Benchmarking SHADE algorithm enhanced with model based optimization on the BBOB noiseless testbed

ABSTRACT

In this paper we evaluate the SHADE-LM algorithm on the BBOB noiseless testbed. The algorithm hybridizes the SHADE algorithm with a model based optimization. This hybridization is performed in a transparent manner for both optimizers, with SHADE having access to the samples provided by model based optimization, and models of square functions are fitted on the current population. The paper compares this extended version with the performance of the version of SHADE by Tanabe and Fukunaga.

CCS CONCEPTS

•**Computing methodologies → Continuous space search;**

KEYWORDS

SHADE, Model based optimization, Black-box optimization

ACM Reference format:

. 2021. Benchmarking SHADE algorithm enhanced with model based optimization on the BBOB noiseless testbed. In *Proceedings of The Genetic and Evolutionary Computation Conference 2021, Lille, France, July 10–14, 2021 (GECCO '21),* 8 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

R-SHADE [9] algorithm has been proposed as one of the more successful modification of a Differential Evolution, following the path of adapting the scale and cross-over probability factors, employing the archive of previous best samples, utilizing the current-to-best position update and restart mechanism based on population locations or values spread.

Hybridizing plain Particle Swarm Optimization (PSO) and Differential Evolution (DE) algorithms with model based optimizers has been proved to improve their performance [7, 11, 12].

In this paper we utilize Generalized Adaptive Particle Swarm Optimization (GAPSO) framework [7, 10] and implement within it a version of SHADE [9] and a model based optimizer [12].

2 GAPSO FRAMEWORK

The concept of GAPSO framework is to allow for hybridization of optimization algorithms, in a way which will be transparent to the hybridized methods. This approach comes from the observation that methods such as PSO or DE, need only a minimal amount of information (i.e. locations and values of the previously sampled locations). Please note that PSO's velocity fits into this need as it is a difference between previous and current location. Therefore an algorithm which stores current, previous and best location for each individual (particle) allows to employ sampling of both DE and PSO based algorithms, in a transparent manner from the point of view of an individual (particle).

3 SHADE-LM ALGORITHM

SHADE-LM hybridizes the utilization of population based SHADE algorithm, based on [9] and model based optimizer utilizing square functions [12].

3.1 SHADE

SHADE is a form of population based Differential Evolution algorithm utilizing:

- archive of samples which were previous-best
- list of adaptable cross-over and scale probabilities
- current-to-pbest type of DE mutation operator (1)

$$u^{(i)} = x^{(i)} + F_{it} * (x^{(pBest)} - x^{(i)}) + F_{it} * (x^{(rand1)} - x^{(rand2)}) \quad (1)$$

Single iteration of the SHADE pseudocode is given in Algorithm 1.

Algorithm 1 Single iteration of SHADE

$F_{it} \leftarrow SelectScaleFactorFromSlot(slot)$
$cp_{it} \leftarrow SelectCrossOverFactor(slot)$
for $i \in 1topop.size$ **do**
  $F_i \leftarrow SampleFromCauchyDistribution(F_{it}, 0.1)$
  $cp_i \leftarrow SampleFromNormalDistribution(cp_{it}, 0.1)$
  $x^{(pBest)} \leftarrow SelectOneOfPBestIndividuals(pBestRatio)$
  $x^{(rand1)} \leftarrow SelectIndividualFromCurrentPopulation()$
  $x^{(rand2)} \leftarrow SelectIndividualFromCurrentPopulationOrArchive()$
  $u^{(i)} \leftarrow GetSample(x^{(pBest)}, x^{(rand1)}, x^{(rand2)}, x^{(i)}, F_i)$
  $y^{(i)} \leftarrow ApplyCrossOver(x^{(pBest)}, u^{(i)}, cp_i)$
  if $f(y^{(i)}) < f(x^{(i)})$ **then**
    $PushToArchive(x^{(i)})$
    $x^{(i)} \leftarrow y^{(i)}$
    $StoreSuccessfulFactors(F_i, cp_i)$
  end if
end for
$AdaptScaleAndCrossOverFactors()$
$slot \leftarrow slot + 1$

3.2 Model based optimizers

Model based optimizer fits the linear combination of a_i, b_i, c coefficients for a simple N-dimensional square function (2), or if the population size allows it a full N-dimensional square function (3).

$$\hat{f}_{simple}(x) = \sum_{i=1}^{N} (a_i x_i^2 + b_i x_i) + c \quad (2)$$

GECCO '21, Lille, France

DOI: 10.1145/nnnnnnn.nnnnnnn

$$\hat{f}_{full}(x) = \sum_{i=1}^{N}\left(b_i x_i + \sum_{j=1}^{i}(a_{i,j}x_i x_j)\right) + c \quad (3)$$

Model is fitted on the samples already gathered during the optimization process, regardless of their source. In the case of SHADE-LM models are fitted on the current population. If the population size is big enough a full square model is fitted, otherwise the simple one is chosen.

For simplicity the system of linear equations (4) for the coefficients of simple square model is given.

$$\begin{bmatrix} (x_1^{(1)})^2 & x_1^{(1)} & \dots & 1 \\ \vdots & \vdots & \vdots & \vdots \\ (x_1^{(pop.size)})^2 & x_1^{(pop.size)} & \dots & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ b_1 \\ \vdots \\ a_N \\ b_N \\ c \end{bmatrix} = \begin{bmatrix} f(x^{(1)}) \\ \vdots \\ f(x^{(pop.size)}) \end{bmatrix} \quad (4)$$

After a model is fitted the optimizer samples the stationary point x_θ of the model $\hat{f}$ function by solving the system of linear equations for coefficients of first derivatives of $\hat{f}$.

$$\begin{bmatrix} 2a_{1,1} & \dots & a_{N,1} \\ \vdots & \ddots & \vdots \\ a_{N,1} & \dots & 2a_{N_N} \end{bmatrix} \begin{bmatrix} x_{\theta 1} \\ \vdots \\ x_{\theta N} \end{bmatrix} = \begin{bmatrix} -b_1 \\ \vdots \\ -b_N \end{bmatrix} \quad (5)$$

3.3 Proposed algorithm

Pseudocode of the proposed algorithm is given in Algorithm 2

Algorithm 2 SHADE-LM pseudocode

```
InitializePopulation()
while OptimizationBudgetIsLeft() do
    ModelIndividualsSet ← SelectIndividualsForModelApplication()
    SelectSHADEFactors()
    for Individual in Population do
        if Individual ∈ ModelIndividualsSet then
            y ← GetSampleAsModelFunctionStationaryPoint()
        else
            y ← GetSampleAsInSHADEAndStoreShadeFactors()
        end if
        if f(y) < f(Individual.x) then
            PushToArchive(Individual.x)
            Individual.x ← y^(i)
        end if
    end for
    AdaptSHADEFactors()
    if ShouldBeRestarted(Population) then
        ReInitializePopulation()
        ResetSHADE()
    end if
end while
```

4 EXPERIMENTAL PROCEDURE

We have run 2 configurations of our proposed approach, with the settings as given in the 1. Both configuration utilize the same restart procedure, relying on population values or location convergence below a certain threshold or no improvements in the global best value for a certain amount of evaluations. Both configurations rely on the SHADE algorithm, configured roughly like its original version [9], with the exception of omitting population size decrease during a single run of the algorithm. Both configurations utilize the same proportion of samples taken from square function model optimum. The difference between SHADE-LM and SHADE-LM-POP4-to-10, is within the management of population size. SHADE-LM employs a large population of 10× function dimensionality (D), while SHADE-LM-POP4-to-10 gradually increases its population from $4D$ to $10D$ gradually increasing it after algorithm restart by 1.2 factor.

Table 1: Settings of the SHADE-LM algorithm

General settings	
Optimization budget	$10^6 \times D$
Specific SHADE-LM settings	
Population size	$10D$
Specific SHADE-LM-POP4-to-10 settings	
Population size	$4D$ - $10D$
Population size increase after restart	1.2
Model based optimizer parameters	
Model based optimizer use count	0.05× pop. size
Restart parameters	
No improvement evaluations	$5000D$
Values convergence	10^{-12}
Locations convergence	10^{-12}
SHADE parameters	
Initial cross-over probability	0.9
Initial mutation scaling factor	0.38
Parameters slots	11
pBest count	0.11× pop. size
Archive size	0.12× pop. size

5 CPU TIMING

In order to evaluate the CPU timing of the algorithm, we have run the SHADE-LM on the bbob test suite [5] with restarts for a maximum budget equal to $10^6 D$) function evaluations according to [6]. The Java code was run on single core of a Windows Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz. The time per function evaluation for dimensions 2, 3, 5, 10, 20 equals 1.70×10^{-5}, 1.08×10^{-5}, 1.49×10^{-5}, 2.42×10^{-5}, and 5.20×10^{-5} seconds respectively.

6 RESULTS

Results from experiments according to [6] and [2] on the benchmark functions given in [1, 5] are presented in Figures 1, 2 and 3 and in Tables 2 and 3. The experiments were performed with COCO [4], version 2.3, the plots were produced with version 2.4.

The **expected runtime (ERT)**, 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 [3, 8]. **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.

Proposed SHADE-LM and SHADE-LM-POP4-to-10 configurations, improved the overall performance of the original R-SHADE algorithm. Original R-SHADE was found to be definitely better only for $f21$ and $f22$ function on $20D$.

Additionally, we have observed that starting with a smaller population size of $4D$ improves the performance for low optimization budget, up to 10^3 $timesD$. Setting population size to its final value of $10D$ proves beneficial in the long run for the optimization budgets higher than 10^4 $timesD$, especially for dimensions 10 and 20.

For future work we plan to include within this hybrid also the CMA-ES optimizer which poses an additional challenge of adapting CMA-ES' covariance matrix and σ on the basis of other algorithms samples. Initial experiments proved it to be non-trivial as CMA-ES is easily destabilized after recalculating those parameters on the basis of external samples.

REFERENCES

[1] 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.

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

[3] 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

[4] N. Hansen, A. Auger, R. Ros, O. Mersmann, T. Tušar, and D. Brockhoff. 2020. COCO: A Platform for Comparing Continuous Optimizers in a Black-Box Setting. *Optimization Methods and Software* (2020). DOI: http://dx.doi.org/https://doi.org/10.1080/10556788.2020.1808977

[5] 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.

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

[7] Michał Okulewicz, Mateusz Zaborski, and Jacek Mańdziuk. 2020. Generalized Self-Adapting Particle Swarm Optimization algorithm with archive of samples. (2020). https://arxiv.org/abs/2002.12485

[8] Kenneth Price. 1997. Differential evolution vs. the functions of the second ICEO. In *Proceedings of the IEEE International Congress on Evolutionary Computation.* IEEE, Piscataway, NJ, USA, 153–157. DOI: http://dx.doi.org/10.1109/ICEC.1997.592287

[9] R. Tanabe and A. S. Fukunaga. 2014. Improving the search performance of SHADE using linear population size reduction. In *2014 IEEE Congress on Evolutionary Computation (CEC).* 1658–1665. DOI: http://dx.doi.org/10.1109/CEC.2014.6900380

[10] Mateusz Uliński, Adam Żychowski, Michał Okulewicz, Mateusz Zaborski, and Hubert Kordulewski. 2018. Generalized Self-adapting Particle Swarm Optimization Algorithm. In *International Conference on Parallel Problem Solving from Nature.* Springer, 29–40.

[11] Mateusz Zaborski, M Okulewicz, and Jacek Mandziuk. 2019. Generalized Self-Adapting Particle Swarm Optimization algorithm with model-based optimization enhancements. In *2nd PP-RAI Conference (PPRAI-19).* 380–383.

[12] Mateusz Zaborski, Michał Okulewicz, and Jacek Mańdziuk. 2020. Analysis of statistical model-based optimization enhancements in Generalized Self-Adapting Particle Swarm Optimization framework. *Foundations of Computing and Decision Sciences* 45 (2020).

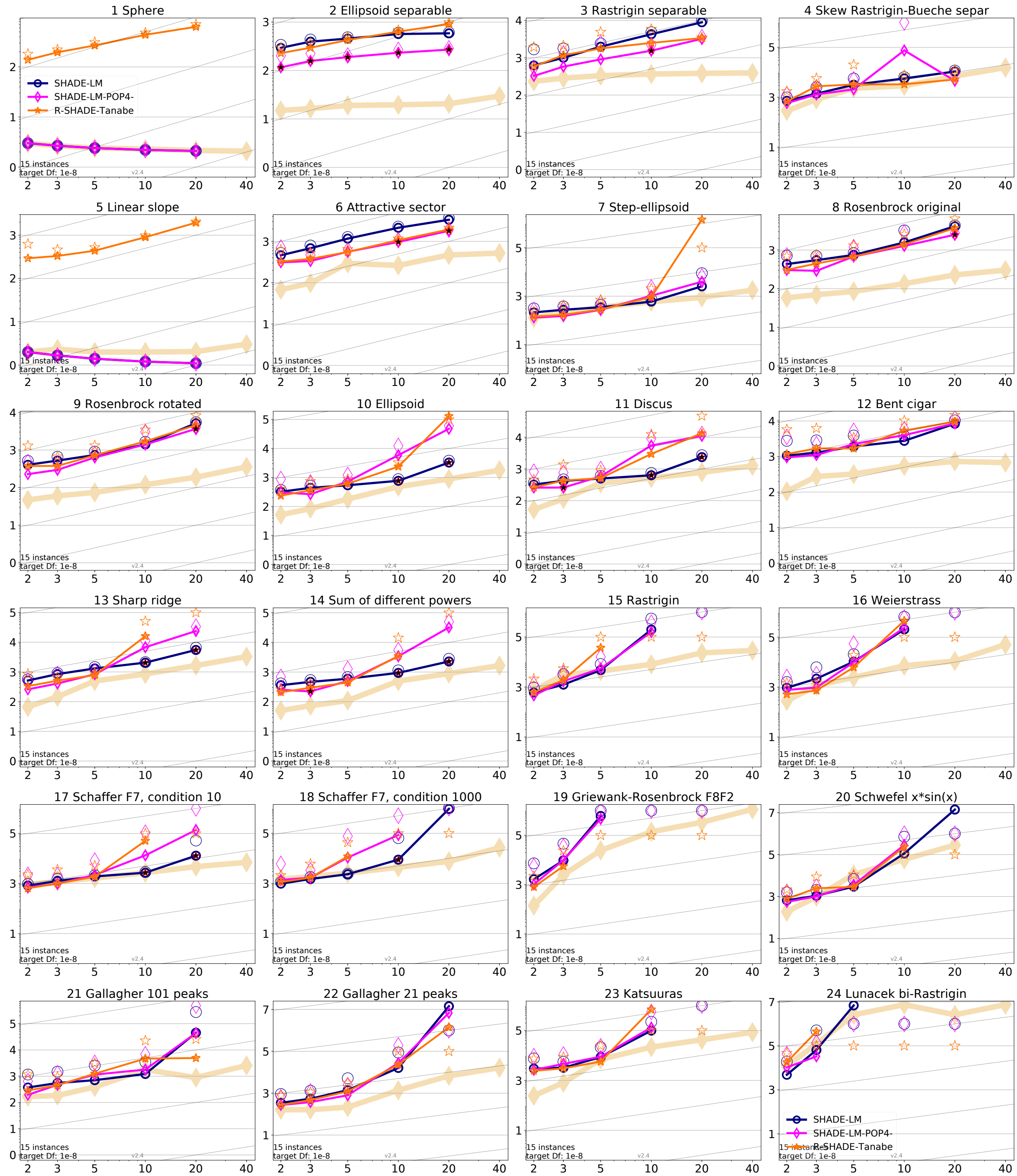

Figure 1: Expected running time (ERT in number of f-evaluations as $\log_{10}$ value), divided by dimension for target function value 10^{-8} versus dimension. Slanted grid lines indicate quadratic scaling with the dimension. Different symbols correspond to different algorithms given in the legend of 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: ○: R-SHADE-Tanabe, ◇: SHADE-LM, ⋆: SHADE-LM-POP4-to-10

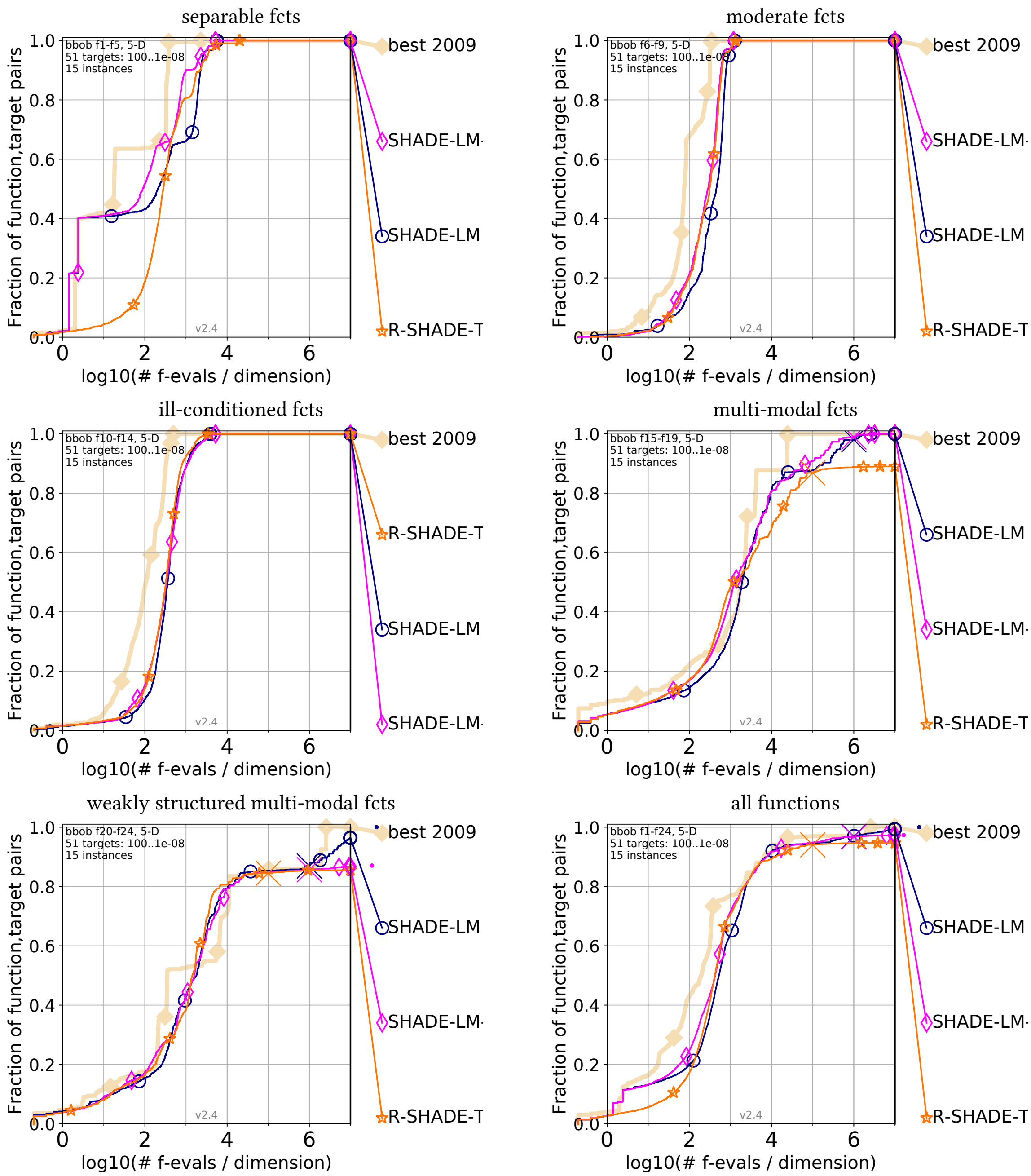

Figure 2: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for 51 targets with target precision in $10^{[-8..2]}$ for all functions and subgroups in 5-D. As reference algorithm, the best algorithm from BBOB 2009 is shown as light thick line with diamond markers.

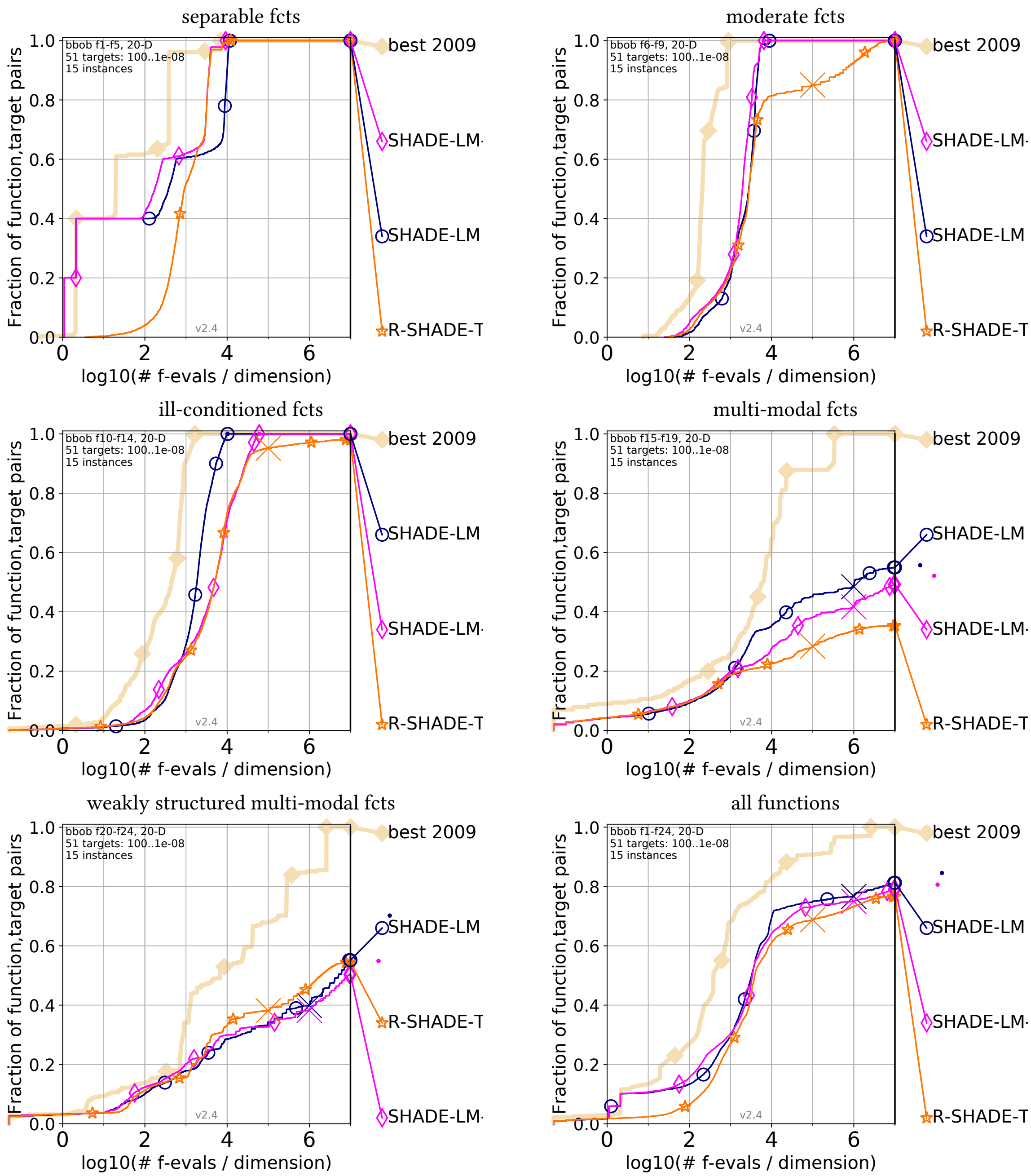

Figure 3: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for 51 targets with target precision in $10^{[-8..2]}$ for all functions and subgroups in 20-D. As reference algorithm, the best algorithm from BBOB 2009 is shown as light thick line with diamond markers.

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
SHADE-L	1.1(0.1)	**1**(0)	**1**(0)	**1**(0)	**1**(0)	**1**(0)	**1**(0)	15/15
SHADE-L	**1.0**(0.2)	**1**(0)	**1**(0)	**1**(0)	**1**(0)	**1**(0)	**1**(0)	15/15
R-SHADE	4.1(4)	14(3)	25(4)	36(3)	48(8)	72(10)	97(13)	15/15
f2	83	87	88	89	90	92	94	15/15
SHADE-L	8.8(1)	11(1)	12(1)	13(1)	16(1)	19(1)	23(2)	15/15
SHADE-L	**3.7**(0.5)$^{\star 4}$	**4.4**(0.8)$^{\star 4}$	**5.0**(0.8)$^{\star 4}$	**5.6**(0.7)$^{\star 4}$	**6.5**(0.9)$^{\star 4}$	**8.0**(0.7)$^{\star 4}$	**9.3**(0.7)$^{\star 4}$	15/15
R-SHADE	7.5(2)	8.7(1.0)	10(2)	12(2)	14(1)	17(3)	21(2)	15/15
f3	716	1622	1637	1642	1646	1650	1654	15/15
SHADE-L	1.6(0.9)	4.2(0.8)	5.0(1)	5.3(0.6)	5.3(1.0)	5.6(1)	5.8(0.8)	15/15
SHADE-L	1.1(0.4)	**1.7**(0.3)	**2.3**(2)	**2.4**(0.5)	**2.4**(1)	**2.6**(2)	**2.7**(2)	15/15
R-SHADE	**1.1**(0.5)	1.7(0.9)	4.5(5)	4.7(4)	4.8(6)	5.1(5)	5.3(4)	15/15
f4	809	1633	1688	1758	1817	1886	1903	15/15
SHADE-L	3.1(1)	5.0(0.9)	7.1(5)	7.4(4)	7.5(4)	7.7(4)	8.1(2)	15/15
SHADE-L	**1.3**(0.3)	**2.2**(2)	**5.0**(6)	**5.0**(6)	**5.1**(6)	**5.2**(2)	**5.4**(5)	15/15
R-SHADE	1.4(0.5)	4.5(6)	8.7(6)	8.5(16)	8.3(5)	8.3(5)	8.4(4)	15/15
f5	10	10	10	10	10	10	10	15/15
SHADE-L	**0.70**(0)	**0.70**(0)	**0.70**(0)	**0.70**(0)	**0.70**(0)	**0.70**(0)	**0.70**(0)	15/15
SHADE-L	**0.70**(0)	**0.70**(0)	**0.70**(0)	**0.70**(0)	**0.70**(0)	**0.70**(0)	**0.70**(0)	15/15
R-SHADE	21(9)	43(8)	62(12)	84(14)	106(20)	151(26)	198(28)	15/15
f6	114	214	281	404	580	1038	1332	15/15
SHADE-L	3.3(1)	4.2(0.8)	5.1(0.7)	4.8(0.6)	4.4(0.3)	3.7(0.3)	3.9(0.3)	15/15
SHADE-L	**2.0**(1)	**2.3**(0.5)	**2.7**(0.7)	2.6(0.5)	**2.3**(0.4)	1.8(0.2)	1.9(0.2)	15/15
R-SHADE	2.1(0.8)	2.5(0.9)	2.8(0.6)	**2.6**(0.4)	2.3(0.4)	**1.8**(0.2)	**1.9**(0.2)	15/15
f7	24	324	1171	1451	1572	1572	1597	15/15
SHADE-L	**3.6**(2)	1.1(0.5)	0.58(0.2)	0.82(0.1)	0.80(0.2)	0.80(0.2)	0.89(0.2)	15/15
SHADE-L	3.6(1)	**0.82**(0.3)	**0.49**(0.4)	**0.60**(0.2)	**0.70**(0.5)	**0.70**(0.5)	**0.72**(0.5)	15/15
R-SHADE	4.3(2)	1.3(1)	0.72(0.9)	0.75(0.5)	0.73(0.3)	0.73(0.7)	0.79(0.7)	15/15
f8	73	273	336	372	391	410	422	15/15
SHADE-L	7.3(2)	6.5(2)	7.4(0.7)	7.7(0.8)	7.8(0.8)	8.1(0.5)	8.6(0.9)	15/15
SHADE-L	**3.7**(1)	4.8(1)	5.8(1)	6.2(3)	6.4(5)	7.0(5)	7.6(3)	15/15
R-SHADE	4.9(2)	**4.7**(5)	**5.3**(2)	**5.8**(4)	**6.1**(3)	**6.7**(3)	**7.5**(3)	15/15
f9	35	127	214	263	300	335	369	15/15
SHADE-L	15(4)	14(4)	11(3)	11(2)	10(2)	10(1)	10(1)	15/15
SHADE-L	8.1(1)	**7.6**(3)	**7.4**(3)	**7.4**(2)	**7.4**(2)	**7.8**(2)	**8.1**(2)	15/15
R-SHADE	**8.0**(2)	11(14)	9.4(6)	9.0(4)	8.7(4)	9.0(4)	9.3(3)	15/15
f10	349	500	574	607	626	829	880	15/15
SHADE-L	2.4(0.8)	**2.1**(0.5)	**2.2**(0.4)	**2.5**(0.2)	**2.6**(0.3)	**2.5**(0.2)	**2.8**(0.2)	15/15
SHADE-L	2.4(1)	2.5(1)	2.8(2)	3.4(2)	3.7(3)	3.4(2)	3.8(2)	15/15
R-SHADE	**2.2**(0.9)	2.3(0.7)	2.4(0.5)	2.6(1)	2.9(0.6)	2.9(0.9)	3.2(0.7)	15/15
f11	143	202	763	977	1177	1467	1673	15/15
SHADE-L	4.4(2)	4.9(1.0)	1.4(0.3)	**1.2**(0.2)	**1.2**(0.1)	**1.3**(0.2)	**1.3**(0.2)	15/15
SHADE-L	3.2(2)	3.7(1)	**1.3**(0.4)	1.3(0.4)	1.3(0.2)	1.4(0.4)	1.6(0.4)	15/15
R-SHADE	**3.1**(0.7)	**3.5**(1)	1.3(1.0)	1.3(0.8)	1.2(0.6)	1.3(0.5)	1.4(0.5)	15/15
f12	108	268	371	413	461	1303	1494	15/15
SHADE-L	12(4)	10(11)	11(10)	12(12)	13(12)	5.8(4)	6.1(5)	15/15
SHADE-L	**10**(10)	9.5(16)	12(18)	14(6)	15(12)	6.8(5)	7.1(5)	15/15
R-SHADE	10(3)	**7.1**(8)	**8.0**(6)	**8.9**(6)	**10**(4)	**5.0**(2)	**5.3**(3)	15/15

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
SHADE-L	4.4(1.0)	5.3(0.8)	6.3(0.7)	6.4(0.5)	2.0(0.1)	2.1(0.1)	2.5(0.2)	15/15
SHADE-L	**3.5**(1)	4.5(1)	5.1(1)	5.4(0.9)	1.7(0.2)	1.7(0.4)	1.7(0.2)	15/15
R-SHADE	3.6(1)	**4.2**(0.7)	**4.7**(1)	**4.9**(0.9)	**1.5**(0.2)	**1.5**(0.2)	**1.5**(0.1)	15/15
f14	10	41	58	90	139	251	476	15/15
SHADE-L	1.7(1)	5.0(3)	7.2(3)	8.4(2)	7.0(0.8)	6.8(0.5)	5.4(0.6)	15/15
SHADE-L	2.0(1)	**3.1**(1.0)	**3.9**(1)$^{\star 2}$	**4.2**(1)$^{\star 2}$	**4.4**(0.9)	**4.8**(0.9)	**3.8**(0.9)	15/15
R-SHADE	**1.3**(1)	3.4(2)	5.2(1)	5.7(1)	5.4(2)	5.3(1)	4.0(0.7)	15/15
f15	511	9310	19369	19743	20073	20769	21359	14/15
SHADE-L	4.3(3)	**1.1**(0.4)	**1.00**(0.6)	**1.1**(0.5)	**1.1**(0.7)	**1.1**(0.4)	**1.1**(0.6)	15/15
SHADE-L	2.1(2)	1.3(1)	1.3(1)	1.3(2)	1.3(2)	1.3(2)	1.3(1)	15/15
R-SHADE	**1.9**(1)	2.3(4)	6.3(7)	6.2(3)	8.3(7)	9.0(8)	8.7(7)	14/15
f16	120	612	2662	10163	10449	11644	12095	15/15
SHADE-L	2.1(3)	10(4)	11(12)	4.8(4)	4.8(3)	4.4(3)	4.3(3)	15/15
SHADE-L	2.5(3)	5.3(2)	3.2(2)	2.2(3)	4.0(3)	4.1(4)	4.0(4)	15/15
R-SHADE	**1.1**(1)	**2.7**(0.6)$^{\star 3}$	**2.5**(3)	**1.4**(2)	**2.2**(3)	**2.4**(2)	**2.3**(2)	15/15
f17	5.0	215	899	2861	3669	6351	7934	15/15
SHADE-L	3.3(3)	2.7(0.9)	**1.7**(0.3)	0.94(0.1)	1.1(0.1)	**1.0**(0.1)	**1.1**(0.1)	15/15
SHADE-L	**2.4**(3)	**1.4**(0.6)	3.7(11)	1.4(3)	1.3(3)	1.2(0.3)	1.3(0.6)	15/15
R-SHADE	2.9(2)	1.5(1)	1.9(0.2)	**0.81**(1)	**0.94**(2)	1.1(1)	1.1(0.7)	15/15
f18	103	378	3968	8451	9280	10905	12469	15/15
SHADE-L	2.5(2)	2.5(0.7)	0.58(0.1)	**0.46**(0.1)	**0.57**(0.0)	**0.75**(0.1)	**0.84**(0.0)	15/15
SHADE-L	**1.0**(0.4)	1.7(1)	1.7(3)	0.98(3)	1.2(2)	4.6(5)	4.3(8)	15/15
R-SHADE	1.4(0.7)	**1.6**(0.5)	**0.32**(0.1)	0.64(2)	1.2(0.9)	2.6(4)	4.7(3)	15/15
f19	1	1	242	1.0e5	1.2e5	1.2e5	1.2e5	15/15
SHADE-L	**17**(13)	1626(1923)	150(90)	12(16)	20(25)	24(45)	25(16)	12/15
SHADE-L	21(23)	**1423**(1012)	173(202)	**5.0**(4)	**7.1**(5)	**13**(21)	**19**(33)	12/15
R-SHADE	22(20)	1787(2812)	**111**(144)	21(26)	∞	∞	∞ *5e5*	0/15
f20	16	851	38111	51362	54470	54861	55313	14/15
SHADE-L	3.5(2)	4.9(2)	**0.30**(0.2)	0.26(0.1)	0.27(0.2)	0.27(0.1)	0.27(0.2)	15/15
SHADE-L	**3.3**(3)	2.2(3)	0.40(0.4)	0.33(0.2)	0.32(0.3)	0.32(0.2)	0.32(0.2)	15/15
R-SHADE	3.9(2)	**1.9**(2)	0.33(0.4)	**0.26**(0.3)	**0.25**(0.3)	**0.25**(0.3)	**0.26**(0.1)	15/15
f21	41	1157	1674	1692	1705	1729	1757	14/15
SHADE-L	2.1(2)	**1.2**(1)	**1.3**(2)	**1.5**(2)	**1.6**(2)	**1.7**(2)	**1.9**(0.8)	15/15
SHADE-L	**1.9**(2)	3.2(4)	2.9(3)	3.0(3)	3.0(3)	3.1(3)	3.2(2)	15/15
R-SHADE	2.0(2)	2.6(2)	3.3(3)	3.3(2)	3.4(2)	3.4(2)	3.5(3)	15/15
f22	71	386	938	980	1008	1040	1068	14/15
SHADE-L	2.7(2)	5.1(6)	5.9(5)	6.1(9)	6.0(9)	6.1(8)	6.4(2)	15/15
SHADE-L	1.6(2)	3.9(7)	**3.3**(4)	**3.3**(2)	**3.3**(5)	**3.5**(5)	**3.6**(3)	15/15
R-SHADE	**1.5**(1)	**3.2**(6)	5.5(7)	5.5(7)	5.5(7)	5.5(7)	5.6(3)	15/15
f23	3.0	518	14249	27890	31654	33030	34256	15/15
SHADE-L	**2.5**(3)	13(15)	2.5(3)	1.3(1)	1.2(0.8)	1.2(0.8)	1.2(1)	15/15
SHADE-L	3.2(3)	10(9)	2.4(2)	1.4(2)	1.3(2)	1.3(1)	1.3(1)	15/15
R-SHADE	2.9(2)	**6.2**(6)	**1.7**(1)	**0.93**(0.8)	**0.83**(1)	**0.82**(0.4)	**0.82**(0.7)	15/15
f24	1622	2.2e5	6.4e6	9.6e6	9.6e6	1.3e7	1.3e7	3/15
SHADE-L	4.1(1)	2.4(2)	5.4(4)	**3.6**(3)	**3.6**(7)	**2.7**(5)	**2.7**(3)	2/15
SHADE-L	1.7(0.7)	**1.8**(3)	**5.3**(4)	∞	∞	∞	∞ *5e6*	0/15
R-SHADE	**1.7**(1)	2.6(5)	∞	∞	∞	∞	∞ *5e5*	0/15

Table 2: Expected runtime (ERT in number of function evaluations) divided by the respective best ERT measured during BBOB-2009 in dimension 5. This ERT ratio 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 ERT 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 $f_{\text{opt}} + 10^{-8}$. 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 functions (24). A ↓ indicates the same tested against the best algorithm from BBOB 2009. Best results are printed in bold.

Data produced with COCO v2.4

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
SHADE-L	**0.98**(0)	**0.98**(0)	**0.98**(0)	**0.98**(0)	**0.98**(0)	**0.98**(0)	**0.98**(0)	15/15
SHADE-L	**0.98**(0)	**0.98**(0)	**0.98**(0)	**0.98**(0)	**0.98**(0)	**0.98**(0)	**0.98**(0)	15/15
R-SHADE	27(7)	59(9)	89(10)	119(8)	149(14)	208(23)	264(26)	15/15
f2	385	386	387	388	390	391	393	15/15
SHADE-L	12(1)	14(0.6)	15(1)	18(0.6)	19(0.6)	24(0.6)	28(0.7)	15/15
SHADE-L	**5.3**(0.3)$^{\star 4}$	**6.2**(0.4)$^{\star 4}$	**7.2**(0.3)$^{\star 4}$	**8.1**(0.6)$^{\star 4}$	**9.0**(0.4)$^{\star 4}$	**11**(0.5)$^{\star 4}$	**13**(0.7)$^{\star 4}$	15/15
R-SHADE	18(2)	21(2)	25(3)	28(2)	31(3)	37(3)	43(3)	15/15
f3	5066	7626	7635	7637	7643	7646	7651	15/15
SHADE-L	22(2)	21(1)	22(2)	22(1)	23(2)	23(1)	23(1)	15/15
SHADE-L	7.9(1)	7.5(0.4)	8.0(0.5)	8.0(0.5)	**8.1**(0.5)	**8.2**(0.6)	**8.3**(0.6)	15/15
R-SHADE	**7.7**(0.4)	**7.1**(0.2)	**7.7**(0.2)	**7.9**(0.2)	8.1(0.3)	8.4(0.3)	8.7(0.2)	15/15
f4	4722	7628	7666	7686	7700	7758	1.4e5	9/15
SHADE-L	27(3)	24(2)	26(1)	26(1)	26(2)	27(2)	1.5(0.1)	15/15
SHADE-L	10(0.8)	8.4(0.6)	**11**(4)	**11**(4)	**11**(7)	**11**(4)	**0.64**(0.4)	15/15
R-SHADE	**9.4**(0.8)	**8.3**(0.5)	12(0.5)	12(5)	12(8)	13(3)	0.72(0.4)	15/15
f5	41	41	41	41	41	41	41	15/15
SHADE-L	**0.54**(0)	**0.54**(0)	**0.54**(0)	**0.54**(0)	**0.54**(0)	**0.54**(0)	**0.54**(0)	15/15
SHADE-L	**0.54**(0)	**0.54**(0)	**0.54**(0)	**0.54**(0)	**0.54**(0)	**0.54**(0)	**0.54**(0)	15/15
R-SHADE	117(14)	212(21)	308(27)	400(28)	489(31)	679(24)	862(33)	15/15
f6	1296	2343	3413	4255	5220	6728	8409	15/15
SHADE-L	7.4(1)	7.3(0.3)	6.8(0.5)	6.9(0.7)	6.8(0.5)	7.1(0.6)	7.2(0.3)	15/15
SHADE-L	**4.0**(0.5)	**3.7**(0.4)	**3.5**(0.3)	**3.6**(0.4)	**3.6**(0.3)	**3.8**(0.3)	**3.9**(0.3)	15/15
R-SHADE	4.2(0.4)	4.0(0.5)	3.8(0.4)	3.9(0.3)	3.9(0.4)	4.1(0.5)	4.1(0.2)	15/15
f7	1351	4274	9503	16523	16524	16524	16969	15/15
SHADE-L	3.4(0.8)	**4.0**(5)	**4.3**(6)	**2.9**(4)	**2.9**(2)	**2.9**(3)	**2.9**(3)	15/15
SHADE-L	**2.0**(0.4)	7.5(5)	5.4(4)	4.6(3)	4.7(2)	4.7(2)	4.6(2)	15/15
R-SHADE	2.0(0.7)	29(41)	72(103)	1762(1120)	1761(1604)	1761(3268)	1715(1650)	1/15
f8	2039	3871	4040	4148	4219	4371	4484	15/15
SHADE-L	14(0.9)	14(0.5)	15(0.6)	16(0.6)	16(0.5)	17(0.5)	18(0.5)	15/15
SHADE-L	**7.7**(2)$^{\star 2}$	**8.0**(1)$^{\star 4}$	**8.7**(1)$^{\star 4}$	**9.1**(0.9)$^{\star 4}$	**9.5**(0.9)$^{\star 4}$	**10**(1)$^{\star 4}$	**11**(0.9)$^{\star 4}$	15/15
R-SHADE	11(2)	14(8)	15(5)	15(6)	15(8)	16(5)	16(7)	15/15
f9	1716	3102	3277	3379	3455	3594	3727	15/15
SHADE-L	20(1)	21(0.9)	23(0.9)	24(0.5)	25(0.9)	26(0.6)	27(0.9)	15/15
SHADE-L	**13**(1)*	**15**(0.7)$^{\star 3}$	**16**(1)$^{\star 3}$	**17**(1)$^{\star 3}$	**18**(1)$^{\star 3}$	**18**(1)$^{\star 3}$	**19**(0.9)$^{\star 3}$	15/15
R-SHADE	16(2)	20(8)	22(8)	23(3)	23(3)	24(3)	24(7)	15/15
f10	7413	8661	10735	13641	14920	17073	17476	15/15
SHADE-L	**3.4**(0.6)$^{\star 4}$	**3.6**(0.5)$^{\star 4}$	**3.3**(0.3)$^{\star 4}$	**3.0**(0.4)$^{\star 4}$	**3.0**(0.4)$^{\star 4}$	**3.1**(0.3)$^{\star 4}$	**3.5**(0.4)$^{\star 4}$	15/15
SHADE-L	20(9)	29(10)	30(8)	30(6)	35(8)	41(13)	51(9)	15/15
R-SHADE	16(5)	22(5)	25(5)	26(8)	29(6)	32(8)	40(10)	9/15
f11	1002	2228	6278	8586	9762	12285	14831	15/15
SHADE-L	10(2)	**6.6**(1.0)$^{\star 2}$	**3.0**(0.3)$^{\star 4}$	**2.7**(0.4)$^{\star 4}$	**2.8**(0.3)$^{\star 4}$	**2.9**(0.3)$^{\star 4}$	**2.9**(0.4)$^{\star 4}$	15/15
SHADE-L	11(6)	16(9)	9.2(3)	9.4(3)	10(3)	12(2)	13(2)	15/15
R-SHADE	**7.6**(5)	13(7)	8.2(3)	8.4(2)	10(3)	11(3)	12(2)	15/15
f12	1042	1938	2740	3156	4140	12407	13827	15/15
SHADE-L	13(15)	17(16)	22(16)	25(16)	23(14)	11(4)	11(2)	15/15
SHADE-L	**6.1**(7)$^{\star 2}$	**14**(14)	**18**(19)	**21**(15)	**21**(14)	**10**(4)	**11**(4)	15/15
R-SHADE	8.5(1.0)	18(15)	23(15)	26(18)	25(14)	12(5)	13(5)	15/15

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
SHADE-L	11(3)	11(6)	14(8)	15(4)	3.4(0.9)	**3.3**(0.6)$^{\star 2}$	**3.3**(0.5)$^{\star 4}$	15/15
SHADE-L	**8.7**(5)	**8.0**(4)	**9.1**(5)	**11**(5)	**3.0**(1)	6.2(2)	12(6)	15/15
R-SHADE	12(7)	8.0(4)	10(3)	12(3)	3.4(0.8)	5.2(2)	25(43)	0/15
f14	75	239	304	451	932	1648	15661	15/15
SHADE-L	14(5)	10(1)	10(1)	18(2)	14(1)	**14**(2)$^{\star 4}$	**2.4**(0.4)$^{\star 4}$	15/15
SHADE-L	8.5(2)	**5.1**(0.7)$^{\star 4}$	**6.4**(1)$^{\star 4}$	**8.5**(0.6)$^{\star 4}$	**7.3**(1)$^{\star 3}$	48(23)	33(17)	15/15
R-SHADE	**8.2**(2)	9.4(2)	13(1)	13(2)	11(2)	58(32)	1861(958)	0/15
f15	30378	1.5e5	3.1e5	3.2e5	3.2e5	4.5e5	4.6e5	15/15
SHADE-L	14(10)	∞	∞	∞	∞	∞	∞ *2e7*	0/15
SHADE-L	**13**(8)	**1933**(2866)	∞	∞	∞	∞	∞ *2e7*	0/15
R-SHADE	52(68)	∞	∞	∞	∞	∞	∞ *2e6*	0/15
f16	1384	27265	77015	1.4e5	1.9e5	2.0e5	2.2e5	15/15
SHADE-L	52(16)	1116(1836)	**3685**(4869)	∞	∞	∞	∞ *2e7*	0/15
SHADE-L	**24**(13)	4786(6609)	∞	∞	∞	∞	∞ *2e7*	0/15
R-SHADE	27(20)	**246**(391)	∞	∞	∞	∞	∞ *2e6*	0/15
f17	63	1030	4005	12242	30677	56288	80472	15/15
SHADE-L	7.7(3)	3.7(0.8)	**2.6**(0.8)	**1.6**(0.6)	**3.8**(11)	**2.9**(1)$^{\star 2}$	**2.7**(4)$^{\star 2}$	15/15
SHADE-L	3.8(2)	**3.2**(1)	6.5(17)	5.8(11)	8.2(6)	45(38)	34(9)	14/15
R-SHADE	**3.7**(2)	3.4(1)	3.9(7)	18(30)	45(45)	∞	∞ *2e6*	0/15
f18	621	3972	19561	28555	67569	1.3e5	1.5e5	15/15
SHADE-L	3.7(1)	2.8(0.7)	**1.1**(0.2)	**1.3**(0.3)$^{\star 2}$	**86**(180)	**79**(91)	**128**(169)	9/15
SHADE-L	**2.7**(0.8)	**1.8**(0.6)	72(247)	192(391)	549(1765)	2168(1416)	∞ *2e7*	0/15
R-SHADE	3.1(0.8)	2.0(0.8)	30(41)	1050(630)	∞	∞	∞ *2e6*	0/15
f19	1	1	3.4e5	4.7e6	6.2e6	6.7e6	6.7e6	15/15
SHADE-L	280(127)	1.1e7(8e6)	**381**(539)	∞	∞	∞	∞ *2e7*	0/15
SHADE-L	**186**(68)*	**1.5e5**(6e4)$^{\star 3}$	820(918)	∞	∞	∞	∞ *2e7*	0/15
R-SHADE	344(73)	1.1e6(9e5)	∞	∞	∞	∞	∞ *2e6*	0/15
f20	82	46150	3.1e6	5.5e6	5.5e6	5.6e6	5.6e6	14/15
SHADE-L	**5.4**(2)	3.0(0.6)	**19**(27)	**51**(79)	**51**(44)	**50**(38)	**50**(35)	1/15
SHADE-L	5.6(3)	8.3(11)	∞	∞	∞	∞	∞ *2e7*	0/15
R-SHADE	10(3)	**1.3**(0.3)	∞	∞	∞	∞	∞ *2e6*	0/15
f21	561	6541	14103	14318	14643	15567	17589	15/15
SHADE-L	8.3(16)	55(180)	63(51)	63(174)	61(170)	58(160)	51(141)	15/15
SHADE-L	3.5(4)	116(394)	62(180)	61(44)	60(203)	56(163)	50(169)	15/15
R-SHADE	**3.0**(1)	**6.6**(10)	**6.5**(13)	**6.5**(6)	**6.4**(10)	**6.1**(11)	**5.5**(10)	15/15
f22	467	5580	23491	24163	24948	26847	1.3e5	12/15
SHADE-L	7.3(10)	1401(1846)	1.2e4(9791)	1.2e4(3e4)	1.1e4(1e4)	1.1e4(1e4)	2098(2856)	1/15
SHADE-L	**7.0**(17)	1853(2770)	5788(7156)	5627(7243)	5450(6213)	5065(7263)	1009(1347)	2/15
R-SHADE	14(41)	**29**(25)	**1200**(681)	**1167**(1304)	**1130**(1343)	**1051**(1415)	**209**(249)	1/15
f23	3.0	1614	67457	3.7e5	4.9e5	8.1e5	8.4e5	15/15
SHADE-L	2.6(2)	1005(2372)	350(245)	∞	∞	∞	∞ *2e7*	0/15
SHADE-L	**1.5**(2)	**80**(79)	917(1719)	∞	∞	∞	∞ *2e7*	0/15
R-SHADE	2.3(2)	95(80)	**12**(14)$^{\star 2}$	∞	∞	∞	∞ *2e6*	0/15
f24	1.3e6	7.5e6	5.2e7	5.2e7	5.2e7	5.2e7	5.2e7	3/15
SHADE-L	**14**(15)	∞	∞	∞	∞	∞	∞ *2e7*	0/15
SHADE-L	64(52)	∞	∞	∞	∞	∞	∞ *2e7*	0/15
R-SHADE	∞	∞	∞	∞	∞	∞	∞ *2e6*	0/15

Table 3: Expected runtime (ERT in number of function evaluations) divided by the respective best ERT measured during BBOB-2009 in dimension 20. This ERT ratio 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 ERT 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 by the number of functions (24). A ↓ indicates the same tested against the best algorithm from BBOB 2009. Best results are printed in bold.

Data produced with COCO v2.4