.0, the plots were produced with version 2.0.

The **average runtime (aRT)**, used in the figures and tables, depends on a given target function value, $f_t = f_{opt} + \Delta f$, and is computed over all relevant trials as the number of function evaluations executed during each trial while the best function value did not reach f_t, summed over all trials and divided by the number of trials that actually reached f_t [7, 13]. **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.

From the experiments, we observed that SSEABC solved 11 functions in dimension 5 and 5 functions in dimension 20 with 100% success rate. Over from dimension from 2 to 20, SSEABC solved $f1$, $f5$, $f6$, $f7$, $f8$, $f9$, $f11$, $f12$ and $f21$. It is also seen that $f4$, $f19$ and $f24$ are the most difficult problems for SSEABC. For both 5 and 20 dimensional problems, SSEABC can not reach the optimum result in any instance. On the other hand, when the problem size increases, the performance of the SSEABC algorithm decreases. As seen in 20-D results, 13 out of the 24 functions have not been solved to the hardest target 10^{-8}.

Comparison of SSEABC algorithm to PSO, ABC and GA in previous BBOB workshops are presented in Figure 2. We have seen that SSEABC outperforms PSO, ABC and GA for almost all functions. Moreover, SSEABC obtains better run-time performance than reference algorithms on the moderate, ill-conditioned and multi-modal functions. When the comparison results are examined, SSEABC for $f4$ and $f20$ seems to give bad results from ABC. Although SSEABC is an improved variant of the ABC algorithm, it is surprising at first glance that this situation has emerged. However, this is related to the fact that the entirely selected local search algorithm does not work well on these problems. The use of a certain amount of the function evaluations budget by CMA-ES yields this result.

6 CONCLUSION

In this paper, we present the benchmark results of SSEABC algorithm on BBOB functions testbed. We have also compared the performance of SSEABC to the data obtained by PSO, ABC and GA algorithms. The comparison results showed that SSEABC algorithm can outperforms the compared algorithms and it is very competitive to (1+1)-CMA-ES and BIPOP-CMA-ES in moderate and ill-conditioned functions.

REFERENCES

[1] Doğan Aydın. 2015. Composite artificial bee colony algorithms: From component-based analysis to high-performing algorithms. *Applied Soft Computing* 32 (2015), 266–285.

[2] Doğan Aydın, Tianjun Liao, Marco A Montes de Oca, and Thomas Stützle. 2011. Improving performance via population growth and local search: the case of the artificial bee colony algorithm. In *International Conference on Artificial Evolution (Evolution Artificielle)*. Springer, 85–96.

[3] Doğan Aydın, Gürcan Yavuz, and Thomas Stützle. 2017. ABC-X: a generalized, automatically configurable artificial bee colony framework. *Swarm Intelligence* 11, 1 (2017), 1–38.

[4] Marco A Montes de Oca, Doğan Aydın, and Thomas Stützle. 2011. An incremental particle swarm for large-scale continuous optimization problems: an example of tuning-in-the-loop (re) design of optimization algorithms. *Soft Computing* 15, 11 (2011), 2233–2255.

[5] S. Finck, N. Hansen, R. Ros, and A. Auger. 2009. *Real-Parameter Black-Box Optimization Benchmarking 2009: Presentation of the Noiseless Functions.* Technical Report 2009/20. Research Center PPE. http://coco.lri.fr/downloads/download15.03/bbobdocfunctions.pdf Updated February 2010.

[6] N. Hansen, A Auger, D. Brockhoff, D. Tušar, and T. Tušar. 2016. COCO: Performance Assessment. *ArXiv e-prints* arXiv:1605.03560 (2016).

[7] N. Hansen, A. Auger, S. Finck, and R. Ros. 2012. *Real-Parameter Black-Box Optimization Benchmarking 2012: Experimental Setup.* Technical Report. INRIA. http://coco.gforge.inria.fr/bbob2012-downloads

[8] N. Hansen, A. Auger, O. Mersmann, T. Tušar, and D. Brockhoff. 2016. COCO: A Platform for Comparing Continuous Optimizers in a Black-Box Setting. *ArXiv e-prints* arXiv:1603.08785 (2016).

[9] N. Hansen, S. Finck, R. Ros, and A. Auger. 2009. *Real-Parameter Black-Box Optimization Benchmarking 2009: Noiseless Functions Definitions.* Technical Report RR-6829. INRIA. http://coco.lri.fr/downloads/download15.03/bbobdocfunctions.pdf Updated February 2010.

[10] N. Hansen, T. Tušar, O. Mersmann, A. Auger, and D. Brockhoff. 2016. COCO: The Experimental Procedure. *ArXiv e-prints* arXiv:1603.08776 (2016).

[11] Dervis Karaboga and Bahriye Basturk. 2007. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of global optimization* 39, 3 (2007), 459–471.

[12] Václav Klemøs Petr Poøsík. 2012. Benchmarking the Differential Evolution with Adaptive Encoding on Noiseless Functions. In *GECCO 2012 (Companion)*, Terence Soule (Ed.). ACM.

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

[14] Gurcan Yavuz, Doğan Aydin, and Thomas Stützle. 2016. Self-adaptive search equation-based artificial bee colony algorithm on the CEC 2014 benchmark functions. In *Evolutionary Computation (CEC), 2016 IEEE Congress on*. IEEE, 1173–1180.

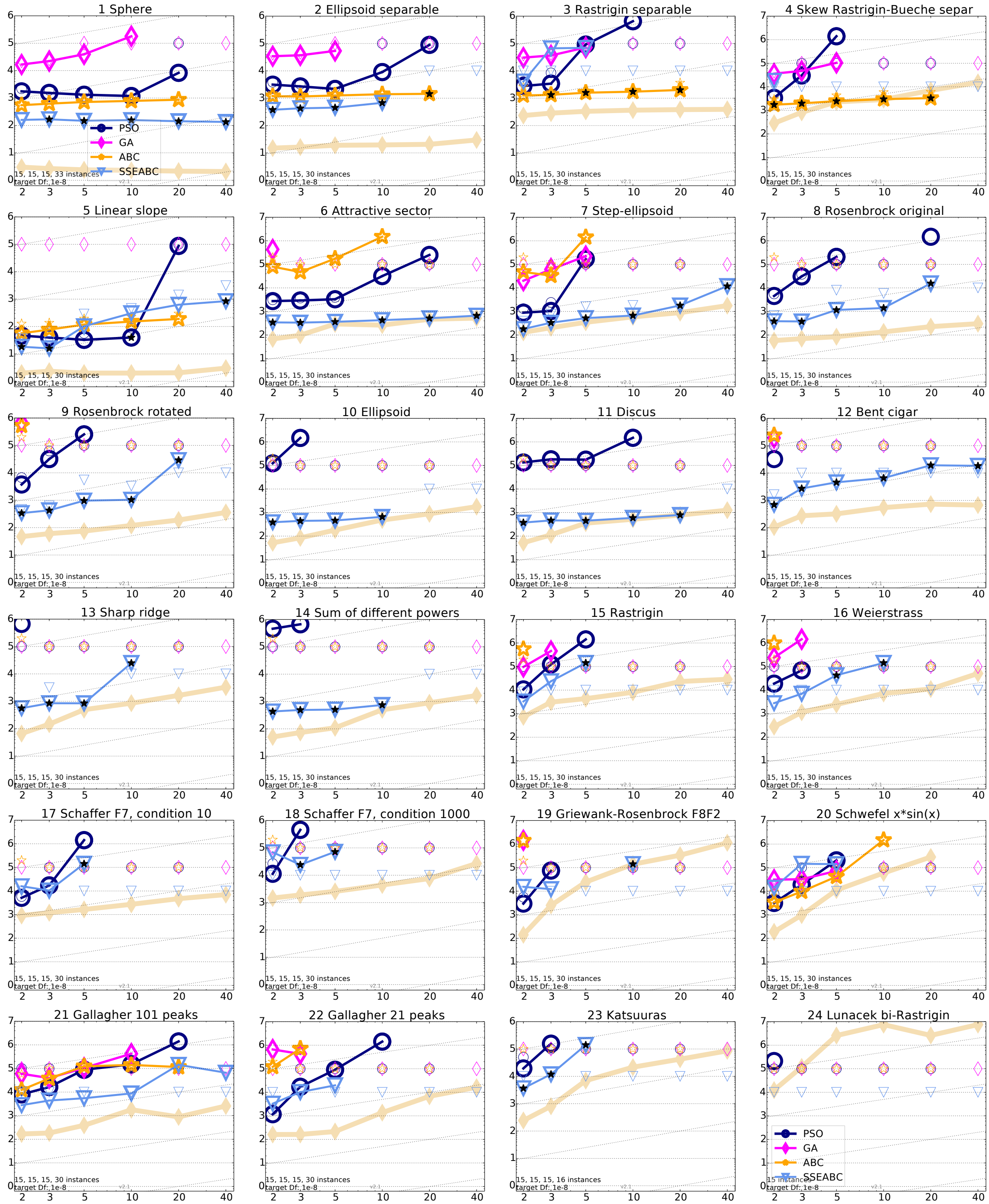

Figure 1: Average running time (aRT in number of f-evaluations as $\log_{10}$ value), divided by dimension for target function value 10^{-8} versus dimension. Slanted grid lines indicate quadratic scaling with the dimension. Different symbols correspond to different algorithms given in the legend of 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: ○: PSO, ◇: GA, ⋆: ABC, ▽: SSEABC

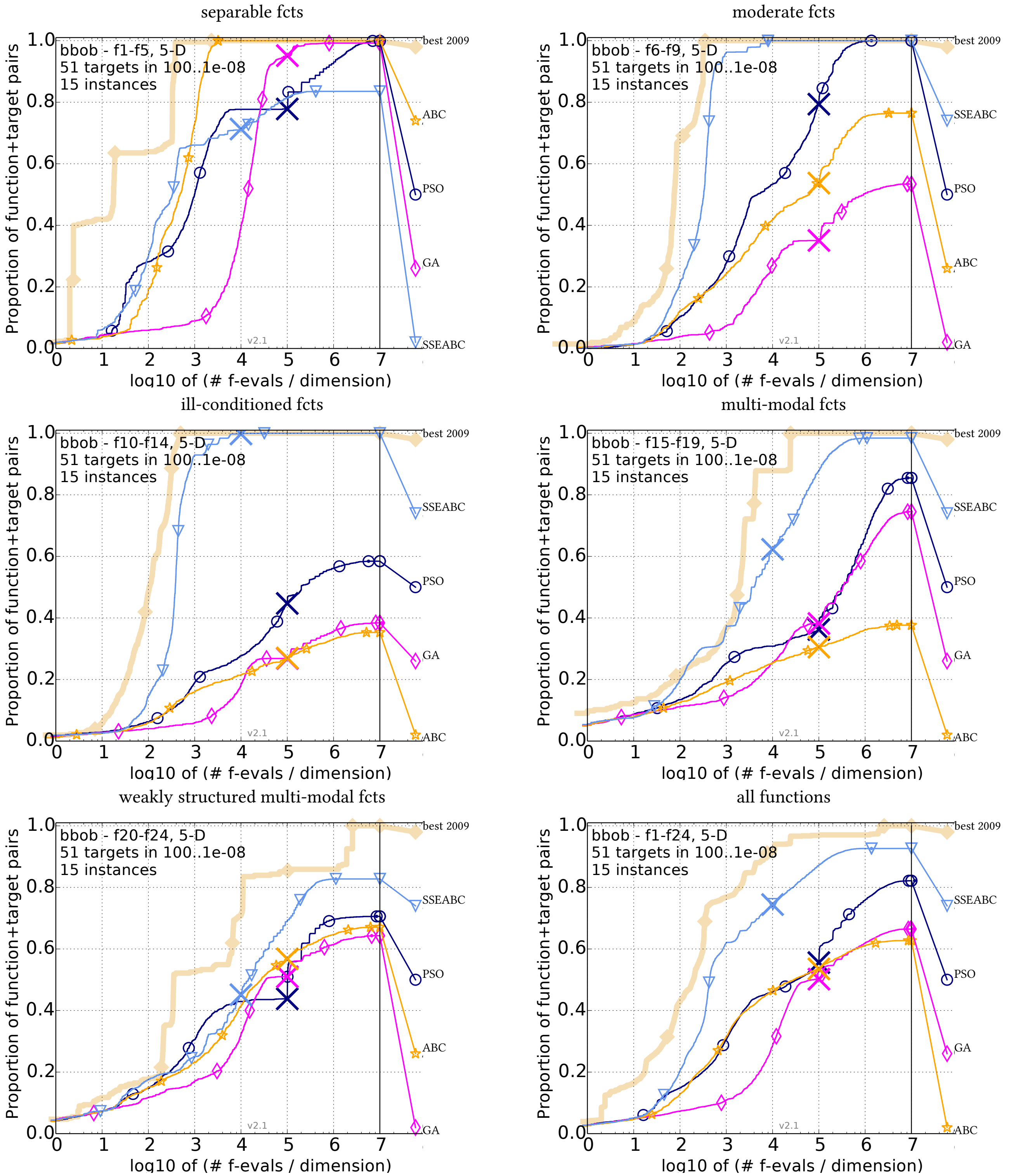

Figure 2: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for 51 targets with target precision in $10^{[-8..2]}$ for all functions and subgroups in 5-D. The "best 2009" line corresponds to the best aRT observed during BBOB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

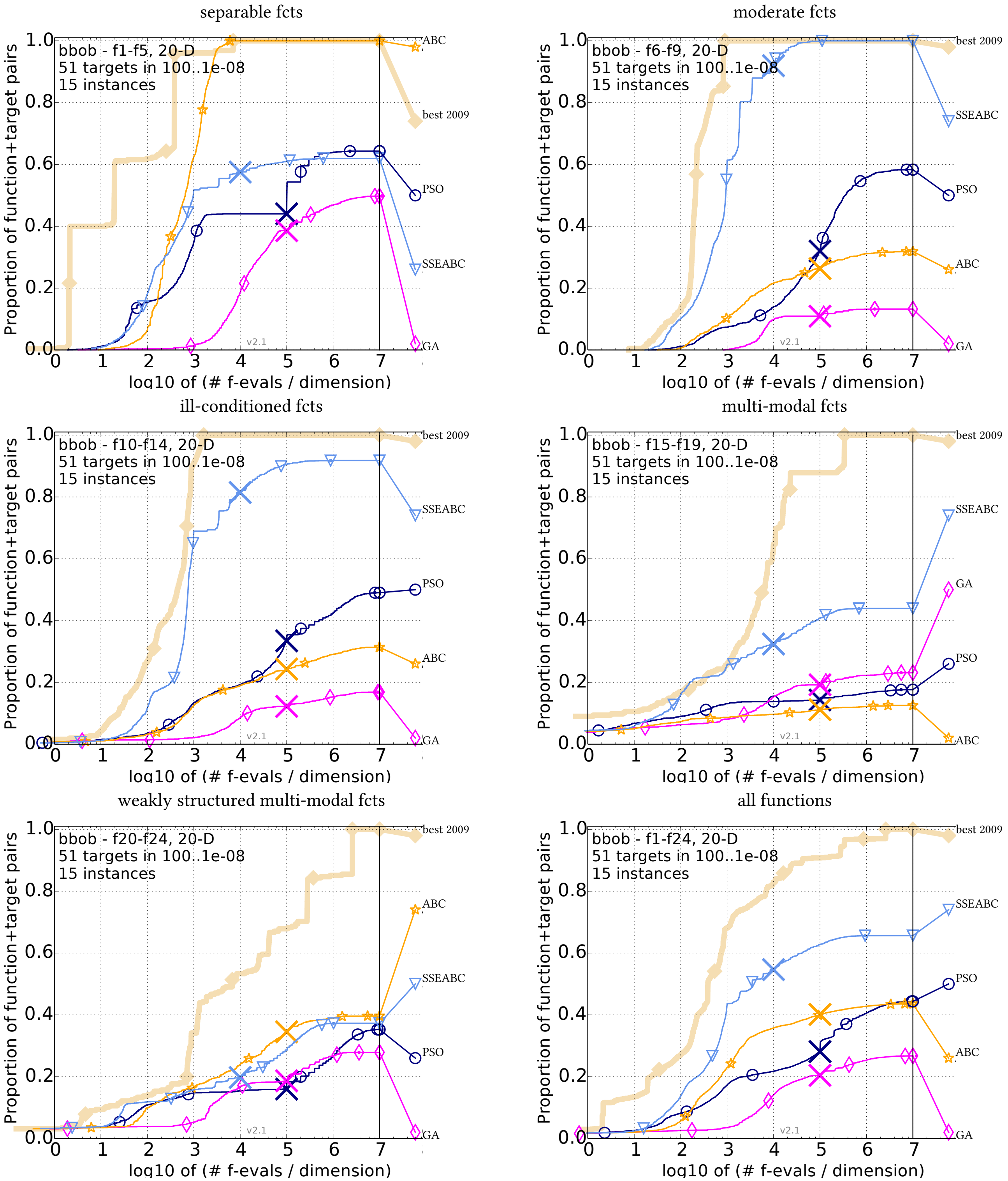

Figure 3: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for 51 targets with target precision in $10^{[-8..2]}$ for all functions and subgroups in 20-D. The "best 2009" line corresponds to the best aRT observed during BBOB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

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
PSO	**3.7**(2)	22(9)	56(18)	117(24)	185(15)	322(40)	458(67)	15/15
GA	8.7(16)	369(261)	1202(122)	2130(335)	2989(447)	5474(410)	8468(495)	13/15
ABC	12(14)	32(20)	63(22)	90(42)	124(24)	194(18)	259(14)	15/15
SSEABC	5.9(3)	**12**(2)$^{\star 2}$	**18**(0.9)$^{\star 3}$	**26**(3)$^{\star 4}$	**32**(3)$^{\star 4}$	**45**(2)$^{\star 4}$	**56**(4)$^{\star 4}$	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
PSO	32(8)	41(7)	49(6)	60(6)	68(11)	89(16)	106(13)	15/15
GA	334(55)	459(34)	609(100)	772(60)	1301(1434)	2156(4125)	2535(117)	13/15
ABC	**11**(3)	18(9)	26(8)	30(17)	38(10)	50(8)	62(7)	15/15
SSEABC	15(4)	**18**(4)	**19**(2)	**20**(2)	**21**(2)$^{\star 3}$	**22**(2)$^{\star 4}$	**23**(1)$^{\star 4}$	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
PSO	52(1)	55(3)	275(306)	275(305)	275(685)	276(304)	276(229)	8/15
GA	19(2)	18(1)	25(2)	34(3)	43(5)	112(80)	200(156)	11/15
ABC	**1.0**(0.5)	**1.5**(0.7)$^{\star 3}$	**1.8**(0.6)$^{\star 4}$	**2.4**(0.4)$^{\star 4}$	**2.7**(0.5)$^{\star 4}$	**3.6**(0.4)$^{\star 4}$	**4.4**(0.7)$^{\star 4}$	15/15
SSEABC	1.2(1)	9.2(7)	49(70)	74(122)	132(122)	132(83)	210(165)	2/15

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
PSO	3.0(0.9)	141(232)	4152(2296)	3988(4764)	3859(4403)	3719(5435)	3687(2562)	1/15
GA	18(3)	20(2)	26(1)	33(4)	41(6)	58(4)	185(70)	9/15
ABC	**1.1**(0.5)	**2.4**(0.5)$^{\star 3}$	**2.9**(2)$^{\star 4}$	**3.4**(1)$^{\star 4}$	**4.3**(2)$^{\star 3}$	**4.9**(0.9)$^{\star 4}$	**6.0**(1)$^{\star 4}$	15/15
SSEABC	2.7(3)	∞	∞	∞	∞	∞	∞ *5e4*	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
PSO	10(1)	**14**(2)	**16**(4)	**16**(7)	**16**(7)	**16**(7)	**16**(6)	15/15
GA	481(227)	2072(340)	3983(177)	6349(609)	9220(737)	1.7e4(1635)	3.4e4(2591)	0/15
ABC	32(34)	49(20)	58(30)	59(29)	59(32)	59(34)	59(29)	15/15
SSEABC	**6.6**(3)$^{\star 2}$	22(26)	34(32)	41(43)	46(29)	51(32)	51(68)	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
PSO	4.7(5)	9.0(3)	11(3)	12(3)	11(3)	10(2)	11(1)	15/15
GA	66(34)	148(61)	382(73)	3680(4345)	1.2e4(2e4)	∞	∞ *5e5*	0/15
ABC	4.9(3)	15(9)	365(36)	408(51)	619(665)	498(362)	507(705)	6/15
SSEABC	**2.0**(0.6)$^{\star 2}$	**1.9**(0.3)$^{\star 4}$	**2.0**(0.5)$^{\star 4}$	**1.8**(0.2)$^{\star 4}$	**1.6**(0.3)$^{\star 4}$	**1.2**(0.1)$^{\star 4}$	**1.2**(0.1)$^{\star 4}$	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
PSO	11(15)	9.5(14)	587(953)	475(487)	541(513)	541(644)	533(895)	6/15
GA	49(47)	35(8)	57(218)	245(265)	524(648)	524(415)	523(1024)	5/15
ABC	19(30)	16(8)	62(51)	464(514)	957(1649)	957(1331)	1359(2374)	1/15
SSEABC	**5.9**(3)	**1.9**(2)	**1.4**(0.6)$^{\star 3}$	**1.4**(1)$^{\star 3}$	**1.4**(0.6)$^{\star 4}$	**1.4**(0.8)$^{\star 4}$	**1.4**(1)$^{\star 4}$	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
PSO	13(5)	153(4)	201(54)	313(18)	467(364)	781(90)	1104(98)	7/15
GA	187(62)	837(2016)	∞	∞	∞	∞	∞ *5e5*	0/15
ABC	6.0(1)	12(13)	52(120)	449(875)	2510(1785)	∞	∞ *5e5*	0/15
SSEABC	**4.7**(1)	**5.6**(7)	**8.7**(1.0)$^{\star 2}$	**14**(51)$^{\star 3}$	**14**(9)$^{\star 4}$	**14**(1.0)$^{\star 4}$	**14**(8)$^{\star 4}$	15/15

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
PSO	24(11)	938(1013)	678(631)	794(1464)	1131(1699)	2363(2310)	2753(1061)	5/15
GA	418(128)	5.6e4(5e4)	∞	∞	∞	∞	∞ *5e5*	0/15
ABC	14(12)	69(66)	699(628)	3994(3662)	∞	∞	∞ *5e5*	0/15
SSEABC	**9.0**(3)	**11**(12)$^{\star 3}$	**9.0**(9)$^{\star 3}$	**14**(0.8)$^{\star 3}$	**15**(0.7)$^{\star 4}$	**14**(29)$^{\star 4}$	**13**(17)$^{\star 4}$	15/15

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
PSO	1741(4116)	3259(4409)	∞	∞	∞	∞	∞ *5e5*	0/15
GA	2375(3287)	∞	∞	∞	∞	∞	∞ *5e5*	0/15
ABC	2.1e4(2e4)	∞	∞	∞	∞	∞	∞ *5e5*	0/15
SSEABC	**3.5**(1)$^{\star 4}$	**3.1**(0.3)$^{\star 4}$	**3.0**(0.1)$^{\star 4}$	**3.0**(0.4)$^{\star 4}$	**3.1**(0.2)$^{\star 4}$	**2.5**(0.2)$^{\star 4}$	**2.5**(0.2)$^{\star 4}$	15/15

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
PSO	91(89)	235(144)	123(68)	140(37)	164(63)	243(170)	391(177)	8/15
GA	338(1814)	7124(4983)	9351(1e4)	∞	∞	∞	∞ *5e5*	0/15
ABC	160(289)	6082(5255)	9334(7701)	∞	∞	∞	∞ *5e5*	0/15
SSEABC	**8.5**(0.7)*	**7.2**(0.7)$^{\star 4}$	**2.1**(0.2)$^{\star 4}$	**1.8**(0.3)$^{\star 4}$	**1.6**(0.1)$^{\star 4}$	**1.4**(0.1)$^{\star 4}$	**1.3**(0.1)$^{\star 4}$	15/15

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
PSO	748(3477)	3747(7929)	5406(6065)	7886(1e4)	1.5e4(2e4)	∞	∞ *5e5*	0/15
GA	933(2397)	2447(4671)	1.9e4(1e4)	∞	∞	∞	∞ *5e5*	0/15
ABC	99(133)	547(878)	5846(4067)	∞	∞	∞	∞ *5e5*	0/15
SSEABC	**7.2**(8)$^{\star 3}$	**7.4**(13)$^{\star 3}$	**11**(2)$^{\star 3}$	**13**(3)$^{\star 3}$	**17**(11)$^{\star 3}$	**8.3**(10)$^{\star 4}$	**11**(4)$^{\star 4}$	12/15

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
PSO	1582(4308)	1.0e4(1e4)	2.8e4(2e4)	∞	∞	∞	∞ *5e5*	0/15
GA	243(34)	726(57)	4390(4528)	2.3e4(2e4)	∞	∞	∞ *5e5*	0/15
ABC	18(15)	187(310)	6613(8618)	∞	∞	∞	∞ *5e5*	0/15
SSEABC	**3.8**(1)$^{\star 2}$	**5.9**(2)$^{\star 4}$	**6.5**(4)$^{\star 4}$	**6.7**(1)$^{\star 4}$	**1.9**(0.4)$^{\star 4}$	**1.8**(0.2)$^{\star 4}$	**1.8**(0.2)$^{\star 4}$	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
PSO	**1.8**(2)	5.6(2)	15(2)	21(5)	29(8)	219(291)	∞ *5e5*	0/15
GA	2.1(1)	90(44)	267(55)	305(32)	349(66)	∞	∞ *5e5*	0/15
ABC	3.4(2)	11(10)	19(6)	29(11)	677(1236)	∞	∞ *5e5*	0/15
SSEABC	3.0(4)	**3.7**(0.7)	**4.4**(1)$^{\star 4}$	**4.4**(0.9)$^{\star 4}$	**4.8**(0.6)$^{\star 4}$	**5.7**(0.6)$^{\star 4}$	**4.6**(0.3)$^{\star 4}$	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
PSO	16(72)	221(253)	366(910)	359(437)	353(467)	342(590)	333(275)	1/15
GA	35(6)	91(175)	367(538)	361(242)	355(331)	345(429)	∞ *5e5*	0/15
ABC	15(6)	243(161)	∞	∞	∞	∞	∞ *5e5*	0/15
SSEABC	**1.2**(0.3)$^{\star 3}$	**0.93**(0.3)$^{\star 3}$	**1.0**(0.9)$^{\star 4}$	**1.1**(0.6)$^{\star 4}$	**1.7**(1.0)$^{\star 4}$	**7.9**(8)$^{\star 4}$	**10**(15)$^{\star 4}$	1/15

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
PSO	2.4(3)	6.2(7)	59(59)	55(28)	89(96)	300(333)	580(486)	0/15
GA	**2.1**(2)	84(65)	93(95)	71(114)	148(111)	621(345)	605(706)	0/15
ABC	2.3(2)	10(7)	95(93)	∞	∞	∞	∞ *5e5*	0/15
SSEABC	2.8(2)	**2.0**(3)*	**4.1**(6)	**1.4**(1)*	**1.9**(2)$^{\star 2}$	**3.8**(7)$^{\star 2}$	**18**(35)$^{\star 2}$	3/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f17	5.0	215	899	2861	3669	6351	7934	15/15
PSO	**3.4**(2)	169(583)	142(0.8)	156(350)	548(239)	514(433)	420(438)	1/15
GA	5.6(2)	46(13)	36(1)	52(4)	189(190)	550(1166)	∞ *5e5*	0/15
ABC	6.8(7)	15(7)	64(46)	1259(798)	∞	∞	∞ *5e5*	0/15
SSEABC	4.4(5)	**1.1**(0.3)$^{\star 3}$	**1.4**(3)$^{\star 2}$	**1.2**(1)$^{\star 2}$	**1.2**(0.9)$^{\star 3}$	**20**(18)*	**90**(109)*	1/15

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
PSO	2.3(2)	6.6(5)	113(95)	253(414)	∞	∞	∞ *5e5*	0/15
GA	22(18)	59(15)	34(3)	134(154)	∞	∞	∞ *5e5*	0/15
ABC	5.1(4)	27(27)	300(368)	∞	∞	∞	∞ *5e5*	0/15
SSEABC	**1.5**(0.5)	**2.2**(5)$^{\star 2}$	**0.89**(1)	**0.89**(0.1)$^{\star 3}$	**1.4**(1)$^{\star 4}$	**21**(21)$^{\star 4}$	**29**(73)$^{\star 4}$	2/15

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
PSO	35(30)	3381(3064)	2450(4336)	67(85)	60(105)	61(69)	61(32)	0/15
GA	35(23)	1.2e4(7520)	700(368)	68(48)	60(57)	∞	∞ *5e5*	0/15
ABC	**34**(48)	2898(1540)	3826(3344)	69(51)	∞	∞	∞ *5e5*	0/15
SSEABC	47(43)	**2379**(5798)	**158**(101)$^{\star 2}$	**3.4**(3)$^{\star 2}$	**6.0**(3)$^{\star 2}$	**6.0**(3)$^{\star 2}$	**5.9**(6)$^{\star 2}$	0/15

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
PSO	8.7(4)	3.1(1.0)	27(17)	20(22)	19(21)	19(11)	18(23)	5/15
GA	47(51)	21(6)	1.00(0.1)	1.0(0.2)	1.3(0.1)	2.6(0.3)	5.0(7)	11/15
ABC	7.2(3)	**1.5**(0.7)$^{\star 2}$	**0.55**(0.5)	**0.48**(0.5)	**0.58**(0.2)	**1.5**(2)	**2.6**(2)	15/15
SSEABC	**6.9**(3)	12(15)	18(25)	14(31)	13(25)	13(14)	13(14)	1/15

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
PSO	**2.0**(2)	379(541)	262(523)	260(370)	258(734)	255(506)	252(214)	8/15
GA	4.6(3)	5.5(2)	61(161)	68(232)	70(154)	139(219)	291(292)	8/15
ABC	3.2(2)	**1.8**(2)	**6.7**(8)	**10**(8)	13(14)	84(138)	265(149)	8/15
SSEABC	2.1(1)	5.0(8)	13(16)	13(15)	**13**(13)	**13**(10)	**15**(22)	12/15

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
PSO	**2.6**(2)	326(1296)	469(534)	450(1022)	439(496)	429(726)	422(704)	8/15
GA	6.0(4)	18(13)	387(402)	648(1160)	1489(1743)	6832(1e4)	∞ *5e5*	0/15
ABC	5.1(4)	**7.6**(13)	**35**(85)	237(384)	374(436)	3312(2932)	6897(4331)	0/15
SSEABC	3.2(0.6)	40(112)	51(51)	**60**(94)	**68**(66)	**81**(88)	**93**(201)	6/15

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
PSO	2.2(3)	20(15)	243(189)	∞	∞	∞	∞ *5e5*	0/15
GA	**1.5**(0.8)	59(71)	∞	∞	∞	∞	∞ *5e5*	0/15
ABC	2.2(2)	19(38)	∞	∞	∞	∞	∞ *5e5*	0/15
SSEABC	2.6(2)	**10**(10)	**6.1**(13)*	**25**(33)*	**22**(21)*	**21**(14)*	**21**(16)*	1/15

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
PSO	5.7(8)	∞	∞	∞	∞	∞	∞ *5e5*	0/15
GA	21(11)	∞	∞	∞	∞	∞	∞ *5e5*	0/15
ABC	13(11)	∞	∞	∞	∞	∞	∞ *5e5*	0/15
SSEABC	**2.8**(2)	∞	∞	∞	∞	∞	∞ *5e4*	0/15

Table 2: Average runtime (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 reference 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 $f_{opt} + 10^{-8}$. 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 of 110. A ↓ indicates the same tested against the best algorithm from BBOB 2009. Best results are printed in bold.

Data produced with COCO v2.1

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
PSO	22(8)	3399(1e4)	3446(30)	3507(20)	3563(31)	3680(1e4)	3808(1e4)	14/15
GA	876(70)	1905(150)	3205(297)	1.2e4(1e4)	3.1e4(4e4)	6.7e5(7e5)	∞ *2e6*	0/15
ABC	36(20)	66(39)	94(44)	136(51)	177(58)	292(21)	374(29)	15/15
SSEABC	**9.4**(0.8)$^{\star 3}$	**16**(1)$^{\star 4}$	**22**(2)$^{\star 4}$	**28**(2)$^{\star 4}$	**34**(1)$^{\star 4}$	**48**(2)$^{\star 4}$	**61**(3)$^{\star 4}$	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f2	385	386	387	388	390	391	393	15/15
PSO	4576(6494)	4571(7774)	4565(3877)	4560(5152)	4543(1e4)	4546(8955)	4537(6364)	8/15
GA	3228(7787)	6769(4016)	7.3e4(8e4)	∞	∞	∞	∞ *2e6*	0/15
ABC	**12**(3)$^{\star 4}$	**16**(3)$^{\star 4}$	**23**(4)$^{\star 4}$	**30**(6)$^{\star 4}$	**40**(8)$^{\star 3}$	**56**(5)$^{\star 4}$	**70**(4)$^{\star 4}$	15/15
SSEABC	37(6)	44(4)	52(3)	101(67)	242(128)	1031(1407)	∞ *2e5*	0/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f3	5066	7626	7635	7637	7643	7646	7651	15/15
PSO	∞	∞	∞	∞	∞	∞	∞ *2e6*	0/15
GA	29(2)	3709(3871)	∞	∞	∞	∞	∞ *2e6*	0/15
ABC	**1.5**(0.7)$^{\star 4}$	**2.7**(1)$^{\star 4}$	**3.0**(2)$^{\star 4}$	**3.2**(2)$^{\star 4}$	**3.6**(1)$^{\star 4}$	**4.1**(0.8)$^{\star 4}$	**4.9**(0.5)$^{\star 4}$	15/15
SSEABC	28(17)	∞	∞	∞	∞	∞	∞ *2e5*	0/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f4	4722	7628	7666	7686	7700	7758	1.4e5	9/15
PSO	5940(9530)	∞	∞	∞	∞	∞	∞ *2e6*	0/15
GA	65(3)	3751(4657)	∞	∞	∞	∞	∞ *2e6*	0/15
ABC	**2.0**(1)$^{\star 4}$	**4.2**(1)$^{\star 4}$	**5.6**(3)$^{\star 4}$	**6.0**(1)$^{\star 4}$	**6.3**(1)$^{\star 4}$	**7.3**(5)$^{\star 4}$	**0.45**(0.3)$^{\star 4}$	15/15
SSEABC	∞	∞	∞	∞	∞	∞	∞ *2e5*	0/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f5	41	41	41	41	41	41	41	15/15
PSO	4.3e4(5e4)	4.3e4(4e4)	4.3e4(9e4)	4.3e4(7e4)	4.3e4(7e4)	4.3e4(7e4)	4.3e4(6e4)	8/15
GA	2137(191)	4548(244)	7446(322)	1.1e4(482)	1.4e4(491)	2.2e4(780)	2.3e5(3e5)	0/15
ABC	68(15)	89(32)	**92**(40)	**92**(37)	**92**(23)	**92**(29)	**92**(24)	15/15
SSEABC	**25**(7)	**74**(29)	107(77)	145(106)	190(142)	245(195)	282(46)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f6	1296	2343	3413	4255	5220	6728	8409	15/15
PSO	1082(3109)	1009(1499)	705(735)	586(942)	502(619)	423(394)	577(477)	5/15
GA	1955(4251)	∞	∞	∞	∞	∞	∞ *2e6*	0/15
ABC	46(15)	453(260)	2587(2379)	∞	∞	∞	∞ *2e6*	0/15
SSEABC	**1.3**(0.2)$^{\star 4}$	**1.2**(0.1)$^{\star 4}$	**1.1**(0.1)$^{\star 4}$	**1.1**(0.1)$^{\star 4}$	**1.1**(0.1)$^{\star 4}$	**1.1**(0.1)$^{\star 4}$	**1.1**(0.0)$^{\star 4}$	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f7	1351	4274	9503	16523	16524	16524	16969	15/15
PSO	427(776)	∞	∞	∞	∞	∞	∞ *2e6*	0/15
GA	77(8)	∞	∞	∞	∞	∞	∞ *2e6*	0/15
ABC	251(319)	∞	∞	∞	∞	∞	∞ *2e6*	0/15
SSEABC	**0.94**(0.2)$^{\star 4}$	**5.0**(2)$^{\star 4}$	**3.4**(0.7)$^{\star 4}$	**2.1**(0.6)$^{\star 4}$	**2.1**(0.0)$^{\star 4}$	**2.1**(0.3)$^{\star 4}$	**2.1**(0.3)$^{\star 4}$	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f8	2039	3871	4040	4148	4219	4371	4484	15/15
PSO	90(41)	307(305)	350(436)	406(574)	466(205)	894(1065)	3276(5241)	1/15
GA	∞	∞	∞	∞	∞	∞	∞ *2e6*	0/15
ABC	**3.9**(2)	5.9(2)	10(2)	37(28)	353(285)	∞	∞ *2e6*	0/15
SSEABC	4.1(0.9)	**5.7**(6)	**5.9**(6)*	**8.4**(12)$^{\star 2}$	**12**(18)$^{\star 3}$	**34**(29)$^{\star 3}$	**68**(192)$^{\star 3}$	7/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f9	1716	3102	3277	3379	3455	3594	3727	15/15
PSO	670(245)	∞	∞	∞	∞	∞	∞ *2e6*	0/15
GA	∞	∞	∞	∞	∞	∞	∞ *2e6*	0/15
ABC	699(538)	∞	∞	∞	∞	∞	∞ *2e6*	0/15
SSEABC	**4.6**(0.6)$^{\star 4}$	**6.1**(8)$^{\star 4}$	**6.5**(0.4)$^{\star 4}$	**12**(4)$^{\star 4}$	**17**(29)$^{\star 4}$	**38**(38)$^{\star 4}$	**70**(71)$^{\star 4}$	4/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f10	7413	8661	10735	13641	14920	17073	17476	15/15
PSO	∞	∞	∞	∞	∞	∞	∞ *2e6*	0/15
GA	∞	∞	∞	∞	∞	∞	∞ *2e6*	0/15
ABC	∞	∞	∞	∞	∞	∞	∞ *2e6*	0/15
SSEABC	**1.8**(0.3)$^{\star 4}$	**1.9**(0.2)$^{\star 4}$	**1.7**(0.1)$^{\star 4}$	**1.7**(1)$^{\star 4}$	**3.7**(2)$^{\star 4}$	**13**(11)$^{\star 4}$	**80**(74)$^{\star 4}$	0/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f11	1002	2228	6278	8586	9762	12285	14831	15/15
PSO	143(56)	186(32)	109(13)	114(15)	132(14)	148(13)	2019(2596)	0/15
GA	2.9e4(3e4)	∞	∞	∞	∞	∞	∞ *2e6*	0/15
ABC	∞	∞	∞	∞	∞	∞	∞ *2e6*	0/15
SSEABC	**10**(0.4)$^{\star 4}$	**5.0**(0.2)$^{\star 4}$	**1.9**(0.1)$^{\star 4}$	**1.5**(0.0)$^{\star 4}$	**1.4**(0.0)$^{\star 4}$	**1.2**(0.0)$^{\star 4}$	**1.0**(0.0)$^{\star 4}$	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f12	1042	1938	2740	3156	4140	12407	13827	15/15
PSO	1703(2413)	∞	∞	∞	∞	∞	∞ *2e6*	0/15
GA	∞	∞	∞	∞	∞	∞	∞ *2e6*	0/15
ABC	26(5)	67(123)	429(391)	2184(3212)	7243(1e4)	∞	∞ *2e6*	0/15
SSEABC	**2.6**(4)$^{\star 4}$	**3.3**(3)$^{\star 4}$	**3.9**(2)$^{\star 4}$	**5.2**(1)$^{\star 4}$	**6.0**(7)$^{\star 4}$	**6.3**(2)$^{\star 4}$	**19**(22)$^{\star 4}$	6/15

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
PSO	6153(1e4)	6441(9154)	1.0e4(6543)	7993(8982)	1495(2000)	∞	∞ *2e6*	0/15
GA	5114(7690)	∞	∞	∞	∞	∞	∞ *2e6*	0/15
ABC	25(12)	73(78)	627(764)	3931(4428)	1555(1840)	∞	∞ *2e6*	0/15
SSEABC	**4.0**(3)$^{\star 3}$	**5.6**(9)$^{\star 3}$	**22**(19)$^{\star 3}$	**31**(47)$^{\star 3}$	**15**(21)$^{\star 2}$	∞	∞ *2e5*	0/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f14	75	239	304	451	932	1648	15661	15/15
PSO	6.7(3)	12(3)	20(3)	27(5)	54(16)	∞	∞ *2e6*	0/15
GA	277(55)	320(51)	477(46)	2898(3457)	∞	∞	∞ *2e6*	0/15
ABC	18(9)	18(11)	28(12)	53(7)	3379(2409)	∞	∞ *2e6*	0/15
SSEABC	**5.2**(1)	**3.2**(0.4)$^{\star 4}$	**4.0**(0.3)$^{\star 4}$	**4.5**(0.5)$^{\star 4}$	**4.3**(0.5)$^{\star 4}$	**6.4**(0.7)$^{\star 4}$	**3.8**(2)$^{\star 4}$	0/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f15	30378	1.5e5	3.1e5	3.2e5	3.2e5	4.5e5	4.6e5	15/15
PSO	∞	∞	∞	∞	∞	∞	∞ *2e6*	0/15
GA	∞	∞	∞	∞	∞	∞	∞ *2e6*	0/15
ABC	∞	∞	∞	∞	∞	∞	∞ *2e6*	0/15
SSEABC	**0.61**(0.0)$^{\star 4}$	∞	∞	∞	∞	∞	∞ *2e5*	0/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f16	1384	27265	77015	1.4e5	1.9e5	2.0e5	2.2e5	15/15
PSO	111(364)	∞	∞	∞	∞	∞	∞ *2e6*	0/15
GA	138(24)	1037(1853)	∞	∞	∞	∞	∞ *2e6*	0/15
ABC	13(12)	∞	∞	∞	∞	∞	∞ *2e6*	0/15
SSEABC	**1.3**(1)$^{\star 3}$	**0.40**(0.9)$^{\star 4}$	**4.9**(9)$^{\star 4}$	∞	∞	∞	∞ *2e5*	0/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f17	63	1030	4005	12242	30677	56288	80472	15/15
PSO	3.2(2)	2503(1944)	∞	∞	∞	∞	∞ *2e6*	0/15
GA	57(60)	92(7)	7071(8495)	∞	∞	∞	∞ *2e6*	0/15
ABC	34(24)	∞	∞	∞	∞	∞	∞ *2e6*	0/15
SSEABC	**3.1**(1.0)	**0.95**(0.2)$^{\star 4}$	**1.3**(2)$^{\star 4}$	**1.8**(2)$^{\star 4}$	**16**(22)$^{\star 4}$	**50**(48)$^{\star 4}$	∞ *2e5*	0/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f18	621	3972	19561	28555	67569	1.3e5	1.5e5	15/15
PSO	236(1613)	∞	∞	∞	∞	∞	∞ *2e6*	0/15
GA	76(11)	311(644)	∞	∞	∞	∞	∞ *2e6*	0/15
ABC	4.6e4(5e4)	∞	∞	∞	∞	∞	∞ *2e6*	0/15
SSEABC	**1.1**(0.4)$^{\star 4}$	**0.76**(0.1)$^{\star 4}$	**1.1**(0.9)$^{\star 4}$	**31**(41)$^{\star 4}$	**20**(17)$^{\star 4}$	∞	∞ *2e5*	0/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f19	1	1	3.4e5	4.7e6	6.2e6	6.7e6	6.7e6	15/15
PSO	382(192)	∞	∞	∞	∞	∞	∞ *2e6*	0/15
GA	1.4e4(3172)	6.5e5(8e4)	∞	∞	∞	∞	∞ *2e6*	0/15
ABC	2292(1664)	∞	∞	∞	∞	∞	∞ *2e6*	0/15
SSEABC	**329**(60)	**4.7e4**(4e4)$^{\star 4}$	∞	∞	∞	∞	∞ *2e5*	0/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f20	82	46150	3.1e6	5.5e6	5.5e6	5.6e6	5.6e6	14/15
PSO	17(4)	50(65)	∞	∞	∞	∞	∞ *2e6*	0/15
GA	502(60)	2.8(0.3)	∞	∞	∞	∞	∞ *2e6*	0/15
ABC	16(5)	**0.12**(0.1)$^{\star 3}$	∞	∞	∞	∞	∞ *2e6*	0/15
SSEABC	**6.9**(1.0)$^{\star 4}$	62(63)	∞	∞	∞	∞	∞ *2e5*	0/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f21	561	6541	14103	14318	14643	15567	17589	15/15
PSO	1785(2675)	4281(4434)	1986(1985)	1956(1362)	1913(2527)	1799(1831)	1593(2047)	1/15
GA	90(41)	625(612)	398(426)	397(455)	904(1162)	∞	∞ *2e6*	0/15
ABC	5.0(3)	**22**(52)	**25**(45)	**26**(46)	**27**(26)	**35**(59)	**85**(104)	8/15
SSEABC	**4.6**(5)	29(12)	203(152)	200(206)	195(260)	184(167)	163(316)	1/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f22	467	5580	23491	24163	24948	26847	1.3e5	12/15
PSO	**5.0**(10)	411(897)	∞	∞	∞	∞	∞ *2e6*	0/15
GA	110(39)	1452(718)	∞	∞	∞	∞	∞ *2e6*	0/15
ABC	10(6)	**47**(71)	**77**(57)	**578**(559)	∞	∞	∞ *2e6*	0/15
SSEABC	70(298)	77(105)	∞	∞	∞	∞	∞ *2e5*	0/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f23	3.0	1614	67457	3.7e5	4.9e5	8.1e5	8.4e5	15/15
PSO	2.4(2)	1554(1433)	∞	∞	∞	∞	∞ *2e6*	0/15
GA	1.8(0.7)	4570(2790)	∞	∞	∞	∞	∞ *2e6*	0/15
ABC	**1.2**(0.9)	78(63)	∞	∞	∞	∞	∞ *2e6*	0/15
SSEABC	2.1(2)	**31**(24)	**4.9**(5)$^{\star 2}$	∞	∞	∞	∞ *2e5*	0/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f24	1.3e6	7.5e6	5.2e7	5.2e7	5.2e7	5.2e7	5.2e7	3/15
PSO	∞	∞	∞	∞	∞	∞	∞ *2e6*	0/15
GA	∞	∞	∞	∞	∞	∞	∞ *2e6*	0/15
ABC	∞	∞	∞	∞	∞	∞	∞ *2e6*	0/15
SSEABC	∞	∞	∞	∞	∞	∞	∞ *2e5*	0/15

Table 3: Average runtime (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 reference 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 of 110. A ↓ indicates the same tested against the best algorithm from BBOB 2009. Best results are printed in bold.

Data produced with COCO v2.1