MEMPSODE: Comparing Particle Swarm Optimization and Differential Evolution Within a Hybrid Memetic Global Optimization Framework

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

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

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

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

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

ABSTRACT

MEMPSODE is a recently published optimization software that implements memetic Particle Swarm Optimization and Differential Evolution approaches. It combines previously proposed variants of the two algorithms, with the Merlin optimization environment, which includes a variety of established local search methods for continuous optimization. The present study aims at comparing the performance of the memetic variants produced by the two metaheuristics within the framework of MEMPSODE. The algorithms are assessed on the noiseless testbed of the Black–Box Optimization Benchmarking 2012 workshop, providing useful insight regarding their relative efficiency and effectiveness.

Categories and Subject Descriptors

G.1.6 [**Numerical Analysis**]: Optimization—*global optimization, memetic algorithms, particle swarm optimization, differential evolution, local search*; G.4 [**Mathematical Software**]

Keywords

Memetic global optimization, Particle swarm optimization, Differential evolution, Black-box optimization, Merlin optimization environment

*Corresponding author.

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

1. INTRODUCTION

Particle Swarm Optimization (PSO) and Differential Evolution (DE) have been established as promising optimization tools for solving continuous global optimization problems [2, 8, 13]. They are essentially based on models that draw inspiration from physical systems, exhibiting remarkable capability of locating global solutions in various optimization tasks. However, in many cases, the exploration capability of these algorithms is not accompanied by proportional exploitation properties, i.e., their solution fine–tuning capability is inferior than their global search quality.

This weakness motivated the development of a closely related category of algorithms, namely *Memetic Algorithms* (MAs). MAs constitute a class of hybrid metaheuristics that combine population–based optimization algorithms with established local search procedures [9]. Recently, the MEMPSODE (MEMetic Particle Swarm Optimization and Differential Evolution) software was published [17]. MEMPSODE implements memetic schemes of a generalized PSO variant, namely Unified PSO (UPSO) [12], as well as DE. It combines the memetic schemes proposed in [14] originally for PSO, with the algorithmic local search artillery of the established Merlin optimization environment [10].

The present paper aims at comparing the performance of memetic variants implemented within the framework of MEMPSODE, on the noiseless testbed of the Black–Box Optimization Benchmarking 2012 (BBOB'12) workshop. The primary target is to determine the impact of the choice between UPSO and DE, within the memetic framework presented in [14]. No extensive benchmarking of all the available MEMPSODE variants was possible. Instead, we used a default set of parameters for both UPSO and DE, as well as the same local search procedure, in order to gain insight regarding the most promising uncalibrated approaches offered by MEMPSODE.

The rest of the paper is organized as follows: Section 2 offers brief descriptions of all the employed algorithms, while

Section 3 explains the experimental setting. The obtained experimental results are reported in Section 5, and the paper concludes in Section 6.

2. EMPLOYED ALGORITHMS

As previously mentioned, the MEMPSODE software follows closely the PSO–based memetic schemes proposed in [14], and extends them also to the DE framework. In the following paragraphs, we provide brief descriptions of the algorithms integrated in MEMPSODE, along with a pseudocode that summarizes the corresponding memetic framework.

2.1 Unified Particle Swarm Optimization

The original PSO algorithm was first introduced by Eberhart and Kennedy [1]. UPSO was introduced later as a generalization of PSO that harnesses its local and global variant [12]. If the n–dimensional continuous optimization problem:

$$\min_{x \in X \subset \mathbb{R}^n} f(x), \tag{1}$$

is under investigation, with the search space X being defined as $X \equiv [l_1, r_1] \times [l_2, r_2] \times \cdots \times [l_n, r_n]$, then PSO employs a set, called a *swarm*, of N search agents, called the *particles*, $S = \{x_1, x_2, \ldots, x_N\}$, to probe X. The i–th particle is defined as $x_i = (x_{i1}, x_{i2}, \ldots, x_{in})^\top \in X$, $i \in I = \{1, 2, \ldots, N\}$, and moves in X by assuming an adaptable *velocity* (position shift), $v_i = (v_{i1}, v_{i2}, \ldots, v_{in})^\top$, $i \in I$, while retaining in memory the *best position* it has ever visited, $p_i = (p_{i1}, p_{i2}, \ldots, p_{in})^\top \in X$, $i \in I$.

Also, each particle assumes a *neighborhood* of particles that share information with it. The neighborhood, $\mathcal{N}_i$, is usually defined as a set of indices of the communicating particles, while the shared information is the best position they have ever detected. Henceforth, $g_i \in \mathcal{N}_i$ will denote the index of the particle with the best detected position in the neighborhood of x_i.

If $\mathcal{N}_i \equiv I$, then the whole swarm is considered as the neighborhood of each particle, defining the *global* (also denoted as *gbest*) PSO variant, while $\mathcal{N}_i \subset I$ defines *local* (also denoted as *lbest*) variants. Obviously, in the gbest model it holds that $g_i = g, \forall i \in I$, where g is the index of the particle with the overall best position ever detected.

The classical PSO model can be generalized in the UPSO scheme [12]. If we assume that $G_i^{(t+1)}$ and $L_i^{(t+1)}$ denote the velocity update of x_i in the gbest and lbest PSO model, respectively, and t denotes the iteration counter, it holds that [12]:

$$G_{ij}^{(t+1)} = \chi \left[v_{ij}^{(t)} + c_1 \mathcal{R}_1 \left(p_{ij}^{(t)} - x_{ij}^{(t)} \right) + c_2 \mathcal{R}_2 \left(p_{gj}^{(t)} - x_{ij}^{(t)} \right) \right],$$

$$L_{ij}^{(t+1)} = \chi \left[v_{ij}^{(t)} + c_1 \mathcal{R}_1 \left(p_{ij}^{(t)} - x_{ij}^{(t)} \right) + c_2 \mathcal{R}_2 \left(p_{gij}^{(t)} - x_{ij}^{(t)} \right) \right],$$

where $i \in I$; $j = 1, 2, \ldots, n$; g is the index of the overall best particle; and $\mathcal{R}_1$, $\mathcal{R}_2$, are random variables uniformly distributed in $[0, 1]$. Then, the particles are updated as follows [11]:

$$v_{ij}^{(t+1)} = u\, G_{ij}^{(t+1)} + (1 - u)\, L_{ij}^{(t+1)}, \tag{2}$$

$$x_{ij}^{(t+1)} = x_{ij}^{(t)} + v_{ij}^{(t+1)}. \tag{3}$$

The parameter $u \in [0, 1]$ is called the *unification factor* and balances the influence (trade–off) of the gbest and lbest velocity update. Obviously, the lbest PSO model is retrieved for $u = 0$, while for $u = 1$ the gbest PSO model is obtained. All intermediate values produce combinations with diverse convergence properties. The presented UPSO variant is implemented in the MEMPSODE software [17] that was employed in our study.

2.2 Differential Evolution

DE was introduced by Storn and Price [15] as a population–based stochastic optimization algorithm for numerical optimization problems. DE is formulated similarly to PSO. A population, $P = \{x_1, x_2, \ldots, x_N\}$, of N individuals is utilized to probe the search space, $X \subset \mathbb{R}^n$. The population is randomly initialized, usually following a uniform distribution within the search space.

Each individual is an n–dimensional vector, $x_i = (x_{i1}, x_{i2}, \ldots, x_{in})^\top \in X$, $i \in I = \{1, 2, \ldots, N\}$, serving as a candidate solution of the problem at hand. The population is iteratively evolved by applying two operators, *mutation* and *recombination*, on each individual to produce new candidate solutions. Then, a selection takes place, using both new and old individuals, to construct the new population consisting of the N best individuals. The procedure continues in the same manner until a termination criterion is satisfied.

The mutation operator produces a new vector, v_i, for each individual, x_i, $i \in I$, by combining some of the rest of the individuals. There are various DE mutation operators. The most common one is defined as follows:

$$v_i^{(t+1)} = x_g^{(t)} + F \left(x_{r_1}^{(t)} - x_{r_2}^{(t)} \right), \tag{4}$$

where t denotes the iteration counter; $F \in (0, 1]$ is a fixed user–defined parameter; g denotes the index of the best individual in the population, i.e., the one with the lowest function value; and $r_1, r_2 \in I$, are mutually different randomly selected indices that differ also from the index i.

After the mutation, a recombination operator is applied producing a trial vector:

$$u_i = (u_{i1}, u_{i2}, \ldots, u_{in}), \qquad i \in I,$$

for each individual. This vector is defined as follows:

$$u_{ij}^{(t+1)} = \begin{cases} v_{ij}^{(t+1)}, & \text{if } R_j \leqslant CR \text{ or } j = \mathrm{RI}(i), \\ x_{ij}^{(t)}, & \text{if } R_j > CR \text{ and } j \neq \mathrm{RI}(i), \end{cases} \tag{5}$$

where $j = 1, 2, \ldots, n$; R_j is a random variable uniformly distributed in the range $[0, 1]$; $CR \in [0, 1]$ is a user–defined crossover constant; and $\mathrm{RI}(i) \in \{1, 2, \ldots, n\}$, is a randomly selected index.

Finally, the trial vector is compared against the corresponding individual and the best one is selected to be inherited to the next generation, i.e.:

$$x_i^{(t+1)} = \begin{cases} u_i^{(t+1)}, & \text{if } f\left(u_i^{(t+1)}\right) < f\left(x_i^{(t)}\right), \\ x_i^{(t)}, & \text{otherwise.} \end{cases} \tag{6}$$

The presented DE operator, along with the rest of the basic DE operators, are implemented in the MEMPSODE software [17].

2.3 Local Search

A local solution to an optimization problem can be obtained by applying local search (LS) methods. The hybrid schemes implemented in MEMPSODE employ deterministic

LS procedures. These procedures require a starting point, $x^{(0)}$, and generate a sequence of points, $x^{(t)}$, $t = 1, 2, \ldots$, which approximates a minimizer within a prescribed accuracy.

The generation of a new point, $x^{(t+1)}$, in the sequence is based on information collected for the current iterate, $x^{(t)}$. Typically, this information includes the function value at $x^{(t)}$, as well as the first– and probably second–order derivatives of $f(x)$ at $x^{(t)}$. In all cases, the aim is to find a new iterate with lower function value than the current one.

The *Merlin* optimization environment [10] is an efficient and robust general purpose optimization package. It is designed to solve multi–dimensional optimization problems. Merlin offers a variety of well established gradient–based and gradient–free optimization algorithms. Gradient–based algorithms include three methods from the conjugate gradient family, the method of Levenberg–Marquardt, the DFP and several variations of the BFGS algorithms (BFGS) [4]. The gradient–free algorithms include a pattern search and the nonlinear Simplex method.

2.4 Memetic Framework

The design of Memetic PSO (MPSO) in [14] was based on three fundamental schemes, called the *memetic strategies*:

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

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

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

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

Of course, many other memetic strategies can be considered. For instance, a simple one would be the application of LS on every particle. However, such an approach would be costly in terms of function evaluations. In practice, only a small number of particles are considered as starting points for LS, as pointed out in [7]. The memetic strategies proposed in [14] are also implemented in MEMPSODE for both PSO– and DE–based MAs. The general procedure of the memetic algorithms discussed in our study, is given with the pseudocode of Algorithm 1. Notice that the pseudocode uses the same vector notation for both UPSO and DE.

Algorithm 1: Pseudocode of the implemented memetic algorithms.

Input: Objective function, $f : X \subset \mathbb{R}^n \to \mathbb{R}$; algorithm: *algo* (PSO/DE); swarm size: N; unification factor: `UF`; probability for local search: ρ
Output: Best detected solution: x^*, $f(x^*)$.

```
// Initialization
for i = 1, 2, ..., N do
    Initialize x_i and u_i
    Set p_i ← x_i // Initialize best position
    f_i ← f(x_i) // Evaluate particle
    f_i^p ← f_i // Best position value
end
// Main Iteration Loop
Set t ← 0
while (termination criterion) do
    Calculate global best index g_1 and local best index g_2
    // Update Swarm/Population
    if algo = 'PSO' then
        for i = 1, 2, ..., N do
            Calculate lbest velocity update, L_i, using g_2
            Calculate gbest velocity update, G_i, using g_1
            u_i ← UF L_i + (1 − UF) G_i // Unified PSO
            x_i = x_i + u_i // Update particle's position
        end
    else if algo = 'DE' then
        for i = 1, 2, ..., N do
            x_i ← p_i // Replicate best positions
        end
        for i = 1, 2, ..., N do
            Calculate v_i using Eq. (4) // Mutation
        end
        for i = 1, 2, ..., N do
            Calculate u_i using Eq. (5) // Recombination
        end
        for i = 1, 2, ..., N do
            Calculate x_i using Eq. (6)
        end
    end
    // Evaluate Swarm/Population
    for i = 1, 2, ..., N do
        f_i ← f(x_i)
    end
    // Update Best Positions/Individuals
    for i = 1, 2, ..., N do
        if f_i < f(p_i) then
            p_i ← x_i
            f_i^p ← f_i
        end
    end
    // Apply Memetic Strategy
    Apply one of the memetic strategies with probability ρ
end
```

3. EXPERIMENTAL SETUP

We applied the memetic algorithms for a maximum of $10^5 \times n$ function evaluations per run. Whenever MEMPSODE terminated before reaching the global minimum (e.g., when LS resulted in a local minimizer), we performed an independent restart until the maximum number of function evaluations was reached.

The memetic Scheme 3 was applied with probability of local search set to $\rho_i = 0.05$. Both UPSO and DE used a swarm size $N = 25$ particles. In UPSO the unification factor u was set to 1 (gbest) and the initial velocity vector was restraint by a factor of 0.01. For the DE experiments we used the values $F = 0.5$ and $CR = 0.7$. Each local search assumed a maximum number of 2000 function evaluations.

The employed LS algorithm was the BFGS method using the default parameters provided by Merlin. For the first order derivatives we applied an $O(h)$ finite differences formula where h was an adaptable step size (see [16] for details). The experiments were conducted on an Intel I7-2600 3.4 GHz machine with 8GB RAM using GNU compiler suite v.4.4.3.

4. CPU TIMING EXPERIMENT

For the timing experiment, according to [5], the experimental procedure described above was run on f8 with at most 1000 function evaluations in each call to MEMPSODE and restarted until at least 30 seconds had passed. The timing experiment was performed on the same platform as the experimental procedure. The results for the UPSO variant were 2.8, 1.8, 1.2, 0.5, 0.26, and 0.24 times 10^{-4} seconds per function evaluation and for the DE variant 3.1, 2.0, 1.2, 0.5,

0.27, and 0.22 times 10^{-4} seconds per function evaluation, for dimensions 2, 3, 5, 10, 20, and 40, respectively.

5. EXPERIMENTAL RESULTS

Results from experiments according to [5] on the benchmark functions given in [3, 6] are presented in Figs. 1, 2 and 3 and in Table 1. The *expected running time* (ERT), reported in figures and tables, depends on a given target function value, $f_t = f_{opt} + \Delta f$, and it 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 [5].

A direct comparison between the memetic gbest variant of UPSO and the DE variant implemented in MEMPSODE, can be deduced by observing the scatter plots in Fig. 2 and the starred records in Table 1. In the separable case, DE variant outperforms gbest PSO, especially as dimensionality increases. For the moderate category, DE seems to outperform the gbest PSO variant only on function 7 (step ellipsoid) and scores marginally better in all other cases. The same behaviour is repeated for the ill-conditioned cases where DE variant is slightly superior. Finally, DE variant outperforms gbest PSO in all multimodal functions but the opposite is observed for the weak structured cases.

The DE variant's superiority against gbest PSO is also obvious by inspecting Fig. 3. In almost all cases (except the weak structured functions) the ECDF of DE variant lies higher than the corresponding ECDF of PSO variant and this pattern is repeated in all levels of accuracy.

It is also worth mentioning that the DE variant scored the best recorded ERT for some accuracy levels in the case of ill–conditioned functions (functions 10–14). From the corresponding lines of Table 1, we can see that for relatively low levels of accuracy the achieved ERT scores are quite competitive. Overall, the results suggest that the algorithms implemented in MEMPSODE can be very competitive.

6. CONCLUSIONS

We presented an empirical evaluation between two implementations of the memetic algorithm introduced in [14]. The original one uses gbest PSO for search space exploration, and the variation applies DE. Both methods are included in the MEMPSODE software that provides a versatile environment for global optimization. The results from the comparison on the noiseless testbed indicate that the DE variant has a relative advantage against the gbest PSO model. Both methods attain competitive performance, scoring way above the average, against the majority of algorithms on the same noiseless testbed, as suggested by the results.

7. REFERENCES

[1] R. C. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Proceedings Sixth Symposium on Micro Machine and Human Science*, pages 39–43, Piscataway, NJ, 1995. IEEE Service Center.

[2] A. P. Engelbrecht. *Fundamentals of Computational Swarm Intelligence*. Wiley, 2006.

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

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

[6] 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.

[7] W. E. Hart. *Adaptive Global Optimization with Local Search*. PhD thesis, University of California, San Diego,USA, 1994.

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

[9] P. Moscato. Memetic algorithms: A short introduction. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 219–235. McGraw–Hill, London, 1999.

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

[11] K. E. Parsopoulos and M. N. Vrahatis. UPSO: A unified particle swarm optimization scheme. In *Lecture Series on Computer and Computational Sciences, Vol. 1, Proceedings of the International Conference of Computational Methods in Sciences and Engineering (ICCMSE 2004)*, pages 868–873. VSP International Science Publishers, Zeist, The Netherlands, 2004.

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

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

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

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

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

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

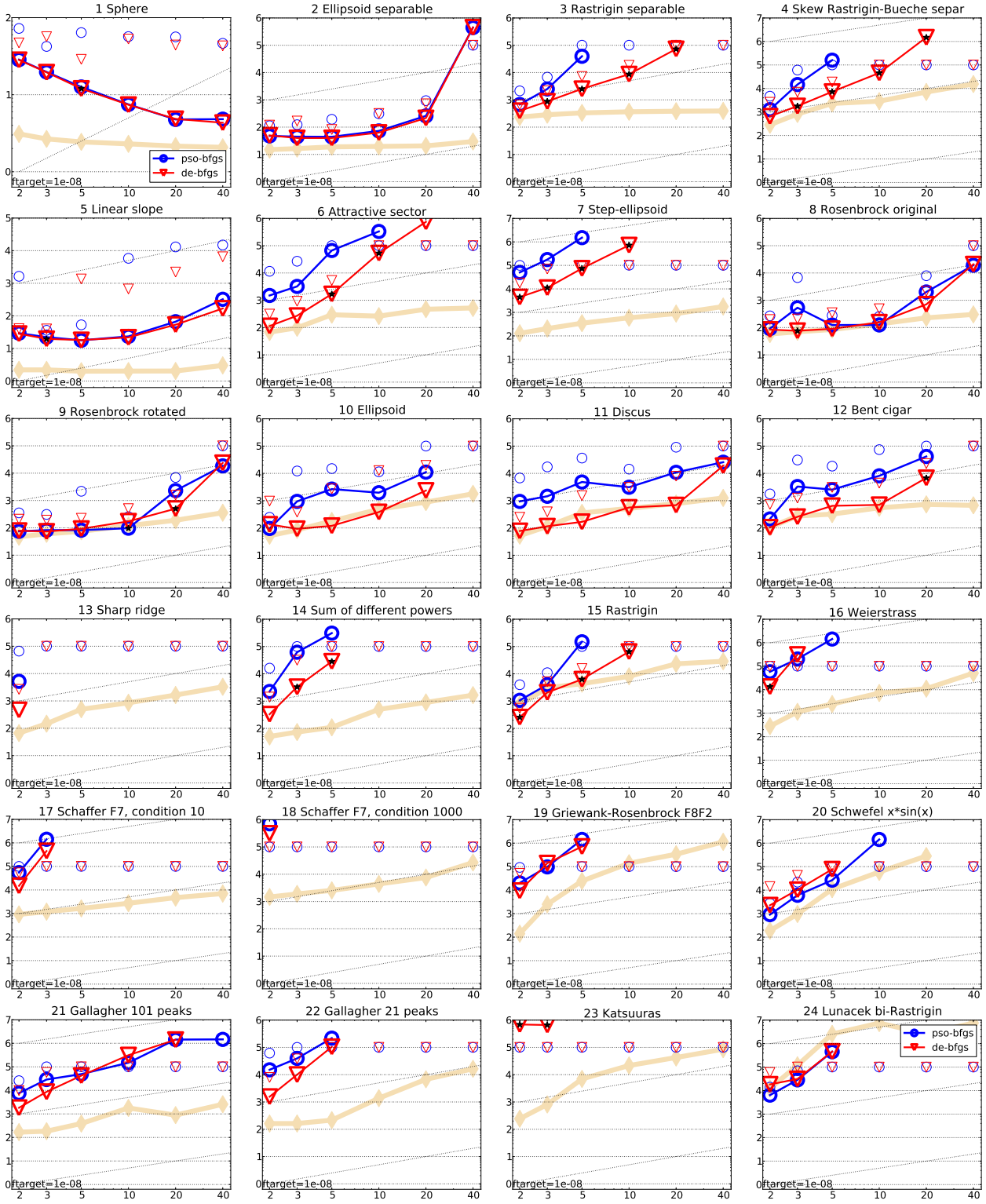

Figure 1: Expected running time (ERT in number of f-evaluations) divided by dimension for target function value 10^{-8} as $\log_{10}$ values versus dimension. Different symbols correspond to different algorithms given in the legend of f_1 and f_{24}. Light symbols give the maximum number of function evaluations from the longest trial divided by dimension. Horizontal lines give linear scaling, slanted dotted lines give quadratic scaling. Black stars indicate statistically better result compared to all other algorithms with $p < 0.01$ and Bonferroni correction number of dimensions (six). Legend: ○: pso-bfgs, ▽: de-bfgs.

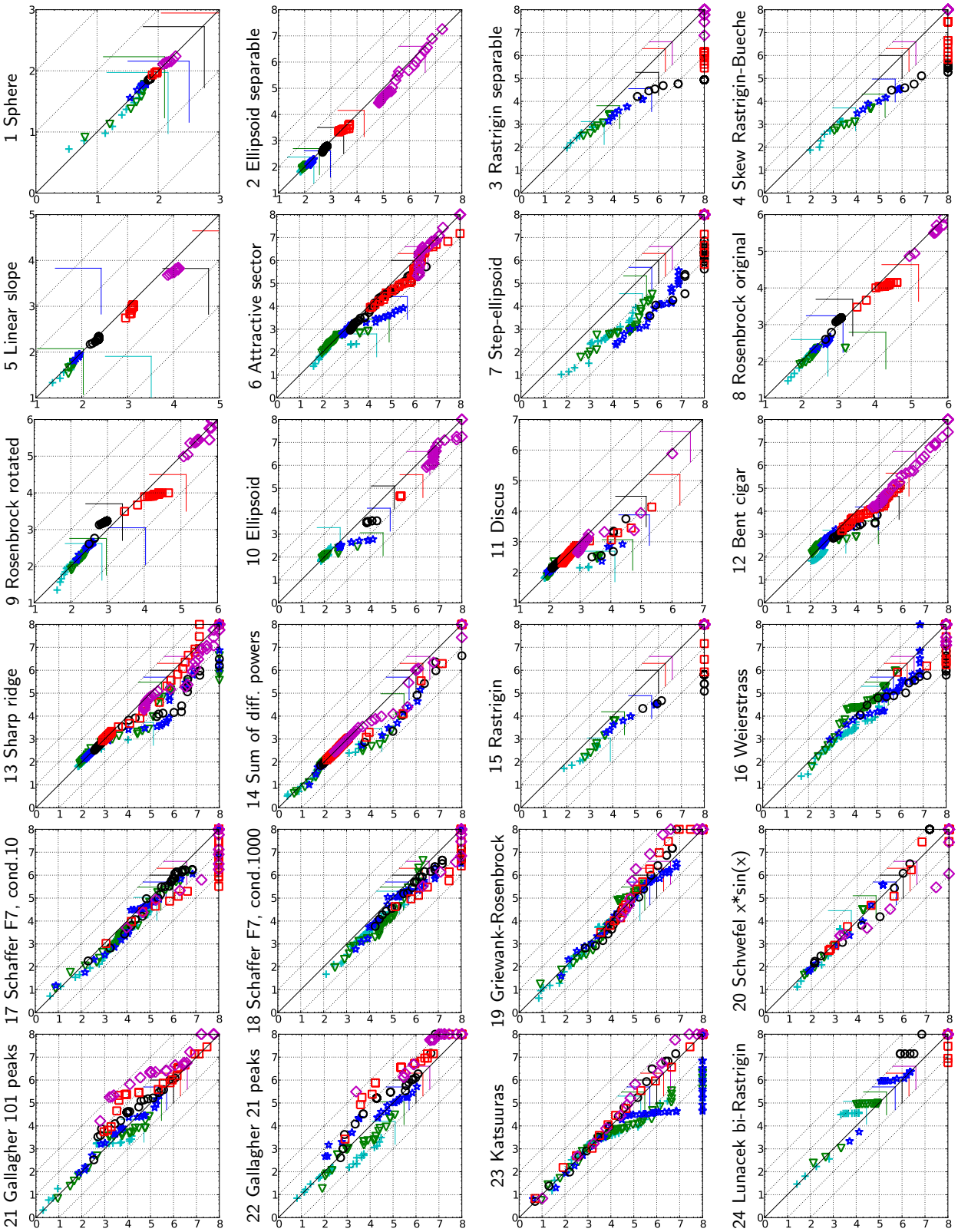

Figure 2: Expected running time (ERT in $\log_{10}$ **of number of function evaluations) of pso-bfgs** (x**-axis) versus de-bfgs** (y**-axis) for 46 target values** $\Delta f \in [10^{-8}, 10]$ **in each dimension on functions** f_1–f_{24}**. Markers on the upper or right edge indicate that the target value was never reached. Markers represent dimension: 2:+, 3:▽, 5:⋆, 10:○, 20:□, 40:◇.**

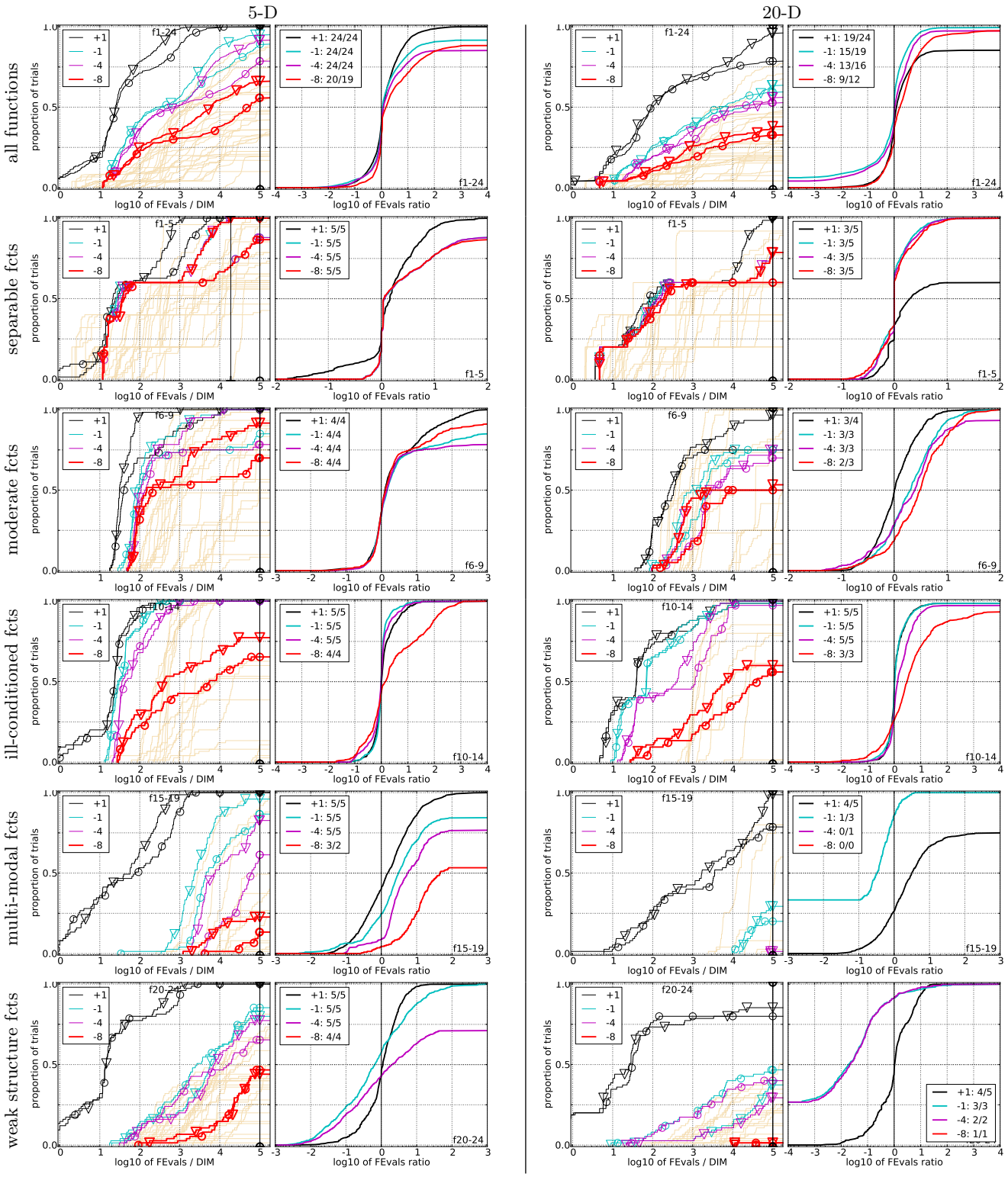

Figure 3: Empirical cumulative distributions (ECDF) of run lengths and speed-up ratios in 5-D (left) and 20-D (right). Left sub-columns: ECDF of the number of function evaluations divided by dimension D (FEvals/D) to reach a target value $f_{\text{opt}}+\Delta f$ with $\Delta f = 10^k$, where $k \in \{1, -1, -4, -8\}$ is given by the first value in the legend, for pso-bfgs ($\circ$) and de-bfgs ($\triangledown$). Light beige lines show the ECDF of FEvals for target value $\Delta f = 10^{-8}$ of all algorithms benchmarked during BBOB-2009. Right sub-columns: ECDF of FEval ratios of pso-bfgs divided by de-bfgs, all trial pairs for each function. Pairs where both trials failed are disregarded, pairs where one trial failed are visible in the limits being > 0 or < 1. The legends indicate the number of functions that were solved in at least one trial (pso-bfgs first).

5-D

Δf	1e+1	1e-1	1e-3	1e-5	1e-7	#succ
f_1	11	12	12	12	12	15/15
1: pso	3.1(3)	5.2(0.2)	5.2(0.2)	5.2(0.2)	5.2(0.2)	15/15
2: de	3.3(2)	4.9(0.2)	4.9(0.2)	4.9(0.2)	4.9(0.2)	15/15
f_2	83	88	90	92	94	15/15
1: pso	1.7(0.7)	1.8(0.7)	1.9(0.7)	2.1(0.7)	2.3(0.8)	15/15
2: de	1.4(0.3)	1.5(0.3)	1.7(0.3)	1.9(0.3)	2.0(0.3)	15/15
f_3	716	1637	1646	1650	1654	15/15
1: pso	9.4(10)	121(155)	120(159)	120(158)	120(154)	13/15
2: de	**2.0**(1)*	**7.7**(5)$^{\star 3}$	**7.6**(5)$^{\star 3}$	**7.6**(5)$^{\star 3}$	**7.6**(5)$^{\star 3}$	15/15
f_4	809	1688	1817	1886	1903	15/15
1: pso	14(12)	410(479)	381(414)	367(399)	364(392)	7/15
2: de	**3.8**(2)$^{\star 2}$	**20**(17)$^{\star 3}$	**19**(15)$^{\star 3}$	**18**(15)$^{\star 3}$	**18**(15)$^{\star 3}$	15/15
f_5	10	10	10	10	10	15/15
1: pso	6.8(0.3)	8.8(0.6)	8.9(0.6)	8.9(0.6)	8.9(0.6)	15/15
2: de	6.5(2)	9.0(2)	9.0(2)	9.0(2)	9.0(2)	15/15
f_6	114	281	580	1038	1332	15/15
1: pso	6.4(9)	3.1(4)	1.9(2)	1.5(2)	54(98)	11/15
2: de	5.1(2)	2.5(0.9)	1.5(0.6)	1.3(2)	2.8(3)	15/15
f_7	24	1171	1572	1572	1597	15/15
1: pso	589(940)	930(922)	2363(2591)	2363(2733)	2326(2372)	1/15
2: de	**8.9**(6)$^{\star 3}$	**10**(22)$^{\star 3}$	**9.2**(17)$^{\star 3}$	**9.2**(17)$^{\star 3}$	**41**(64)$^{\star 3}$	10/15
f_8	73	336	391	410	422	15/15
1: pso	3.0(2)	1.6(0.8)	1.5(0.7)	1.5(0.7)	1.4(0.7)	15/15
2: de	2.8(2)	1.1(0.5)	1.1(0.4)	1.1(0.4)	1.1(0.4)	15/15
f_9	35	214	300	335	369	15/15
1: pso	4.1(1)	1.6(0.9)	1.3(0.6)	1.2(0.5)	1.1(0.5)	15/15
2: de	4.2(0.7)	1.9(0.9)	1.5(0.7)	1.3(0.6)	1.2(0.5)	15/15
f_{10}	349	574	626	829	880	15/15
1: pso	1.3(1)	0.84(0.9)	0.79(0.8)	0.62(0.6)	1.7(0.6)	15/15
2: de	0.75(0.7)$^{\downarrow}$	0.49(0.4)$^{\downarrow 2}$	0.48(0.4)$^{\downarrow 2}$	0.39(0.3)$^{\downarrow 2}$	0.50(0.9)$^{\downarrow}$	15/15
f_{11}	143	763	1177	1467	1673	15/15
1: pso	0.60(0.1)$^{\downarrow 3}$	0.13(0.0)$^{\downarrow 4}$	0.10(0.0)$^{\downarrow 4}$	0.09(0.0)$^{\downarrow 4}$	0.11(0.0)$^{\downarrow 4}$	15/15
2: de	0.70(0.2)$^{\downarrow}$	0.15(0.0)$^{\downarrow 4}$	0.11(0.0)$^{\downarrow 4}$	0.11(0.0)$^{\downarrow 4}$	0.12(0.1)$^{\downarrow 4}$	15/15
f_{12}	108	371	461	1303	1494	15/15
1: pso	2.8(2)	1.5(1)	1.7(1)	0.73(0.5)	0.86(0.9)	15/15
2: de	2.8(1)	1.9(1)	1.9(1)	0.80(0.5)	1.1(0.7)	15/15
f_{13}	132	250	1310	1752	2255	15/15
1: pso	0.99(0.1)	0.84(0.1)$^{\downarrow 4}$	0.22(0.0)$^{\downarrow 4}$	181(286)	∞*5.0e5*	0/15
2: de	1.0(0.2)	0.87(0.1)$^{\downarrow 2}$	0.23(0.0)$^{\downarrow 4}$	2.6(1.0)	**529**(597)$^{\star 2}$	0/15
f_{14}	10	58	139	251	476	15/15
1: pso	2.3(3)	1.6(0.2)	0.98(0.1)	0.76(0.1)	134(203)	4/15
2: de	1.0(1)	1.5(0.3)	0.95(0.1)	0.71(0.1)	6.4(10)	13/15
f_{15}	511	19369	20073	20769	21359	14/15
1: pso	10(7)	39(36)	37(38)	36(33)	35(32)	8/15
2: de	**3.8**(3)*	**1.6**(1)$^{\star 2}$	**1.6**(1)$^{\star 2}$	**1.5**(1)$^{\star 2}$	**1.5**(1)$^{\star 2}$	15/15
f_{16}	120	2662	10449	11644	12095	15/15
1: pso	6.9(8)	75(97)	111(130)	298(323)	593(622)	1/15
2: de	8.0(8)	**21**(9)*	33(48)	30(43)	585(663)	0/15
f_{17}	5.2	899	3669	6351	7934	15/15
1: pso	1.5(1.0)	21(19)	13(21)	9.2(12)	907(979)	0/15
2: de	3.0(3)	35(110)	**10**(27)*	**6.9**(16)*	145(160)	0/15
f_{18}	103	3968	9280	10905	12469	15/15
1: pso	24(18)	16(10)	38(38)	53(58)	∞*5.0e5*	0/15
2: de	**5.7**(7)*	28(64)	19(27)	20(24)	587(645)	0/15
f_{19}	1	242	1.2e5	1.2e5	1.2e5	15/15
1: pso	69(94)	60(51)	3.6(3)	3.6(3)	19(19)	1/15
2: de	40(38)	71(65)	4.3(4)	4.3(4)	11(11)	2/15
f_{20}	16	38111	54470	54861	55313	14/15
1: pso	4.9(0.9)	3.5(3)	2.4(2)	2.4(2)	2.4(2)	14/15
2: de	4.1(2)	10(14)	7.1(7)	7.0(9)	7.0(10)	10/15
f_{21}	41	1674	1705	1729	1757	14/15
1: pso	1.8(2)	1.6(2)	1.6(2)	1.6(2)	29(55)	13/15
2: de	2.2(2)	2.7(3)	2.7(3)	2.7(3)	16(15)	12/15
f_{22}	71	938	1008	1040	1068	14/15
1: pso	1.6(2)	3.4(4)	3.2(4)	3.3(4)	170(237)	5/15
2: de	6.7(15)	23(24)	21(22)	21(22)	131(244)	8/15
f_{23}	3.0	14249	31654	33030	34256	15/15
1: pso	1.4(1)	3.3(4)	30(34)	∞	∞*5.0e5*	0/15
2: de	2.2(2)	2.0(2)	**1.3**(0.9)$^{\star 3}$	**1.4**(0.9)$^{\star 3}$	**96**(117)$^{\star 3}$	0/15
f_{24}	1622	6.4e6	9.6e6	1.3e7	1.3e7	3/15
1: pso	3.2(2)	0.20(0.2)	0.23(0.3)	0.18(0.2)	0.18(0.2)	3/15
2: de	1.3(1)	0.19(0.2)	0.24(0.3)	0.18(0.2)	0.18(0.2)	3/15

20-D

Δf	1e+1	1e-1	1e-3	1e-5	1e-7	#succ
f_1	43	43	43	43	43	15/15
1: pso	1.9(0.2)	2.2	2.2	2.2	2.2	15/15
2: de	2.0(0.2)	2.2	2.2	2.2	2.2	15/15
f_2	385	387	390	391	393	15/15
1: pso	5.2(3)	5.6(3)	6.7(5)	7.6(5)	12(12)	15/15
2: de	5.7(4)	6.2(4)	6.5(4)	7.1(5)	10(5)	15/15
f_3	5066	7635	7643	7646	7651	15/15
1: pso	∞	∞	∞	∞	∞*2.0e6*	0/15
2: de	**56**(24)$^{\star 3}$	**181**(159)$^{\star 3}$	**181**(150)$^{\star 3}$	**181**(151)$^{\star 3}$	**191**(159)$^{\star 3}$	13/15
f_4	4722	7666	7700	7758	1.4e5	9/15
1: pso	∞	∞	∞	∞	∞*2.0e6*	0/15
2: de	**143**(83)$^{\star 3}$	**3893**(4441)$^{\star 3}$	**3876**(3901)$^{\star 3}$	**3847**(4259)$^{\star 3}$	**212**(213)$^{\star 3}$	1/15
f_5	41	41	41	41	41	15/15
1: pso	22(16)	31(32)	33(36)	33(36)	33(36)	15/15
2: de	14(8)	24(16)	24(16)	26(20)	26(20)	15/15
f_6	1296	3413	5220	6728	8409	15/15
1: pso	9.4(8)	40(36)	148(193)	198(267)	352(392)	0/15
2: de	6.8(5)	17(16)	32(36)	96(152)	295(334)	2/15
f_7	1351	9503	16524	16524	16969	15/15
1: pso	∞	∞	∞	∞	∞*2.0e6*	0/15
2: de	**502**(747)$^{\star 3}$	$\infty^{\star 3}$	$\infty^{\star 3}$	$\infty^{\star 3}$	∞***2.0e6***$^{\star 3}$	0/15
f_8	2039	4040	4219	4371	4484	15/15
1: pso	1.6(1)	4.7(5)	6.0(4)	6.3(4)	9.0(12)	15/15
2: de	1.5(1)	2.8(4)	3.2(3)	3.2(3)	3.2(3)	15/15
f_9	1716	3277	3455	3594	3727	15/15
1: pso	1.7(0.7)	5.7(4)	7.1(5)	6.9(5)*	6.9(4)	15/15
2: de	1.8(1)	2.8(1)	**2.7**(1)*	**2.7**(1)*	**2.7**(1)*	15/15
f_{10}	7413	10735	14920	17073	17476	15/15
1: pso	29(15)	20(10)	14(7)	12(6)	12(6)	14/15
2: de	6.1(6)	4.3(4)	3.1(3)	2.7(3)	2.6(3)	15/15
f_{11}	1002	6278	9762	12285	14831	15/15
1: pso	0.25(0.2)$^{\downarrow 3}$	0.05(0.0)$^{\downarrow 4}$	0.04(0.0)$^{\downarrow 4}$	0.04(0.0)$^{\downarrow 4}$	0.05(0.0)$^{\downarrow 4}$	15/15
2: de	0.20(0.1)$^{\downarrow 4}$	0.05(0.0)$^{\downarrow 4}$	0.04(0.0)$^{\downarrow 4}$	0.04(0.0)$^{\downarrow 4}$	0.05(0.0)$^{\downarrow 4}$	15/15
f_{12}	1042	2740	4140	12407	13827	15/15
1: pso	2.2(2)	2.3(2)	3.7(3)	5.0(4)	21(24)	13/15
2: de	1.4(0.8)	1.6(1)	1.7(1)	**1.1**(0.6)$^{\star 3}$	**4.6**(4)$^{\star 2}$	15/15
f_{13}	652	2751	18749	24455	30201	15/15
1: pso	1.2(0.1)	0.51(0.0)$^{\downarrow 4}$	0.21(0.1)$^{\downarrow 3}$	133(166)	∞*2.0e6*	0/15
2: de	1.3(0.1)	0.53(0.0)$^{\downarrow 4}$	0.18(0.2)$^{\downarrow 4}$	92(116)	∞*2.0e6*	0/15
f_{14}	75	304	932	1648	15661	15/15
1: pso	1.7(0.6)	0.98(0.2)	0.63(0.1)$^{\downarrow 4}$	0.57(0.1)$^{\downarrow 4}$	892(958)	0/15
2: de	1.7(0.4)	0.90(0.2)	0.62(0.1)$^{\downarrow 4}$	0.56(0.1)$^{\downarrow 4}$	125(151)	0/15
f_{15}	30378	3.1e5	3.2e5	4.5e5	4.6e5	15/15
1: pso	∞	∞	∞	∞	∞*2.0e6*	0/15
2: de	**28**(15)$^{\star 3}$	$\infty^{\star 3}$	$\infty^{\star 3}$	$\infty^{\star 3}$	∞***2.0e6***$^{\star 3}$	0/15
f_{16}	1384	77015	1.9e5	2.0e5	2.2e5	15/15
1: pso	542(550)	∞	∞	∞	∞*2.0e6*	0/15
2: de	595(508)	∞	∞	∞	∞*2.0e6*	0/15
f_{17}	63	4005	30677	56288	80472	15/15
1: pso	19(15)	∞	∞	∞	∞*2.0e6*	0/15
2: de	17(19)	**327**(293)$^{\star 3}$	**952**(1028)$^{\star 3}$	$\infty^{\star 3}$	∞***2.0e6***$^{\star 3}$	0/15
f_{18}	621	19561	67569	1.3e5	1.5e5	15/15
1: pso	210(159)	∞	∞	∞	∞*2.0e6*	0/15
2: de	**69**(61)*	**1503**(1689)$^{\star 3}$	∞	∞	∞*2.0e6*	0/15
f_{19}	1	3.4e5	6.2e6	6.7e6	6.7e6	15/15
1: pso	3614(6276)	**1.4**(1)*	$\infty^{\star 2}$	$\infty^{\star 2}$	∞***2.0e6***$^{\star 2}$	0/15
2: de	3206(3658)	5.4(6)	∞	∞	∞*2.0e6*	0/15
f_{20}	82	3.1e6	5.5e6	5.6e6	5.6e6	14/15
1: pso	7.1(1)	∞	∞	∞	∞*2.0e6*	0/15
2: de	6.3(4)	∞	∞	∞	∞*2.0e6*	0/15
f_{21}	561	14103	14643	15567	17589	15/15
1: pso	1.7(1)	0.56(0.9)	0.55(0.9)	0.57(1)	96(119)	1/15
2: de	10(11)	17(26)	16(25)	15(24)	205(247)	1/15
f_{22}	467	23491	24948	26847	1.3e5	12/15
1: pso	1.7(2)	**10**(13)$^{\star 3}$	**9.0**(13)$^{\star 3}$	**8.6**(12)$^{\star 3}$	209(238)	0/15
2: de	5.7(13)	151(143)	142(134)	132(124)	∞*2.0e6*	0/15
f_{23}	3.2	67457	4.9e5	8.1e5	8.4e5	15/15
1: pso	1.6(2)	66(74)	∞	∞	∞*2.0e6*	0/15
2: de	2.1(2)	54(64)	∞	∞	∞*2.0e6*	0/15
f_{24}	1.3e6	5.2e7	5.2e7	5.2e7	5.2e7	3/15
1: pso	∞	∞	∞	∞	∞*2.0e6*	0/15
2: de	4.3(5)	∞	∞	∞	∞*2.0e6*	0/15

Table 1: ERT **in number of function evaluations divided by the best** ERT **measured during BBOB-2009 given in the respective first row and the half inter-80%ile in brackets for different Δf values. #succ is the number of trials that reached the final target $f_{\mathrm{opt}}+10^{-8}$. 1:pso is pso-bfgs and 2:de is de-bfgs. Bold entries are statistically significantly better compared to the other algorithm, with $p = 0.05$ or $p = 10^{-k}$ where $k \in \{2, 3, 4, \dots\}$ is the number following the $\star$ symbol, with Bonferroni correction of 48. A $\downarrow$ indicates the same tested against the best BBOB-2009.**