Benchmarking the Multi-View Differential Evolution on the Noiseless BBOB-2012 Function Testbed

Vinícius Veloso de Melo
Institute of Science and Technology (ICT)
Federal University of São Paulo (UNIFESP)
São José dos Campos, SP, Brazil
vinicius.melo@unifesp.br

ABSTRACT

We propose a multi-view DE in which several mutation strategies are applied to the same current population to generate different views for the current iteration. The views are then merged using tournament-selection to generate the next single population. This Multi-View DE is benchmarked on the BBOB-2012 noiseless function testbed.

Categories and Subject Descriptors

G.1.6 [**Numerical Analysis**]: Optimization—*global optimization, unconstrained optimization*; F.2.1 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems

General Terms

Algorithms

Keywords

Benchmarking, Black-box optimization, Evolutionary computation, Differential Evolution

1. INTRODUCTION

Differential Evolution was introduced by Storn and Price in 1995 [10]. It is a floating-point encoding populational metaheuristic, similar to classical evolutionary algorithms, successfully used to solve several benchmarks and real-world problems [8, 9, 11, 1, 6].

A population P of size N and D dimensions is randomly initialized (using a uniform distribution) inside the problem's bounds and evaluated using the fitness function for the problem. Then, until a stop condition is satisfied, the algorithm performs an iterative evolutionary process of mutation, crossover and selection operators.

Several mutation strategies have been proposed, but the classical DE allows the selection of only a single mutation for a run. In this paper we propose an approach to combine the effort of several mutation strategies at the same time, but not at the same population. Moreover, it does not use subpopulations nor co-evolution of different populations. We call the approach Multi-View Differential Evolution (MVDE).

In this paper, MVDE is benchmarked on 24 functions. Results for the number of function evaluations to reach a target function value and estimated time complexity are presented.

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

2. THE MVDE

The idea consists of employing different strategies to generate new vector solutions from the same population, thus providing different views for the same problem. The proposed algorithm is presented in Algorithm 1.

Three modifications in the classical DE are proposed. First of all, a large sample of points in the search-space is created. The fitness is calculated and the N best solutions are selected to be MVDE's population, as proposed in [5]. This allows for a better and possibly unbiased starting sampling of the search-space.

The second and main modification is the generation of multi-views. A view is the resulting set of donor vectors generated from the application of a mutation strategy on the current population. This means that the application of the mutation strategy *MS1* on the current population of size N results in a set of N donor vectors belonging to the view *V1*.

In this paper each view correspond to one of the following strategies: rand/1; local-to-best/1; and current-to-best/1, thus three views (V1, V2, and V3). The Multi-View approach may provide better exploration of the search-space and escape from local optima. This can be achieved by using mutation strategies with such characteristic (*rand*/1, for instance) while performing local exploitation inside views guided toward the best solution found (*local-to-best*/1, for instance).

The third and last modification is that after all views are generated and evaluated, the best vector of all views in the ongoing iteration and the current best solution (x_{best}), are saved (elitism). Then, tournament-selection is applied to choose the remaining best vectors from $V1[i]$, ..., $Vv[i]$, where $i = 1...N$ and v is the number of views. The two worst solutions must be removed from the population to maintain size N. The usual crossover operator is then applied to this merged population.

3. PARAMETER SETTING

No parameter study has been performed. An empirical set of runs suggested that better success rate could be achieved

Algorithm 1 Algorithm of MVDE.

```
1: Generate a large population
2: Evaluate the population
3: Select best N solutions to be the actual
   initial population
4: Initialize views (1 to v)
5: Do

/* Generate the views:  */
6:     For each view
7:         For each individual in the population
8:            Apply the mutation strategy corre-
              sponding to the current view and
              save it in the view's population
9:            Evaluate the vector generated
10:        End for
11:    End for

/* Elitism:  */
12:    Save current best solution and the best
       vector of all views in the merged mutated
       population

/* Merge the views:  */
13:    For index = 1 to N in the merged mutated
       population
14:        For each view
15:              Get the vector from position index
16:        End for
17:        Apply tournament-selection in the taken
           vectors
18:        Save the winner in the merged mutated
           population
19:    End for
20:    Remove the two worst vectors from the merged
       mutated population

/* Usual crossover */
21:    Apply crossover to the merged mutated
       population
22: Until the termination condition is reached
```

with longer exploration of the search-space. Therefore we set the parameters as: F=0.5, CR=0.3, N=10D, $budget$ = 1e6, maximum iterations=$budget/(popsize \times 3)$, stagnation=Inf iterations, and value-to-reach=-Inf. This settings are valid for all strategies (views) in MVDE. Currently there is no auto-tuning, self-adaptation, nor local-search.

The first large sample of the search-space is of size $5N$. Then the N-best solutions are selected to be the inital population. MVDE was developed in the R language and was based on the DEoptim package [7].

4. CPU TIMING EXPERIMENT

For this experiment, MVDE was run on f_8 and restarted until at least 35 seconds had passed. The experiments have been conducted with an Intel Xeon E5645 @ 2.40GHz under Linux Ubuntu 11.04 (GNU/Linux 2.6.38-8-server) x86_64 using R 2.14.2 x86_64. The average time demands per function evaluation are shown in the following table.

D	2	3	5	10	20	40
$seconds \times 10^{-5}$	6.42	6.24	6.08	6.28	6.79	8.00

For dimensions lower than 40, the required CPU time is very similar, from 6.08 to 6.79e-5 seconds. When doubled from $D = 20$ to $D = 40$, the CPU time increased about 15%, to 8e-5 seconds.

Table 2: ERT loss ratio compared to the respective best result from BBOB-2009 for budgets given in the first column (see also Figure 3). The last row RL_{US}/D gives the number of function evaluations in unsuccessful runs divided by dimension. Shown are the smallest, 10%-ile, 25%-ile, 50%-ile, 75%-ile and 90%-ile value (smaller values are better). The ERT Loss ratio equals to one for the respective best algorithm from BBOB-2009. Typical median values are between ten and hundred.

	f_1–f_{24} in 5-D, maxFE/D=200050					
#FEs/D	best	10%	25%	**med**	75%	90%
2	0.91	1.8	2.3	3.0	4.2	6.0
10	3.0	3.2	3.4	4.9	6.8	16
100	4.2	4.8	8.0	14	21	95
1e3	2.8	3.0	7.9	20	49	1.3e2
1e4	0.37	5.9	11	93	1.8e2	3.3e2
1e5	0.37	9.7	26	1.1e2	6.2e2	8.2e2
1e6	0.37	9.7	29	95	7.4e2	2.9e3
RL_{US}/D	2e5	2e5	2e5	2e5	2e5	2e5
	f_1–f_{24} in 20-D, maxFE/D=50010					
#FEs/D	best	10%	25%	**med**	75%	90%
2	1.0	3.7	12	31	40	40
10	4.8	7.4	15	1.2e2	2.0e2	2.0e2
100	17	30	47	74	1.3e2	2.0e3
1e3	24	31	39	62	1.3e2	4.6e2
1e4	4.3	26	51	1.5e2	3.9e2	7.4e2
1e5	0.36	42	1.6e2	5.8e2	1.8e3	3.1e3
1e6	0.36	42	2.9e2	3.3e3	1.4e4	2.4e4
RL_{US}/D	5e4	5e4	5e4	5e4	5e4	5e4

5. RESULTS

Results of Multi-View Differential Evolution (MVDE) from experiments according to [3] on the benchmark functions given in [2, 4] are presented in Figures 1, 2 and 3 and in Tables 1 and 2.

For some functions, MVDE exibit ERT linear behavior with respect to dimension, whereas to other functions MVDE's behavior seems to be quadratic. Several functions could not be solved in high dimensions. For these cases the budget, and maybe the population size, should be strongly increased, and DE's parameters should be changed. Local-search algorithms may also improve success rate.

6. REFERENCES

[1] U. K. Chakraborty. *Advances in Differential Evolution*. Springer Publishing Company, Incorporated, 2008.

[2] 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.

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

[4] N. Hansen, S. Finck, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking

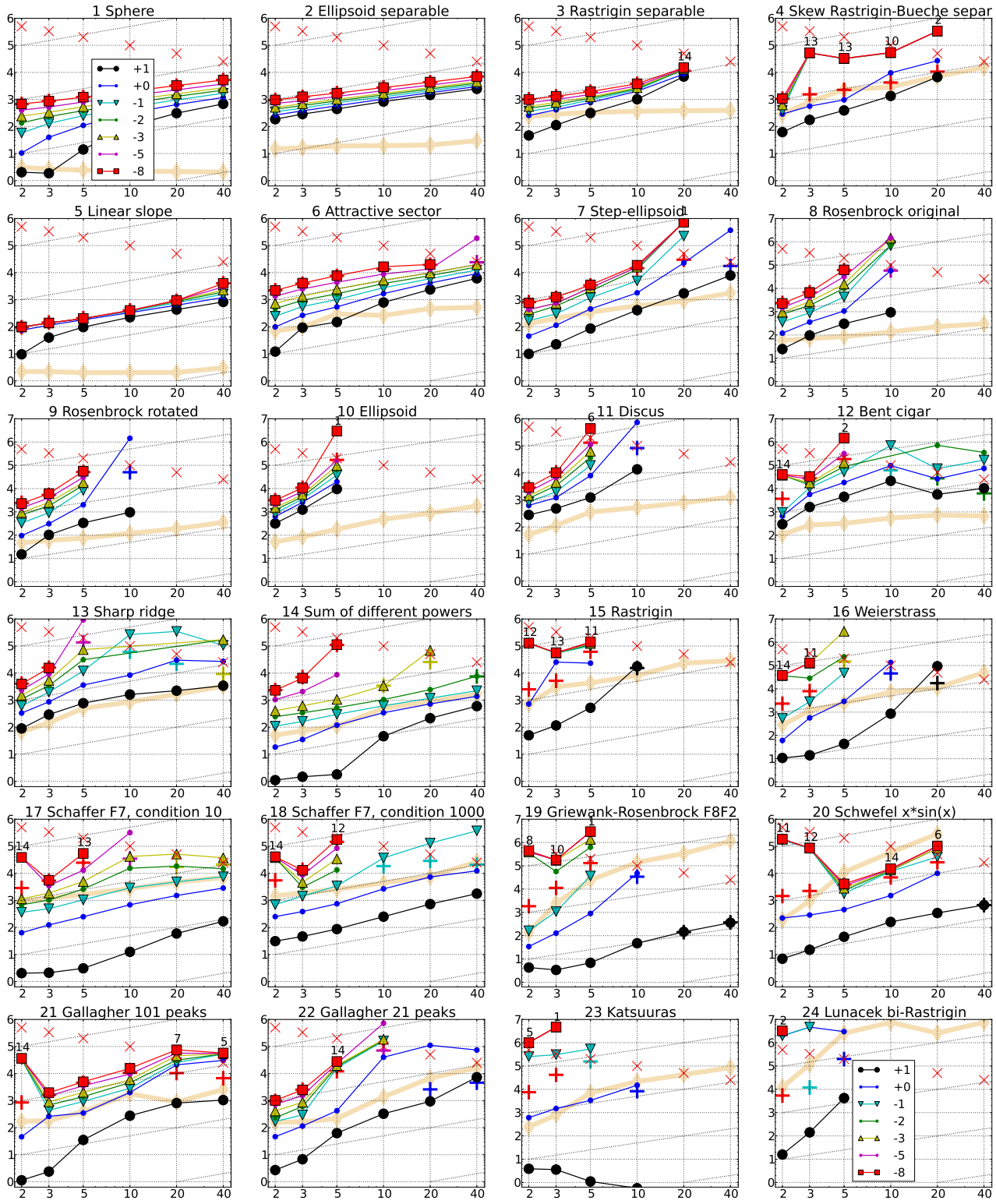

Figure 1: Expected number of f-evaluations (ERT, with lines, see legend) to reach $f_{\mathrm{opt}} + \Delta f$, median number of f-evaluations to reach the most difficult target that was reached at least once ($+$) and maximum number of f-evaluations in any trial ($\times$), all divided by dimension and plotted as $\log_{10}$ values versus dimension. Shown are $\Delta f = 10^{\{1,0,-1,-2,-3,-5,-8\}}$. Numbers above ERT-symbols indicate the number of successful trials. The light thick line with diamonds indicates the respective best result from BBOB-2009 for $\Delta f = 10^{-8}$. Horizontal lines mean linear scaling, slanted grid lines depict quadratic scaling.

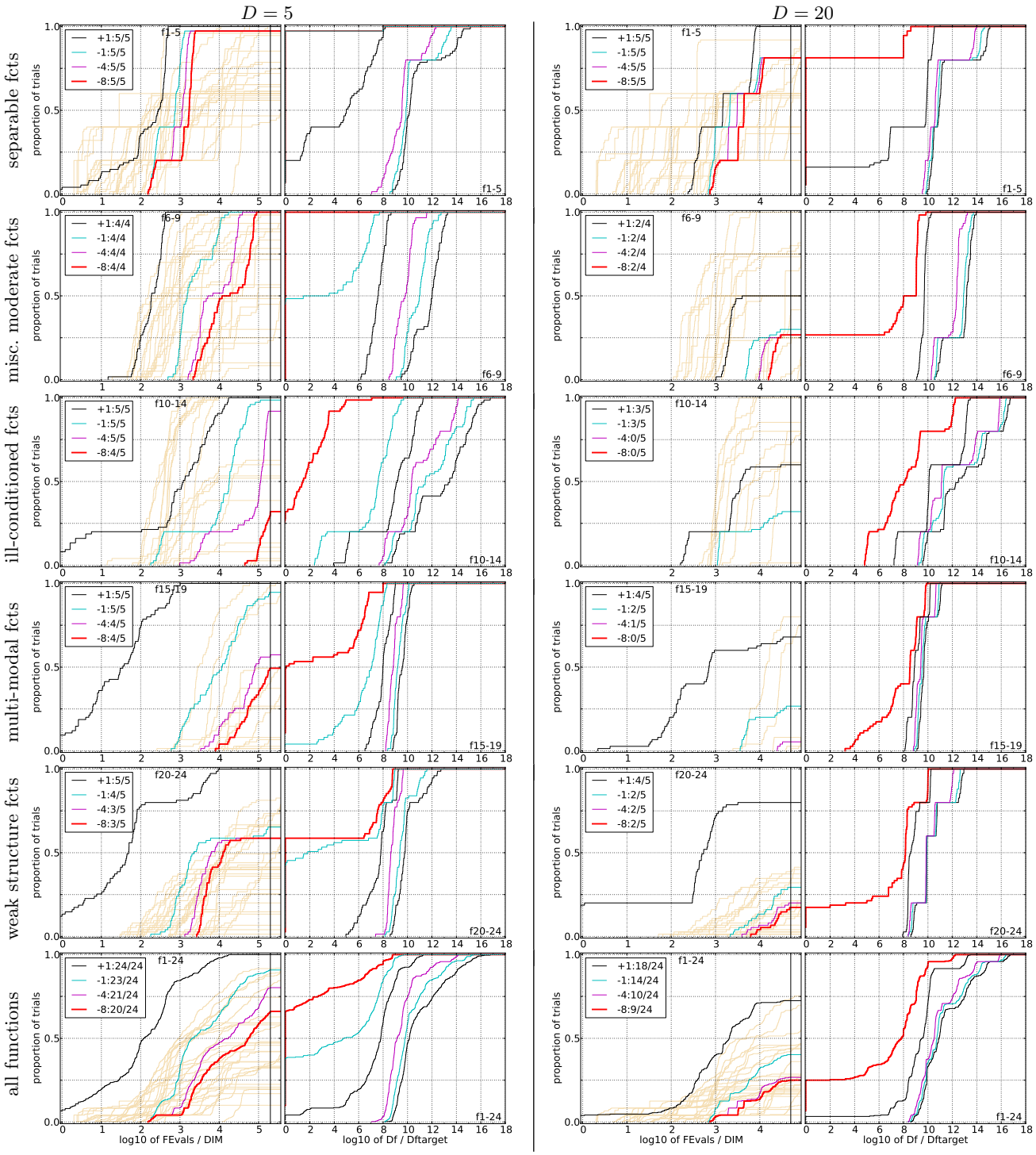

Figure 2: Empirical cumulative distribution functions (ECDFs), plotting the fraction of trials with an outcome not larger than the respective value on the x-axis. Left subplots: ECDF of number of function evaluations (FEvals) divided by search space dimension D, to fall below $f_{\mathrm{opt}} + \Delta f$ with $\Delta f = 10^k$, where k is the first value in the legend. Right subplots: ECDF of the best achieved Δf divided by 10^{-8} for running times of $D, 10\,D, 100\,D, \ldots$ function evaluations (from right to left cycling black-cyan-magenta). The thick red line represents the most difficult target value $f_{\mathrm{opt}} + 10^{-8}$. Legends indicate the number of functions that were solved in at least one trial. Light brown lines in the background show ECDFs for $\Delta f = 10^{-8}$ of all algorithms benchmarked during BBOB-2009.

5-D

Δf	1e+1	1e+0	1e-1	1e-3	1e-5	1e-7	#succ
f_1	11	12	12	12	12	12	15/15
	6.4(8)	45(20)	103(13)	220(20)	329(25)	451(21)	15/15
f_2	83	87	88	90	92	94	15/15
	27(3)	33(4)	42(2)	56(4)	70(4)	83(3)	15/15
f_3	716	1622	1637	1646	1650	1654	15/15
	2.2(0.6)	2.4(0.3)	2.9(0.2)	3.8(0.2)	4.6(0.2)	5.5(0.2)	15/15
f_4	809	1633	1688	1817	1886	1903	15/15
	2.4(0.4)	3.0(0.6)	95(296)	89(275)	86(265)	86(263)	13/15
f_5	10	10	10	10	10	10	15/15
	48(10)	90(16)	99(20)	101(20)	101(20)	101(20)	15/15
f_6	114	214	281	580	1038	1332	15/15
	6.5(3)	13(4)	18(4)	21(4)	21(8)	25(9)	15/15
f_7	24	324	1171	1572	1572	1597	15/15
	18(7)	7.0(4)	5.0(2)	8.7(4)	8.7(4)	9.4(4)	15/15
f_8	73	273	336	391	410	422	15/15
	21(8)	20(16)	64(69)	196(117)	388(152)	589(194)	15/15
f_9	35	127	214	300	335	369	15/15
	49(13)	79(49)	197(94)	304(85)	453(164)	614(232)	15/15
f_{10}	349	500	574	626	829	880	15/15
	137(86)	198(73)	331(134)	780(257)	996(230)	2756(2358)	1/15
f_{11}	143	202	763	1177	1467	1673	15/15
	43(32)	195(125)	123(53)	267(143)	383(201)	795(741)	6/15
f_{12}	108	268	371	461	1303	1494	15/15
	207(116)	342(277)	715(673)	1416(755)	1221(959)	3256(3375)	2/15
f_{13}	132	195	250	1310	1752	2255	15/15
	29(12)	92(44)	240(114)	280(188)	2645(2883)	∞*1.0e6*	0/15
f_{14}	10	41	58	139	251	476	15/15
	0.91(1.0)	15(6)	26(7)	38(9)	173(39)	591(272)	15/15
f_{15}	511	9310	19369	20073	20769	21359	14/15
	5.1(2)	12(12)	26(31)	31(30)	32(30)	32(29)	11/15
f_{16}	120	612	2662	10449	11644	12095	15/15
	1.8(2)	24(11)	90(163)	1411(1484)	∞	∞*1.0e6*	0/15
f_{17}	5.2	215	899	3669	6351	7934	15/15
	2.9(4)	5.8(2)	5.8(2)	6.8(3)	10(6)	32(65)	13/15
f_{18}	103	378	3968	9280	10905	12469	15/15
	4.2(1)	10(4)	4.3(2)	18(10)	39(24)	50(28)	12/15
f_{19}	1	1	242	1.2e5	1.2e5	1.2e5	15/15
	34(26)	4483(4878)	756(722)	56(60)	120(132)	120(135)	1/15
f_{20}	16	851	38111	54470	54861	55313	14/15
	14(8)	2.7(2)	**0.23**$(0.1)^{\downarrow 4}$	0.26(0.1)	0.32(0.1)	0.35(0.1)	15/15
f_{21}	41	1157	1674	1705	1729	1757	14/15
	4.3(3)	1.5(0.8)	2.8(2)	5.9(3)	10(4)	13(4)	15/15
f_{22}	71	386	938	1008	1040	1068	14/15
	4.4(3)	5.4(3)	85(14)	98(52)	113(58)	124(58)	14/15
f_{23}	3.0	518	14249	31654	33030	34256	15/15
	1.8(2)	32(40)	190(179)	∞	∞	∞*1.0e6*	0/15
f_{24}	1622	2.2e5	6.4e6	9.6e6	1.3e7	1.3e7	3/15
	13(10)	69(72)	∞	∞	∞	∞*1.0e6*	0/15

20-D

Δf	1e+1	1e+0	1e-1	1e-3	1e-5	1e-7	#succ
f_1	43	43	43	43	43	43	15/15
	144(16)	299(10)	445(13)	743(10)	1048(21)	1355(25)	15/15
f_2	385	386	387	390	391	393	15/15
	76(2)	92(2)	109(3)	141(2)	173(3)	206(4)	15/15
f_3	5066	7626	7635	7643	7646	7651	15/15
	28(2)	23(3)	33(3)	35(3)	36(3)	38(3)	14/15
f_4	4722	7628	7666	7700	7758	1.4e5	9/15
	28(4)	71(68)	869(1042)	867(1037)	862(967)	48(57)	2/15
f_5	41	41	41	41	41	41	15/15
	212(18)	307(31)	379(42)	449(86)	462(95)	463(95)	15/15
f_6	1296	2343	3413	5220	6728	8409	15/15
	37(7)	35(8)	35(7)	37(7)	40(9)	43(10)	15/15
f_7	1351	4274	9503	16524	16524	16969	15/15
	25(5)	106(122)	470(519)	882(1059)	882(984)	860(914)	1/15
f_8	2039	3871	4040	4219	4371	4484	15/15
	∞	∞	∞	∞	∞	∞*1.0e6*	0/15
f_9	1716	3102	3277	3455	3594	3727	15/15
	∞	∞	∞	∞	∞	∞*1.0e6*	0/15
f_{10}	7413	8661	10735	14920	17073	17476	15/15
	∞	∞	∞	∞	∞	∞*1.0e6*	0/15
f_{11}	1002	2228	6278	9762	12285	14831	15/15
	∞	∞	∞	∞	∞	∞*1.0e6*	0/15
f_{12}	1042	1938	2740	4140	12407	13827	15/15
	108(31)	272(284)	528(582)	∞	∞	∞*1.0e6*	0/15
f_{13}	652	2021	2751	18749	24455	30201	15/15
	68(5)	297(495)	2523(2837)	∞	∞	∞*1.0e6*	0/15
f_{14}	75	239	304	932	1648	15661	15/15
	57(10)	60(6)	78(4)	1465(1535)	∞	∞*1.0e6*	0/15
f_{15}	30378	1.5e5	3.1e5	3.2e5	4.5e5	4.6e5	15/15
	∞	∞	∞	∞	∞	∞*1.0e6*	0/15
f_{16}	1384	27265	77015	1.9e5	2.0e5	2.2e5	15/15
	1353(1560)	∞	∞	∞	∞	∞*1.0e6*	0/15
f_{17}	63	1030	4005	30677	56288	80472	15/15
	19(14)	29(5)	24(5)	34(34)	∞	∞*1.0e6*	0/15
f_{18}	621	3972	19561	67569	1.3e5	1.5e5	15/15
	23(4)	37(18)	133(133)	∞	∞	∞*1.0e6*	0/15
f_{19}	1	1	3.4e5	6.2e6	6.7e6	6.7e6	15/15
	2928(795)	∞	∞	∞	∞	∞*1.0e6*	0/15
f_{20}	82	46150	3.1e6	5.5e6	5.6e6	5.6e6	14/15
	83(11)	4.3(1)	0.26(0.2)	0.31(0.3)	0.36(0.4)	0.36(0.4)	6/15
f_{21}	561	6541	14103	14643	15567	17589	15/15
	29(17)	64(78)	36(40)	62(70)	73(96)	81(86)	7/15
f_{22}	467	5580	23491	24948	26847	1.3e5	12/15
	40(28)	397(467)	∞	∞	∞	∞*1.0e6*	0/15
f_{23}	3.2	1614	67457	4.9e5	8.1e5	8.4e5	15/15
	2.2(3)	∞	∞	∞	∞	∞*1.0e6*	0/15
f_{24}	1.3e6	7.5e6	5.2e7	5.2e7	5.2e7	5.2e7	3/15
	∞	∞	∞	∞	∞	∞*1.0e6*	0/15

Table 1: Expected running time (ERT in number of function evaluations) divided by the best ERT measured during BBOB-2009 (given in the respective first row) for different Δf values for functions f_1–f_{24}. The median number of conducted function evaluations is additionally given in *italics*, if $\mathrm{ERT}(10^{-7}) = \infty$. #succ is the number of trials that reached the final target $f_{\mathrm{opt}} + 10^{-8}$.

2009: Noiseless functions definitions. Technical Report RR-6829, INRIA, 2009. Updated February 2010.

[5] V. V. Melo and A. C. B. Delbem. Easy efficiency-enhancement technique for the ecga. In M. M. B. R. Vellasco, M. C. P. de Souto, and J. J. F. Cerqueira, editors, *Brazilian Symposium on Neural Networks (SBRN'08)*, pages 69–74. IEEE Computer Society, 2008.

[6] V. V. Melo and A. C. B. Delbem. Using smart sampling to discover promising regions and increase the efficiency of differential evolution. In *ISDA '09: Proceedings of the 2009 Ninth International Conference on Intelligent Systems Design and Applications*, pages 1394–1399, Washington, DC, USA, 2009. IEEE Computer Society.

[7] K. Mullen, D. Ardia, D. Gil, D. Windover, and J. Cline. DEoptim: An R package for global optimization by differential evolution. *Journal of Statistical Software*, 40(6):1–26, 2009.

[8] F. Neri, G. Iacca, and E. Mininno. Disturbed exploitation compact differential evolution for limited memory optimization problems. *Information Sciences*, 181(12):2469 – 2487, 2011.

[9] Q.-K. Pan, L. Wang, L. Gao, and W. D. Li. An effective hybrid discrete differential evolution algorithm for the flow shop scheduling with intermediate buffers. *Information Sciences*, 181:668–685, February 2011.

[10] R. Storn and K. Price. Differential Evolution- A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces. Technical report, 1995.

[11] Y. Wang, B. Li, and T. Weise. Estimation of distribution and differential evolution cooperation for large scale economic load dispatch optimization of power systems. *Information Sciences*, 180:2405–2420, June 2010.

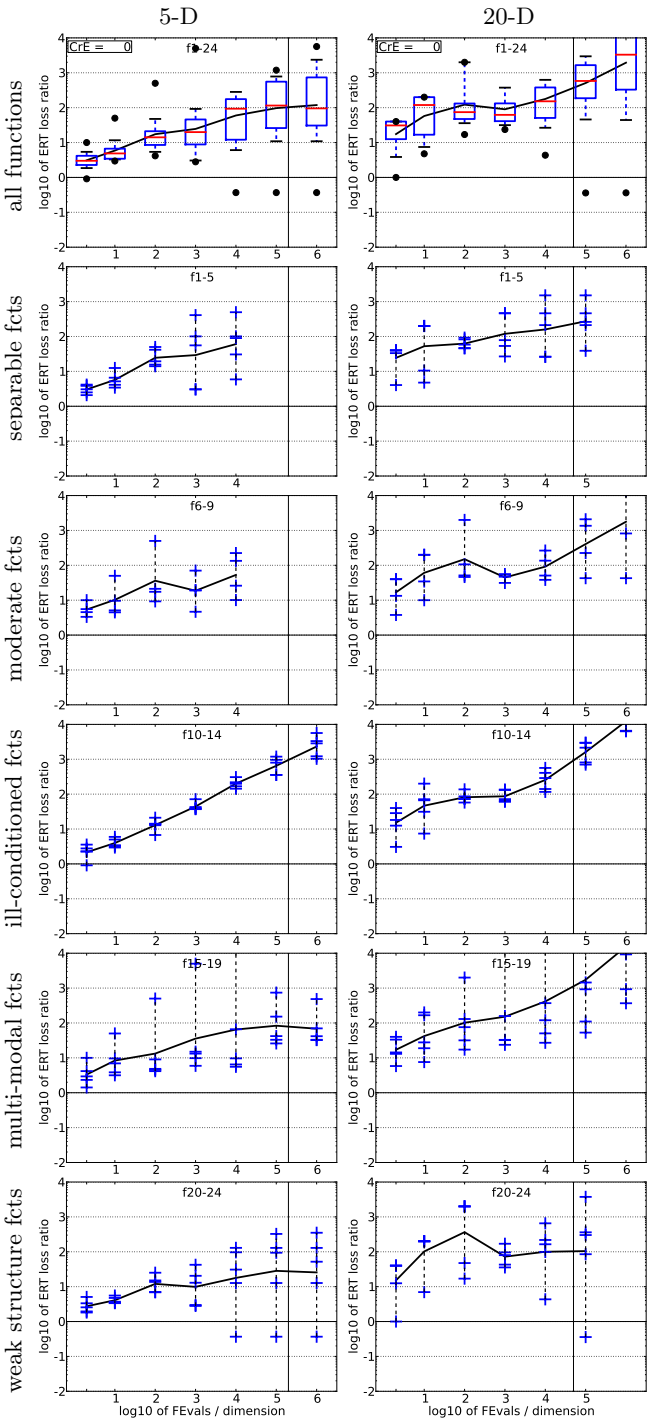

Figure 3: ERT **loss ratio vs. a given budget** FEvals**. Each cross (**+**) represents a single function. The target value** f_t **used for a given** FEvals **is the smallest (best) recorded function value such that** $\mathrm{ERT}(f_t) \leq$ FEvals **for the presented algorithm. Shown is** FEvals **divided by the respective best** $\mathrm{ERT}(f_t)$ **from BBOB-2009 for functions** f_1**–**f_{24} **in 5-D and 20-D. Line: geometric mean. Box-Whisker error bar: 25-75%-ile with median (box), 10-90%-ile (caps), and minimum and maximum** ERT **loss ratio (points). The vertical line gives the maximal number of function evaluations in a single trial in this function subset.**