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

Draft version *

C. Voglis †
Computer Science Department
University of Ioannina
Ioannina, Greece
voglis@cs.uoi.gr

G.S. Piperagkas
Computer Science Department
University of Ioannina
gpiperag@cs.uoi.gr

K. E. Parsopoulos
Computer Science Department
University of Ioannina
kostasp@cs.uoi.gr

D. G. Papageorgiou
Department of Materials Science and Engineering
University of Ioannina
dpapageo@cc.uoi.gr

I. E. Lagaris
Computer Science Department
University of Ioannina
lagaris@cs.uoi.gr

ABSTRACT

In this paper we present an experimental comparison between two well known population-based schemes, namely Particle Swarm Optimization (PSO) and Differential Evolution (DE), that are incorporated in a memetic global optimization framework. We use the recently published MEMPSODE software [16], that implements the memetic global optimization first described in [13] and incorporates Merlin optimization environment [10]. Since the original description of the algorithm in [13] involved only a PSO variant for the exploration phase, using MEMPSODE software we attempt an empirical assessment of the DE. The results based on the noiseless testbed, indicate that the usage of DE may lead to superior performance.

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**]

*Submission deadline: March 28th.

†Corresponding author.

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

Keywords

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

1. INTRODUCTION

Evolutionary Algorithms (EAs) and Swarm Intelligence (SI) approaches have been established as powerful optimization tools for solving optimization problems [2, 8, 12]. They are based on models that draw their inspiration from physical systems. Based on natural selection and evolution schemes these algorithms exhibit remarkable capability of locating global solutions for optimization problems.

The rise of EA and SI algorithms has sparked the development of a closely related category, namely the *Memetic Algorithms* (MAs). MAs constitute a class of hybrid meta–heuristics that combine population–based optimization algorithms with local search procedures [9]. The rationale behind their development was the necessity for powerful algorithms where the global exploration capability of EAs and SI approaches would be complemented with the efficiency and accuracy of classical local optimization techniques.

In this work we examine the performance of a memetic optimization software that incorporates Particle Swarm Optimization (PSO) or Differential Evolution (DE) schemes for exploration in addition to powerful local optimization algorithms for exploitation. Our primary target is to determine the impact of the choice between UPSO and DE in the algorithmic framework presented in [13]. By no means we are attempting an extensive benchmark of all MEMPSODE user defined parameters. Instead we use a default set of parameters both on PSO and on DE and the same local search procedure.

2. ALGORITHM PRESENTATION

The tested software follows closely the PSO–based memetic approaches reported in [13] and extends them also to the DE

framework. More specifically, the Unified PSO (UPSO) approach [11], which harnesses the strengths of standard local and global PSO variants is implemented. In both cases, direct calls to LS procedures are facilitated via the established *Merlin* optimization environment [10], providing the ability to develop a variety of MAs. In order to present the algorithm implemented by MEMSPODE we provide summary for all key algorithms incorporated. A pseudo-code summarising the methodology is presented in Algorithm 1.

2.1 Unified Particle Swarm Optimization

PSO was introduced by Eberhart and Kennedy [1]. The main concept of the method includes a population, also called *swarm*, of search points, also called *particles*, searching for optimal solutions within the search space, simultaneously. The particles move in the search space by assuming an adaptable position shift, called *velocity*, at each iteration.

Moreover, each particle retains in a memory the best position it has ever visited, i.e., the position with the lowest function value. To each particle is assigned a *neighborhood*, which determines the indices of its mates that will share its experience. Obviously, the neighborhood scheme affects the flow of information among the particles. Two well known neighborhood scheme have been used extensively. The local *lbest* scheme were each particle is assumed to communicate only with its mates with adjacent indices and the global *gbest* scheme were any new information (best position) is immediately communicated to every single particle in each iteration.

Putting our description in a mathematical framework, let us assume the n–dimensional continuous optimization problem:

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

where the search space X is an orthogonal hyperbox in $\mathbb{R}^n$:

$$X \equiv [l_1, r_1] \times [l_2, r_2] \times \cdots \times [l_n, r_n].$$

A swarm of N particles is a set of search points:

$$S = \{x_1, x_2, \ldots, x_N\},$$

where the i–th particle is defined as:

$$x_i = (x_{i1}, x_{i2}, \ldots, x_{in})^\top \in X, \qquad i = 1, 2, \ldots, N.$$

The velocity (position shift) of x_i is denoted as:

$$v_i = (v_{i1}, v_{i2}, \ldots, v_{in})^\top, \qquad i = 1, 2, \ldots, N,$$

and its best position as:

$$p_i = (p_{i1}, p_{i2}, \ldots, p_{in})^\top \in X, \qquad i = 1, 2, \ldots, N.$$

Let $g_i = \arg\min_{j \in \mathcal{N}_i} f(p_j)$, and t denote the algorithm's iteration counter.

The classical PSO model can be generalized in the UPSO scheme [11]. Following this scheme and 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:

$$G_{ij}^{(t+1)} = \chi \left[v_{ij}^{(t)} + c_1 r_1 \left(p_{ij}^{(t)} - x_{ij}^{(t)} \right) + c_2 r_2 \left(p_{gj}^{(t)} - x_{ij}^{(t)} \right) \right], \tag{2}$$

$$L_{ij}^{(t+1)} = \chi \left[v_{ij}^{(t)} + c_1 r_1 \left(p_{ij}^{(t)} - x_{ij}^{(t)} \right) + c_2 r_2 \left(p_{g_i j}^{(t)} - x_{ij}^{(t)} \right) \right], \tag{3}$$

where g is the index of the overall best particle, i.e.:

$$g = \arg \min_{j=1,\ldots,N} f(p_j),$$

then the particle is updated as follows [11]:

$$U_{ij}^{(t+1)} = u\, G_{ij}^{(t+1)} + (1-u)\, L_{ij}^{(t+1)}, \tag{4}$$

$$x_{ij}^{(t+1)} = x_{ij}^{(t)} + U_{ij}^{(t+1)}, \tag{5}$$

$$i = 1, 2, \ldots, N, \qquad j = 1, 2, \ldots, n.$$

The parameter $u \in [0, 1]$ is called the *unification factor* and it balances the influence (trade–off) of the global and local 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.

2.2 Differential Evolution

The Differential Evolution (DE) algorithm was introduced by Storn and Price [14] 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:

$$x_i = (x_{i1}, x_{i2}, \ldots, x_{in})^\top \in X, \qquad i = 1, 2, \ldots, N,$$

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, the new and the old individuals are merged and selection takes place 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, $i = 1, 2, \ldots N$, by combining some of the rest individuals of the population. There most common (but not the only) operator proposed to accomplish this task:

$$\text{OP1:} \quad v_i^{(t+1)} = x_g^{(t)} + F \left(x_{r_1}^{(t)} - x_{r_2}^{(t)} \right), \tag{6}$$

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 $r_j \in \{1, 2, \ldots, N\}$, $j = 1, 2, \ldots, 5$, are mutually different randomly selected indices that differ also from the index i. Thus, in order to be able to apply all mutation operators, it must hold that $N > 5$.

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

$$u_i = (u_{i1}, u_{i2}, \ldots, u_{in}), \qquad i = 1, 2, \ldots, N,$$

for each individual. This vector is defined as follows:

$$u_{ij}^{(t+1)} = \begin{cases} v_{ij}^{(t+1)}, & \text{if } R_j \leqslant CR \text{ or } j = \mathrm{RI}(i), \\ x_{ij}^{(t)}, & \text{if } R_j > CR \text{ and } j \neq \mathrm{RI}(i), \end{cases} \tag{7}$$

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, each trial vector is compared against the corresponding individual and the best between them comprise

the new individual in the next generation, i.e.:

$$x_i^{(t+1)} = \begin{cases} u_i^{(t+1)}, & \text{if } f\left(u_i^{(t+1)}\right) < f\left(x_i^{(t)}\right), \\ x_i^{(t)}, & \text{otherwise.} \end{cases} \quad (8)$$

2.3 Local search

A local solution to an optimization problem can be obtained by applying local optimization methods. The hybrid schemes implemented in MEMPSODE have a need for deterministic local search procedures that require a starting point, x_0, and generate a sequence of points, $\{x_k\}_{k=0}^{\infty}$, in order to determine a minimizer within a prescribed accuracy. The generation of a new point, x_{k+1}, in the sequence is based on information collected for the current iterate, x_k. Typically, this information includes the function value at x_k, as well as the first– and probably second–order derivatives of $f(x)$ at x_k. 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) [**?**]. The gradient–free algorithms include a pattern search and the nonlinear Simplex method.

2.4 Memetic Algorithm

The design of MPSO in [13] was based on three fundamental schemes, henceforth 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 localy 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 start points for LS, as pointed out in [6]. The memetic strategies proposed in [13] were also adopted in MEMPSODE for both PSO– and DE–based MAs.

3. EXPERIMENTAL PROCEDURE

We used the default restart mechanism provided by the testbed for a maximum number of $100\,000 \times n$ function evaluations. The third memetic scheme was used with probability of local search set to $p_i = 0.05$. Both UPSO and DE used a swarm size $N = 25$ particles. In UPSO the unification factor u was set to 1 and the initial velocity vector was restraint by a factor of 0.01. For the DE experiments we applied OP1 with default values $F = 0.5$ and $CR = 0.7$.

Each local seach has an upper limit of 4 000 function evaluations. Whenever derivatives were needed (eg. BFGS) we applied an $O(h)$ finite differences formula where h is an adaptable step size(see [15]). For the local search we applied the BFGS method implemented in Merlin.

The experiments have been conducted on an Intel I7-2600 processor on 3.4 GHz with 8GB RAM.

4. RESULTS

Results from experiments according to [4] on the benchmark functions given in [3, 5] are presented in Figures 1, 2 and 3 and in Tables 1. The **expected running time (ERT)**, used in the figures and table, 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 [4].

A direct comparison between UPSO and DE variants of MEMPSODE memetic algorithm can be deduced by observing the scatter plots in figure 2 and the starred records in table 1. In the separable case DE variant outperforms UPSO especially as dimensionality increases. For the moderate category DE seems to outperform 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 better. Finally, DE variant outperforms PSO variant in all multimodal functions but PSO variant seems to behave better for the weak structured cases.

DE variant superiority is also obvious by inspecting figure 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 ETF for some accuracy levels in the case of ill condition 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.

As a general remark, MEMPSODE seems a very promising and competitive new algorithm that incorporates state of the art schemes of swarm intelligence algorithms with the robust and versatile Merlin optimization environment.

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

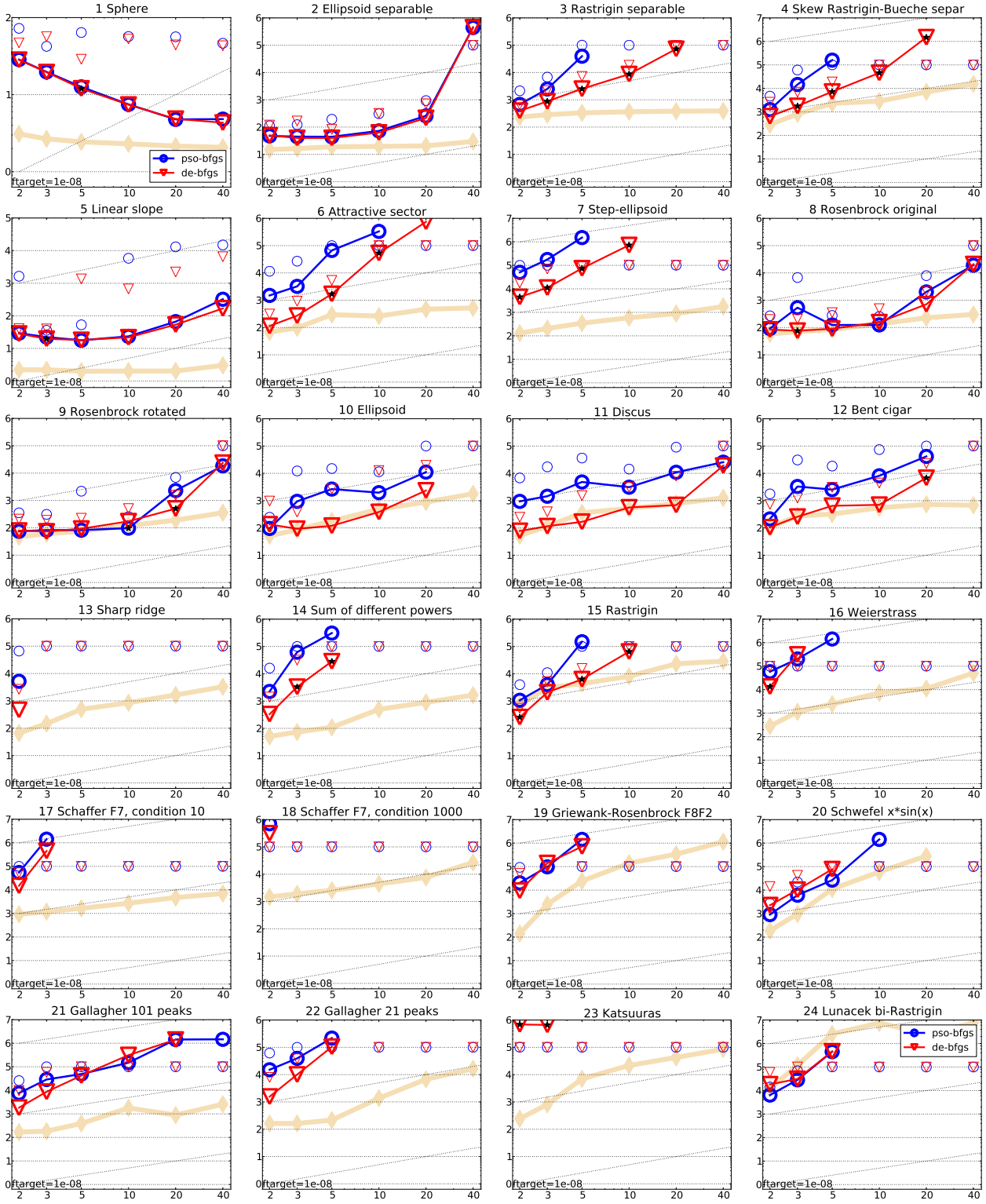

Figure 1: Expected running time (ERT in number of f-evaluations) divided by dimension for target function value 10^{-8} as $\log_{10}$ values versus dimension. Different symbols correspond to different algorithms given in the legend of f_1 and f_{24}. Light symbols give the maximum number of function 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.

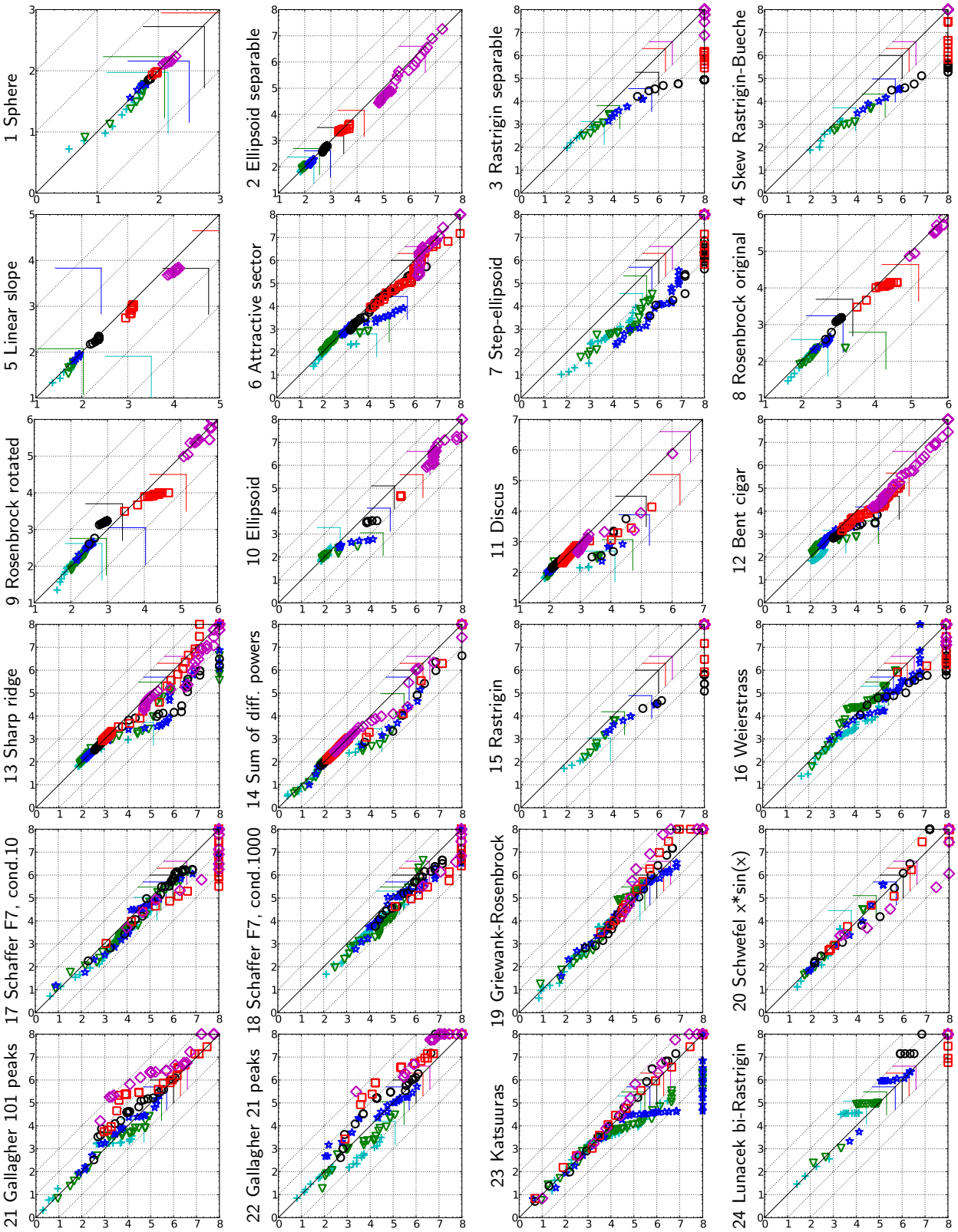

Figure 2: Expected running time (ERT in $\log_{10}$ **of number of function evaluations) of pso-bfgs** (x**-axis) versus de-bfgs** (y**-axis) for 46 target values** $\Delta f \in [10^{-8}, 10]$ **in each dimension on functions** f_1–f_{24}**. Markers on the upper or right edge indicate that the target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

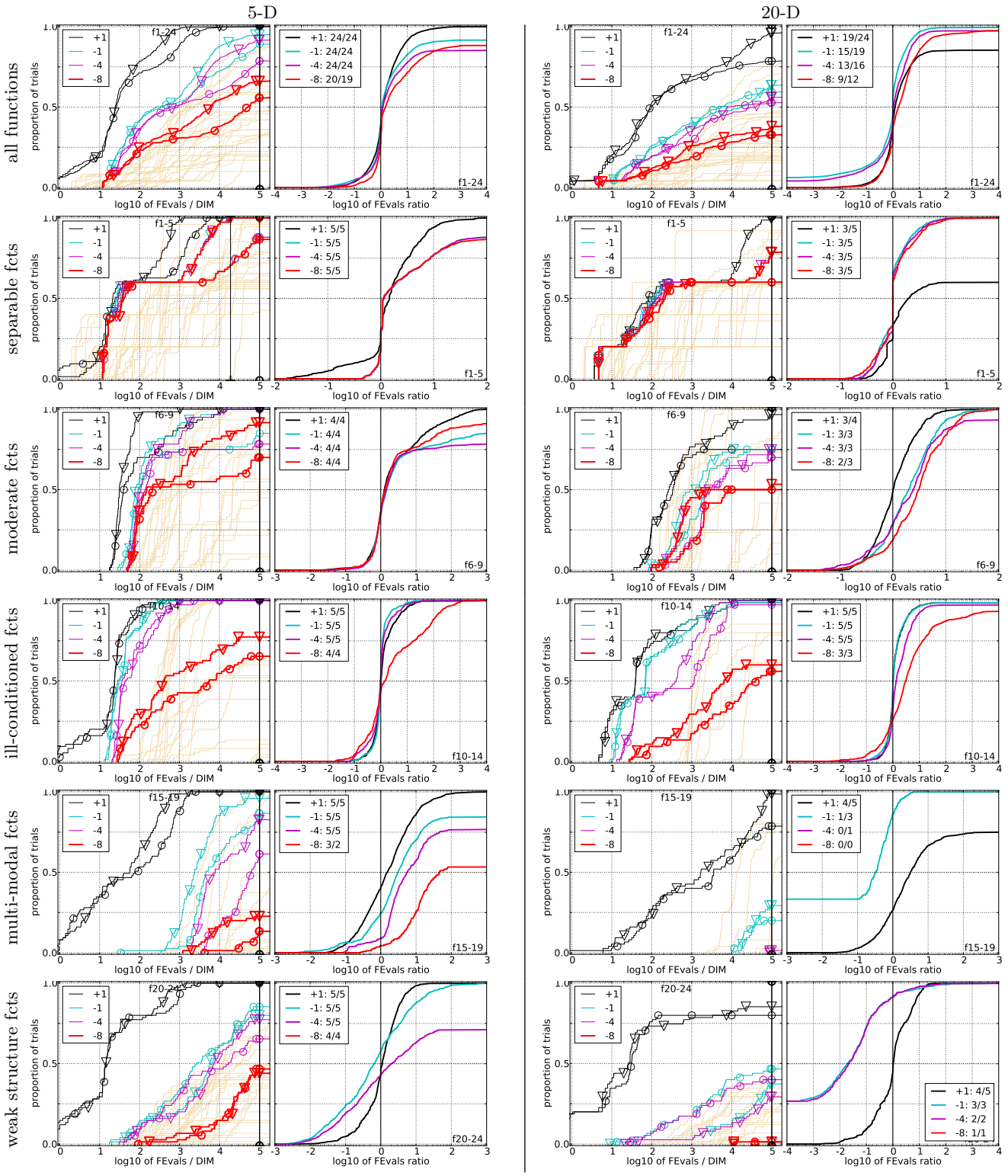

Figure 3: Empirical cumulative distributions (ECDF) of run lengths and speed-up ratios in 5-D (left) and 20-D (right). Left sub-columns: ECDF of the number of function evaluations divided by dimension D (FEvals/D) to reach a target value $f_{\text{opt}}+\Delta f$ with $\Delta f = 10^k$, where $k \in \{1, -1, -4, -8\}$ 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 $f_{\text{opt}}+10^{-8}$. 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.**

[5] 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.

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

[7] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *Proc. IEEE Int. Conf. Neural Networks*, volume IV, pages 1942–1948, Piscataway, NJ, 1995. IEEE Service Center.

[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. *Particle Swarm Optimization and Intelligence: Advances and Applications*. Information Science Publishing (IGI Global), 2010.

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

[14] 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.

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

[16] 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.

Algorithm 1: Pseudocode of the implemented memetic algorithm

Input: Objective function, $f : S \subset \mathbb{R}^n \to \mathbb{R}$; algorithm PSO/DE: *algo* ; swarm size: N; memetic strategy: `memetic`, maximum function evaluations: `maxfev`; unification factor: `UF`; use mutation: mut; probability for local search: ρ

Output: Best detected solution: x^*, $f(x^*)$.

`// Initialization`

for $i = 1, 2, \ldots, N$ **do**
  Initialize position x_i and velocity u_i
  Set $p_i \leftarrow x_i$ `// Initialize best position`
  $f_i \leftarrow f(x_i(0))$ `// Evaluate particle`
  $f_i^p \leftarrow f_i$ `// Best position`
  $local_i \leftarrow 0$ `// Best position is minimum is set to false`
end

`// Update Best Indices`

Calculate global best index g_1 and local best index g_2

`// Main Iteration Loop`

Set $t \leftarrow 0$

while *termination criterion* **do**
  `// Update Swarm`
  if *algo = 'pso'* **then**
    for $i = 1, 2, \ldots, N$ **do**
      Calculate local best velocity update for particle u_i^l using g_1
      Calculate global best velocity update for particle u_i^g using g_2
      if $mut = 1$ **then**
        `// Unified PSO with mutation`
        $R \leftarrow N(\mu, \sigma)$
        if $rand() \leqslant 0.5$ **then**
          $u_i \leftarrow R\texttt{UF}u_i^l + (1 - \texttt{UF})u_i^g$ `// Unified PSO + Mutate local term`
        else
          $u_i \leftarrow \texttt{UF}u_i^l + R(1 - \texttt{UF})u_i^g$ `// Unified PSO + Mutate global term`
        end
      else
        $u_i \leftarrow \texttt{UF}u_i^l + (1 - \texttt{UF})u_i^g$ `// Unified PSO`
      end
      $x_i = x_i + u_i$ `// Update particle's position`
    end
  else if *algo = 'de'* **then**
    for $i = 1, 2, \ldots, N$ **do**
      $x_i \leftarrow p_i$ `// Replicate best positions` p `to swarm array` x
    end
    for $i = 1, 2, \ldots, N$ **do**
      Calculate u_i using a strategy from Eqs. (6)–(**??**)
      if $rand() \leqslant C$ **then**
        $x_i \leftarrow u_i$
      else
        $x_i \leftarrow p_i$
      end
    end
  end
  `// Evaluate Swarm`
  for $i = 1, 2, \ldots, N$ **do**
    $f_i \leftarrow f(x_i)$ `// Evaluate particle`
  end
  `// Update Best Positions`
  for $i = 1, 2, \ldots, N$ **do**
    if $f_i < f(p_i)$ **then**
      $p_i \leftarrow x_i$
      $f_i^p \leftarrow f_i$
    end
  end
  Calculate global best index g_1 and local best index g_2
  Apply one of the schemes using ρ
  Calculate global best index g_1 and local best index g_2
  `// If all best positions are local minima, restart`
  if $\sum_{i=1}^{N} local_i = N$ **then**
    Keep global best particle and reinitialize the swarm
  end
end