Black-Box Optimization Benchmarking of Prototype Optimization with Evolved Improvement Steps for Noiseless Function Testbed

Jiří Kubalík
Department of Cybernetics
Czech Technical University in Prague
Technická 2, 166 27 Prague 6
Czech Republic
kubalik@labe.felk.cvut.cz

ABSTRACT

This paper presents benchmarking of a stochastic local search algorithm called Prototype Optimization with Evolved Improvement Steps (POEMS) on the noise-free BBOB 2009 testbed. Experiments for 2, 3, 5, 10 and 20 D were done, where D denotes the search space dimension. The maximum number of function evaluations is chosen as $10^5 \times D$. Experimental results show that POEMS performs best on all separable functions and the attractive sector function. It works also quite well on multi-modal functions with lower dimensions. On the other hand, the algorithm fails to solve functions with high conditioning.

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, Experimentation, Performance

Keywords

Benchmarking, Black-box optimization, Evolutionary computation, Stochastic local search

1. INTRODUCTION

The Prototype Optimization with Evolved Improvement Steps (POEMS) is a stochastic local search algorithm that uses an evolutionary algorithm for searching the neighborhood of the current best solution. The moves in the search space can be thought of as so-called *evolved hypermutations*. The concept of the evolved hypermutations has been shown to outperform other mutation-based evolutionary algorithms that use pure random hypermutations for generating new points in the search space on several combinatorial optimization problems [1, 3, 4]. It has also been implemented for solving real-valued function optimizations [2].

GECCO'09, July 8–12, 2009, Montréal Québec, Canada.

2. POEMS

Prototype Optimization with Evolved iMprovement Steps is an iterative optimization approach that employs an EA for finding the best modification of the current solution, called *prototype*, in each iteration. Modifications are represented as fixed length sequences of primitive actions defined specifically for the problem at hand. Action sequences are assessed based on how well/badly they modify the current prototype, which is an input parameter to the EA. After the EA finishes, it is checked to determine whether the best evolved sequence improves the current prototype or not. If the best action sequence generates a new solution of at least the same quality as the current prototype, then the modified prototype is considered as a new prototype for the next iteration. Otherwise the current prototype remains unchanged. The iterative process stops after a specified number of iterations. An outline of the POEMS algorithm is shown in Figure 1.

The EA employed in POEMS evolves linear chromosomes of length $MaxGenes$, where each gene represents an instance of certain action chosen from a set of elementary actions defined for the given problem. Each gene is represented by a record, with an attribute *action_type* followed by parameters

```
1   i ← 0
2   generate(Prototype^(i))
3   repeat
4     i ← i + 1
5     BestSequence ← run_EA(Prototype^(i−1))
6     Cand ← apply(BestSequence, Prototype^(i−1))
7     if(Cand better_than_or_equal_to Prototype^(i−1))
8         Prototype^(i) ← Cand
9     else
10        Prototype^(i) ← Prototype^(i−1)
11  until(POEMS termination condition)
12  return Prototype^(i)
```

Figure 1: Outline of the POEMS algorithm

```
1   i ← 0
2   InitializeActionSequences(Population^(i))
3   Evaluate(Population^(i))
4   while not TerminationCondition() do
5     i ← i + 1
6     Parents ← Select(Population^(i))
7     if rand() < P_cross
8         Ch ← Crossover(Parents)
9     else
10        Ch ← Mutate(Parents)
11    Evaluate(Ch)
12    R ← FindReplacement(Population^(i−1))
13    Population^(i) ← Replace(Population^(i−1),R, Ch)
14  BestSequence ← BestOf(Population^(i))
15  return BestSequence
```

Figure 2: Iterative evolutionary algorithm used in POEMS

of the action. Besides actions that truly modify the prototype, there is also a special type of action called *nop* (no operation). Any action with $action\_type = nop$ is interpreted as a void action with no effect on the prototype, regardless of the values of its parameters. Chromosomes can contain one or more instances of the *nop* operation. This way a variable effective length of chromosomes is implemented. However, sequences that do not change the prototype at all (e.g., those composed entirely of *nop* actions) are fatally penalized in order to avoid a convergence to useless trivial "modifications".

During the evolutionary algorithm, the evolved action sequences are varied by means of crossover and mutation operators. A generalized uniform crossover introduced in [1] and mutation operator that changes either the *action_type* or action parameters are used in this work. In each generation, the parents are crossed with probability P_{cross}, otherwise they are mutated so that their every action is changed with probability P_{mutate}. New action sequences resulting from crossover and mutation operator are further adjusted to contain at least one active action. Thus, if the original newly generated sequence consists of all *nop* actions then one of its actions (chosen at random) is changed to an active action.

It is obvious that once the prototype solution is already of a good quality the randomly initialized action sequences will very likely only worsen it. Moreover, action sequences with smaller number of active actions will worsen the prototype less than action sequences with greater number of active actions on average. In order to prevent the evolutionary algorithm from converging towards action sequences with a minimal number of active actions *niching tournament selection* and *niching replacement strategy* are proposed.

A population of size $PopSize$ is split into $MaxGenes$ niches of equal size $PopSize/MaxGenes$ so that each niche $niche_i$ can contain only sequences with the number of active actions greater than or equal to i.

The niching tournament selection first selects by random a niche i and then runs the tournament between n randomly selected action sequences within the niche. Thus, only action sequences with at least i active actions compete for being selected as a parent.

The niching replacement strategy works so that each newly generated action sequence is inserted into the population only if an admissible replacement is found in the current population by function $FindReplacement()$ (step 12 in Figure 2). An action sequence AS_{old} can be replaced by the new one AS_{new} with i active actions iff $fitness(AS_{new})$ is better than or equal to $fitness(AS_{old})$ and AS_{old} is in $niche_j$ such that $j \leq i$. This way both the quality and diversity of action sequences in the population are ensured.

3. TAILORING POEMS TO REAL-VALUED OPTIMIZATION

The prototype solution is represented as a vector of D real valued variables. In all experiments the variables are limited to take values from the interval $\langle lbound, ubound \rangle$, where $lbound$ and $ubound$ are lower and upper bounds of the variable domains.

Prototype initialization. POEMS is an iterative algorithm and as such its performance strongly depends on the initial prototype from which the iterative optimization starts. However, no initialization heuristic was used in this work so the prototype solution was simply sampled uniformly in $[lbound, ubound]^D$.

Actions. Only one active action $changeVariable(i, value)$ was proposed in this work. This action changes the value of the current prototype's variable i by adding the value $value$. The parameter $value$ can be positive or negative number sampled from the normal distribution $N(0, \sigma_i^2)$. Moreover, the $value$ is always chosen so that the constraint

$$lbound \leq prototype[i] + value \leq ubound$$

is satisfied.

The parameters σ_i^2 are initialized to

$$\sigma_i^2 = 0.25 * (ubound - lbound)$$

at the beginning of the POEMS run.

During the course of the run the values of σ_i^2 are adapted between iteration $k-1$ and iteration k according to the following rule

$$\sigma_{i,k}^2 = \sigma_{i,k-1}^2 * (1 - \alpha) + \delta_i * \alpha,$$

where

$$\delta_i = prototype[i]^{(k)} - prototype[i]^{(k-1)}$$

and α is a weighting factor that takes values from the interval $(0, 1)$. Thus, if the prototype's variable i does not change from iteration $k-1$ to iteration k then the corresponding $\sigma_{i,k}^2$ decreases to maximally possible extent. In the opposite case, the $\sigma_{i,k}^2$ is decreased less or it can even increase.

This can be interpreted so that if for the given value of σ_i^2 an improving action sequence that includes a modification of the variable i has been found then there is perhaps no need for decreasing a value of σ_i^2. On the contrary, an absence of an action modifying the variable i in the improving action sequence or if no improving action sequence has been found in the current iteration can indicate that the interval determined by σ_i^2 is too wide. Thus, the search should focus to a closer neighborhood of the current prototype's value of the variable i.

Operators. A generalized uniform crossover as introduced in [1] is used in this work. It works so that given two parental action sequences, the offspring is created by randomly picking the parental actions (from both parents) without replacement.

The mutation operator changes either the type of the action ($changeVariable(i, value) \leftrightarrow nop(i, value)$) or its parameter *value*. When the parameter *value* is to be changed, its new value is sampled uniformly from the interval $\langle 0.5 * value, 2.0 * value \rangle$. Only values for which the $prototype[i] + value$ does not fall below the *lbound* or under the *ubound* are accepted.

Restarted strategy. The algorithm stops either when the maximum number of $10^5 \times D$ function evaluations has been exceeded or when a solution of a quality equal to or better than the target function value $f_{target} = f_{opt} + 10^{-8}$ has been found. Additionally, if the values of σ_i^2 for $i = 1 \dots D$ fall below 10^{-11} then they are reinitialized to the original values $0.25 * (ubound - lbound)$ while the current prototype remains unchanged.

4. EXPERIMENTAL SETUP

No tuning of POEMS control parameters was done. The configuration was parameterized solely by the dimension of the problem at hand. The parameter setting was identical for all functions so the *crafting effort* is zero. The POEMS algorithm was configured as follows:

- $MaxGenes = D$, $NicheSize = 20$,
- $PopSize = MaxGenes * NicheSize$,
- $P_{cross} = 0.75$, $P_{mutate} = 0.5$, $\alpha = 0.2$,
- Tournament selection with $n = 2$,
- $lbound = -5.0$, $ubound = 5.0$,
- Number of fitness evaluations calculated in each iteration: $10 * PopSize$,
- Maximal number of fitness evaluations: $D \times 10^5$.

The simulations for 2, 3, 5, 10 and 20 D were done with a maximum of $10^5 \times D$ function evaluations with the C-code and they took altogether 6 hours and a half.

5. CPU TIMING EXPERIMENTS

For the timing experiment the POEMS algorithm was run with a maximum of $10^5 \times D$ function evaluations and restarted until 30 seconds has passed. The experiments have been conducted with an Intel Pentium-M 1400 MHz under MS Windows using the C-code provided. The time per function evaluation was 2.4, 2.7, 4.0, 8.5, 13×10^{-6} seconds in dimensions 2, 3, 5, 10, 20 respectively.

6. RESULTS

Results from experiments according to [6] on the benchmark functions given in [5, 7] are presented in Figure 3, Figure 4 and in Table 1. The POEMS algorithm performs best on all separable functions and the attractive sector function. These functions are solved for all considered dimensions $D = 2, 3, 5, 10, 20$. The algorithm works quite well on multi-modal functions with lower dimensions. On the other hand, it fails to solve the group of functions with high conditioning; for the Bent cigar and the Sharp ridge the target solution was not found not even for $D = 2$.

7. CONCLUSIONS

In our opinion, the rather low overall performance of the POEMS algorithm on the multi-modal functions with high dimensions and functions with high conditioning is caused by the following two factors:

1. POEMS has weak ability to identify and utilize dependencies among variables.
2. POEMS has weak adaptation mechanism that would determine the proper size of the neighborhood of the current solution prototype to be searched by the EA in the next iteration.

Another observation is that POEMS is not very time efficient. Even for the simplest problems like *Linear slope problem* it takes several hundreds to thousands fitness evaluations to reach a solution of the desired quality. This follows from the nature of the algorithm since each single step from prototype at time t to prototype at time $t + 1$ involves running an evolutionary algorithm.

Acknowledgments

The author would like to acknowledge the great and hard work of the BBOB team. Research described in the paper has been supported by the research program No. MSM 6840770038 "Decision Making and Control for Manufacturing III" of the CTU in Prague.

8. REFERENCES

[1] J. Kubalik and J. Faigl. Iterative Prototype Optimisation with Evolved Improvement Steps. In: P. Collet, M. Tomassini, M. Ebner, A. Ekart and S. Gustafson (Eds.): Proceedings of the 9th European Conference on Genetic Programming, EuroGP 2006, Heidelberg: Springer, pp. 154–165, 2006.

[2] J. Kubalik. Real-Parameter Optimization by Iterative Prototype Optimization with Evolved Improvement Steps. In: 2006 IEEE Congress on Evolutionary Computation [CD-ROM]. Los Alamitos: IEEE Computer Society, pp. 6823–6829, 2006.

[3] J. Kubalik Solving the Sorting Network Problem Using Iterative Optimization with Evolved Hypermutations. Accepted for presentation at GECCO 2009, July 8–12, Montréal Québec, Canada, 2009.

[4] J. Kubalik. Solving Multiple Sequence Alignment Problem Using Prototype Optimization with Evolved Improvement Steps. Accepted for presentation at the ICANNGA 2009, Kuopio, Finland, 23-25, April, 2009.

[5] 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.

[6] N. Hansen, A. Auger, S. Finck, and R. Ros. Real-parameter black-box optimization benchmarking 2009: Experimental setup. Technical Report RR-6828, INRIA, 2009.

[7] 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.

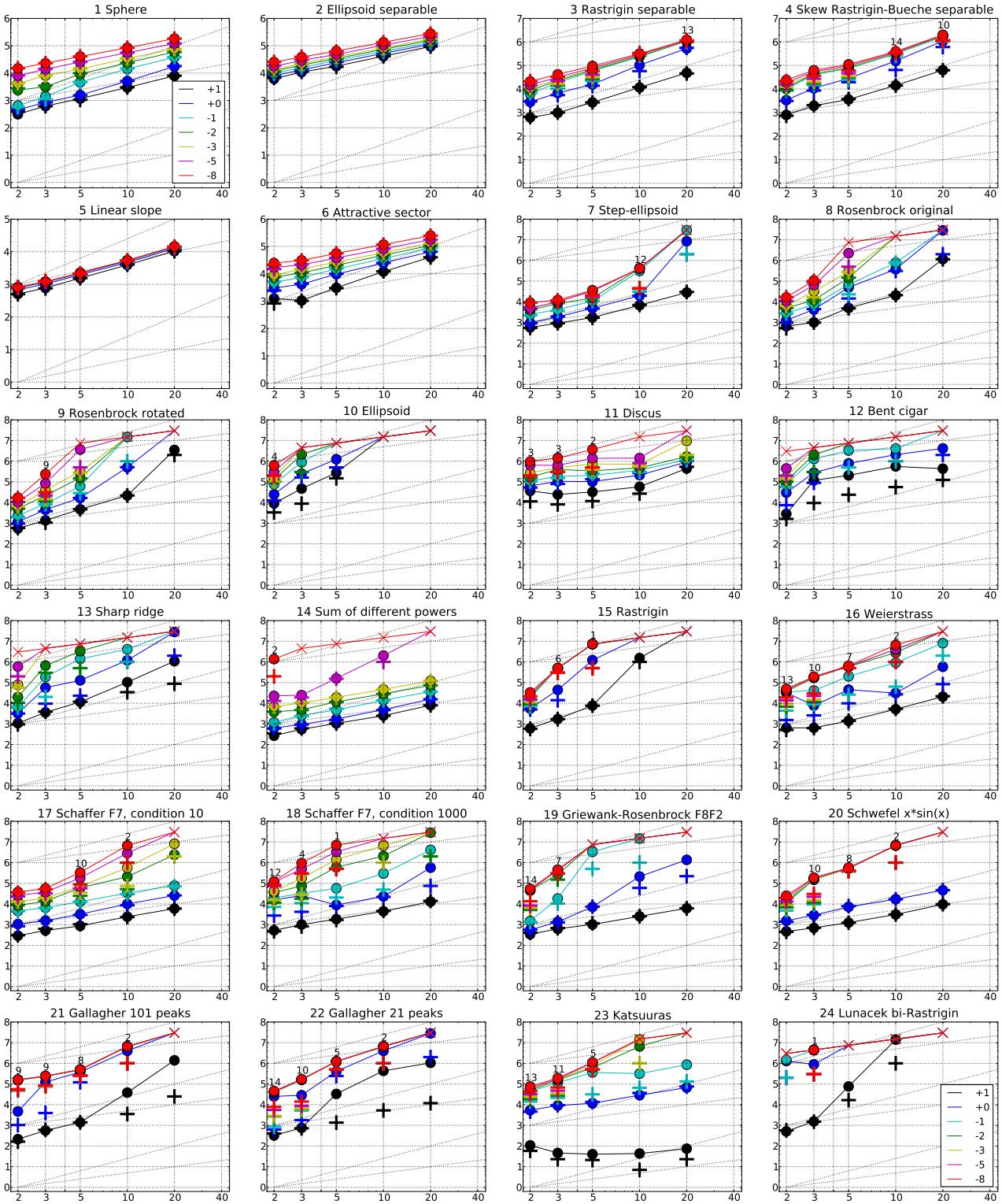

Figure 3: Expected Running Time (ERT, ●) **to reach** $f_{\text{opt}} + \Delta f$ **and median number of function evaluations of successful trials** (+), **shown for** $\Delta f = 10, 1, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-5}, 10^{-8}$ **(the exponent is given in the legend of** f_1 **and** f_{24}**) versus dimension in log-log presentation. The** $\text{ERT}(\Delta f)$ **equals to** $\#\text{FEs}(\Delta f)$ **divided by the number of successful trials, where a trial is successful if** $f_{\text{opt}} + \Delta f$ **was surpassed during the trial. The** $\#\text{FEs}(\Delta f)$ **are the total number of function evaluations while** $f_{\text{opt}} + \Delta f$ **was not surpassed during the trial from all respective trials (successful and unsuccessful), and** f_{opt} **denotes the optimal function value. Crosses (**×**) indicate the total number of function evaluations** $\#\text{FEs}(-\infty)$**. Numbers above ERT-symbols indicate the number of successful trials. Annotated numbers on the ordinate are decimal logarithms. Additional grid lines show linear and quadratic scaling.**

Δf	#	ERT	10%	90%	RT_{succ}
f_1 in 5-D, N=15, mFE=42760					
10	15	1.2e3	1.1e3	1.2e3	1.2e3
1	15	1.6e3	1.5e3	1.6e3	1.6e3
1e−1	15	4.6e3	4.2e3	5.0e3	4.6e3
1e−3	15	1.4e4	1.4e4	1.5e4	1.4e4
1e−5	15	2.5e4	2.5e4	2.6e4	2.5e4
1e−8	15	4.0e4	4.0e4	4.1e4	4.0e4
f_1 in 20-D, N=15, mFE=185686					
10	15	7.9e3	7.7e3	8.1e3	7.9e3
1	15	1.7e4	1.7e4	1.8e4	1.7e4
1e−1	15	3.8e4	3.6e4	3.9e4	3.8e4
1e−3	15	7.7e4	7.6e4	7.8e4	7.7e4
1e−5	15	1.2e5	1.2e5	1.2e5	1.2e5
1e−8	15	1.8e5	1.8e5	1.8e5	1.8e5
f_2 in 5-D, N=15, mFE=70387					
10	15	1.8e4	1.6e4	1.9e4	1.8e4
1	15	2.4e4	2.3e4	2.5e4	2.4e4
1e−1	15	2.9e4	2.8e4	3.1e4	2.9e4
1e−3	15	3.9e4	3.8e4	4.1e4	3.9e4
1e−5	15	4.8e4	4.6e4	5.0e4	4.8e4
1e−8	15	6.4e4	6.2e4	6.5e4	6.4e4
f_2 in 20-D, N=15, mFE=292399					
10	15	9.4e4	9.2e4	9.7e4	9.4e4
1	15	1.1e5	1.1e5	1.2e5	1.1e5
1e−1	15	1.3e5	1.3e5	1.3e5	1.3e5
1e−3	15	1.8e5	1.7e5	1.8e5	1.8e5
1e−5	15	2.2e5	2.2e5	2.2e5	2.2e5
1e−8	15	2.8e5	2.8e5	2.8e5	2.8e5
f_3 in 5-D, N=15, mFE=294785					
10	15	2.7e3	2.5e3	3.0e3	2.7e3
1	15	1.6e4	1.3e4	1.8e4	1.6e4
1e−1	15	5.7e4	3.5e4	8.1e4	5.7e4
1e−3	15	6.9e4	4.6e4	9.3e4	6.9e4
1e−5	15	7.8e4	5.5e4	1.0e5	7.8e4
1e−8	15	9.5e4	7.2e4	1.2e5	9.5e4
f_3 in 20-D, N=15, mFE=2.00e6					
10	15	4.9e4	4.6e4	5.2e4	4.9e4
1	15	5.3e5	4.6e5	5.9e5	5.3e5
1e−1	13	1.1e6	8.3e5	1.4e6	8.8e5
1e−3	13	1.1e6	8.7e5	1.4e6	9.2e5
1e−5	13	1.2e6	9.3e5	1.5e6	9.7e5
1e−8	13	1.2e6	9.8e5	1.5e6	1.0e6
f_4 in 5-D, N=15, mFE=292807					
10	15	3.6e3	3.2e3	4.0e3	3.6e3
1	15	2.8e4	2.0e4	3.7e4	2.8e4
1e−1	15	7.5e4	5.3e4	9.8e4	7.5e4
1e−3	15	8.5e4	6.2e4	1.1e5	8.5e4
1e−5	15	9.8e4	7.4e4	1.2e5	9.8e4
1e−8	15	1.1e5	8.9e4	1.4e5	1.1e5
f_4 in 20-D, N=15, mFE=2.00e6					
10	15	6.3e4	6.0e4	6.7e4	6.3e4
1	13	1.1e6	8.1e5	1.4e6	9.2e5
1e−1	11	1.7e6	1.3e6	2.2e6	1.3e6
1e−3	10	1.9e6	1.4e6	2.6e6	1.2e6
1e−5	10	1.9e6	1.5e6	2.7e6	1.2e6
1e−8	10	2.0e6	1.5e6	2.8e6	1.3e6
f_5 in 5-D, N=15, mFE=3087					
10	15	1.5e3	1.5e3	1.6e3	1.5e3
1	15	2.0e3	1.9e3	2.1e3	2.0e3
1e−1	15	2.1e3	2.0e3	2.2e3	2.1e3
1e−3	15	2.2e3	2.1e3	2.3e3	2.2e3
1e−5	15	2.2e3	2.1e3	2.3e3	2.2e3
1e−8	15	2.2e3	2.1e3	2.3e3	2.2e3
f_5 in 20-D, N=15, mFE=17260					
10	15	1.0e4	1.0e4	1.1e4	1.0e4
1	15	1.3e4	1.2e4	1.3e4	1.3e4
1e−1	15	1.4e4	1.3e4	1.4e4	1.4e4
1e−3	15	1.4e4	1.4e4	1.5e4	1.4e4
1e−5	15	1.4e4	1.4e4	1.5e4	1.4e4
1e−8	15	1.4e4	1.4e4	1.5e4	1.4e4
f_6 in 5-D, N=15, mFE=59096					
10	15	3.1e3	2.7e3	3.4e3	3.1e3
1	15	9.7e3	9.3e3	1.0e4	9.7e3
1e−1	15	1.5e4	1.4e4	1.5e4	1.5e4
1e−3	15	2.7e4	2.6e4	2.7e4	2.7e4
1e−5	15	3.8e4	3.8e4	3.8e4	3.8e4
1e−8	15	5.6e4	5.5e4	5.6e4	5.6e4
f_6 in 20-D, N=15, mFE=260687					
10	15	4.1e4	4.0e4	4.2e4	4.1e4
1	15	6.4e4	6.3e4	6.5e4	6.4e4
1e−1	15	8.8e4	8.6e4	9.0e4	8.8e4
1e−3	15	1.3e5	1.3e5	1.3e5	1.3e5
1e−5	15	1.8e5	1.8e5	1.8e5	1.8e5
1e−8	15	2.5e5	2.5e5	2.5e5	2.5e5
f_7 in 5-D, N=15, mFE=142676					
10	15	1.7e3	1.6e3	1.8e3	1.7e3
1	15	4.9e3	4.5e3	5.3e3	4.9e3
1e−1	15	1.1e4	1.0e4	1.1e4	1.1e4
1e−3	15	3.3e4	1.7e4	4.9e4	3.3e4
1e−5	15	3.3e4	1.7e4	4.9e4	3.3e4
1e−8	15	3.6e4	2.1e4	5.2e4	3.6e4
f_7 in 20-D, N=15, mFE=2.00e6					
10	15	2.8e4	2.6e4	3.0e4	2.8e4
1	3	8.5e6	5.1e6	2.6e7	2.0e6
1e−1	1	2.9e7	1.4e7	>3e7	2.0e6
1e−3	0	*12e−1*	*31e−2*	*22e−1*	1.0e6
1e−5	.	.	.	.	.
1e−8	.	.	.	.	.
f_8 in 5-D, N=15, mFE=500001					
10	15	5.0e3	4.5e3	5.6e3	5.0e3
1	14	4.9e4	1.4e4	9.0e4	4.8e4
1e−1	14	6.7e4	2.8e4	1.1e5	6.5e4
1e−3	14	2.9e5	2.3e5	3.5e5	2.6e5
1e−5	3	2.2e6	1.3e6	6.8e6	4.7e5
1e−8	0	*14e−5*	*18e−7*	*51e−5*	4.5e5
f_8 in 20-D, N=15, mFE=2.00e6					
10	15	1.2e6	1.0e6	1.3e6	1.2e6
1	1	2.9e7	1.4e7	>3e7	2.0e6
1e−1	0	*74e−1*	*18e−1*	*87e−1*	2.0e6
1e−3	.	.	.	.	.
1e−5	.	.	.	.	.
1e−8	.	.	.	.	.
f_9 in 5-D, N=15, mFE=500001					
10	15	4.7e3	4.2e3	5.2e3	4.7e3
1	15	1.7e4	1.6e4	1.8e4	1.7e4
1e−1	15	6.2e4	4.5e4	7.9e4	6.2e4
1e−3	14	3.2e5	2.7e5	3.7e5	3.0e5
1e−5	2	3.6e6	1.8e6	>7e6	4.5e5
1e−8	0	*39e−6*	*18e−7*	*25e−5*	4.5e5
f_9 in 20-D, N=15, mFE=2.00e6					
10	8	3.5e6	2.7e6	4.9e6	1.9e6
1	0	*99e−1*	*88e−1*	*15e+0*	2.0e6
1e−1	.	.	.	.	.
1e−3	.	.	.	.	.
1e−5	.	.	.	.	.
1e−8	.	.	.	.	.
f_{10} in 5-D, N=15, mFE=500001					
10	12	2.8e5	1.9e5	3.8e5	2.1e5
1	5	1.2e6	8.0e5	2.3e6	3.4e5
1e−1	0	*45e−1*	*52e−2*	*11e+0*	4.5e5
1e−3	.	.	.	.	.
1e−5	.	.	.	.	.
1e−8	.	.	.	.	.
f_{10} in 20-D, N=15, mFE=2.00e6					
10	0	*12e+2*	*86e+1*	*22e+2*	2.0e6
1	.	.	.	.	.
1e−1	.	.	.	.	.
1e−3	.	.	.	.	.
1e−5	.	.	.	.	.
1e−8	.	.	.	.	.
f_{11} in 5-D, N=15, mFE=500001					
10	15	3.2e4	1.7e4	4.8e4	3.2e4
1	15	1.0e5	7.9e4	1.3e5	1.0e5
1e−1	15	2.0e5	1.7e5	2.4e5	2.0e5
1e−3	8	7.4e5	5.5e5	1.1e6	3.9e5
1e−5	5	1.4e6	9.4e5	2.4e6	4.6e5
1e−8	2	3.7e6	1.9e6	>7e6	5.0e5
f_{11} in 20-D, N=15, mFE=2.00e6					
10	15	4.4e5	3.8e5	5.0e5	4.4e5
1	15	9.1e5	8.4e5	9.7e5	9.1e5
1e−1	15	1.2e6	1.1e6	1.3e6	1.2e6
1e−3	3	9.6e6	5.7e6	2.9e7	1.9e6
1e−5	0	*34e−4*	*41e−5*	*68e−4*	1.8e6
1e−8	.	.	.	.	.
f_{12} in 5-D, N=15, mFE=500001					
10	11	2.1e5	1.2e5	3.0e5	2.0e5
1	6	7.8e5	5.3e5	1.3e6	3.4e5
1e−1	2	3.3e6	1.8e6	>7e6	5.0e5
1e−3	0	*17e−1*	*21e−3*	*24e+0*	4.5e5
1e−5	.	.	.	.	.
1e−8	.	.	.	.	.
f_{12} in 20-D, N=15, mFE=2.00e6					
10	13	4.3e5	1.7e5	6.8e5	4.1e5
1	5	4.1e6	2.4e6	8.1e6	5.1e5
1e−1	0	*37e−1*	*22e−2*	*17e+0*	1.8e6
1e−3	.	.	.	.	.
1e−5	.	.	.	.	.
1e−8	.	.	.	.	.
f_{13} in 5-D, N=15, mFE=500001					
10	15	1.2e4	1.1e4	1.2e4	1.2e4
1	13	1.3e5	5.7e4	2.1e5	9.5e4
1e−1	4	1.4e6	9.4e5	2.8e6	5.0e5
1e−3	0	*22e−2*	*60e−4*	*10e−1*	1.1e5
1e−5	.	.	.	.	.
1e−8	.	.	.	.	.
f_{13} in 20-D, N=15, mFE=2.00e6					
10	10	1.1e6	5.9e5	1.7e6	6.5e5
1	1	2.8e7	1.3e7	>3e7	1.3e5
1e−1	0	*59e−1*	*17e−1*	*13e+0*	4.5e5
1e−3	.	.	.	.	.
1e−5	.	.	.	.	.
1e−8	.	.	.	.	.
f_{14} in 5-D, N=15, mFE=500001					
10	15	1.1e3	1.0e3	1.1e3	1.1e3
1	15	1.7e3	1.6e3	1.9e3	1.7e3
1e−1	15	4.7e3	4.1e3	5.2e3	4.7e3
1e−3	15	1.9e4	1.9e4	2.0e4	1.9e4
1e−5	14	1.6e5	1.1e5	2.1e5	1.6e5
1e−8	0	*31e−7*	*51e−8*	*72e−7*	4.5e5
f_{14} in 20-D, N=15, mFE=2.00e6					
10	15	8.0e3	7.7e3	8.4e3	8.0e3
1	15	1.6e4	1.5e4	1.6e4	1.6e4
1e−1	15	3.7e4	3.6e4	3.8e4	3.7e4
1e−3	15	1.2e5	1.2e5	1.2e5	1.2e5
1e−5	0	*94e−6*	*56e−6*	*12e−5*	1.8e6
1e−8	.	.	.	.	.
f_{15} in 5-D, N=15, mFE=500001					
10	15	7.7e3	7.0e3	8.5e3	7.7e3
1	5	1.2e6	8.0e5	2.3e6	3.8e5
1e−1	1	7.3e6	3.5e6	>7e6	5.0e5
1e−3	1	7.3e6	3.5e6	>7e6	5.0e5
1e−5	1	7.3e6	3.6e6	>7e6	5.0e5
1e−8	1	7.3e6	3.5e6	>7e6	5.0e5
f_{15} in 20-D, N=15, mFE=2.00e6					
10	0	*34e+0*	*22e+0*	*47e+0*	2.5e5
1	.	.	.	.	.
1e−1	.	.	.	.	.
1e−3	.	.	.	.	.
1e−5	.	.	.	.	.
1e−8	.	.	.	.	.
f_{16} in 5-D, N=15, mFE=499983					
10	15	1.4e3	1.3e3	1.5e3	1.4e3
1	14	4.5e4	9.7e3	8.7e4	4.5e4
1e−1	11	2.0e5	1.0e5	3.4e5	1.1e5
1e−3	7	6.0e5	4.0e5	9.3e5	3.0e5
1e−5	7	6.1e5	4.1e5	9.5e5	3.0e5
1e−8	7	6.6e5	4.3e5	1.1e6	3.1e5
f_{16} in 20-D, N=15, mFE=2.00e6					
10	15	2.1e4	1.9e4	2.2e4	2.1e4
1	12	5.8e5	2.5e5	9.8e5	4.0e5
1e−1	3	8.1e6	4.9e6	2.4e7	2.0e6
1e−3	0	*58e−2*	*25e−3*	*12e−1*	1.8e6
1e−5	.	.	.	.	.
1e−8	.	.	.	.	.
f_{17} in 5-D, N=15, mFE=500001					
10	15	8.7e2	7.0e2	1.0e3	8.7e2
1	15	3.2e3	2.9e3	3.5e3	3.2e3
1e−1	15	1.3e4	1.2e4	1.4e4	1.3e4
1e−3	14	7.1e4	3.6e4	1.1e5	6.8e4
1e−5	12	1.8e5	1.2e5	2.6e5	1.7e5
1e−8	10	3.4e5	2.3e5	4.9e5	2.1e5
f_{17} in 20-D, N=15, mFE=2.00e6					
10	15	5.9e3	5.7e3	6.2e3	5.9e3
1	15	2.6e4	2.5e4	2.6e4	2.6e4
1e−1	15	7.7e4	7.3e4	8.2e4	7.7e4
1e−3	3	8.2e6	4.9e6	2.5e7	2.0e6
1e−5	0	*11e−3*	*55e−5*	*46e−3*	4.5e5
1e−8	.	.	.	.	.
f_{18} in 5-D, N=15, mFE=500001					
10	15	1.8e3	1.7e3	2.0e3	1.8e3
1	15	8.9e3	8.0e3	9.8e3	8.9e3
1e−1	14	5.7e4	2.2e4	9.9e4	5.6e4
1e−3	4	1.4e6	9.4e5	2.8e6	5.0e5
1e−5	2	3.3e6	1.8e6	>7e6	5.0e5
1e−8	1	7.1e6	3.3e6	>7e6	5.0e5
f_{18} in 20-D, N=15, mFE=2.00e6					
10	15	1.3e4	1.2e4	1.4e4	1.3e4
1	12	5.7e5	2.6e5	8.9e5	5.5e5
1e−1	5	4.1e6	2.7e6	7.5e6	1.6e6
1e−3	0	*38e−2*	*35e−3*	*14e−1*	4.5e5
1e−5	.	.	.	.	.
1e−8	.	.	.	.	.
f_{19} in 5-D, N=15, mFE=500001					
10	15	1.0e3	8.8e2	1.1e3	1.0e3
1	15	7.2e3	6.1e3	8.3e3	7.2e3
1e−1	2	3.3e6	1.8e6	>7e6	5.0e5
1e−3	0	*18e−2*	*99e−3*	*30e−2*	4.5e5
1e−5	.	.	.	.	.
1e−8	.	.	.	.	.
f_{19} in 20-D, N=15, mFE=2.00e6					
10	15	6.2e3	6.0e3	6.4e3	6.2e3
1	10	1.4e6	8.7e5	2.0e6	8.6e5
1e−1	0	*94e−2*	*71e−2*	*12e−1*	1.8e6
1e−3	.	.	.	.	.
1e−5	.	.	.	.	.
1e−8	.	.	.	.	.
f_{20} in 5-D, N=15, mFE=500001					
10	15	1.3e3	1.2e3	1.3e3	1.3e3
1	15	7.2e3	6.4e3	8.1e3	7.2e3
1e−1	8	5.3e5	3.7e5	7.8e5	3.3e5
1e−3	8	5.4e5	3.8e5	7.9e5	3.4e5
1e−5	8	5.5e5	3.9e5	8.0e5	3.4e5
1e−8	8	5.6e5	4.0e5	8.3e5	3.5e5
f_{20} in 20-D, N=15, mFE=2.00e6					
10	15	9.6e3	9.1e3	1.0e4	9.6e3
1	15	4.6e4	4.4e4	4.8e4	4.6e4
1e−1	5	1.2e6	7.8e5	2.3e6	3.5e5
1e−3	5	1.2e6	7.9e5	2.3e6	3.5e5
1e−5	5	1.2e6	7.8e5	2.3e6	3.5e5
1e−8	5	1.2e6	7.9e5	2.3e6	3.6e5
f_{21} in 5-D, N=15, mFE=500001					
10	15	1.4e3	1.3e3	1.5e3	1.4e3
1	9	3.8e5	2.3e5	5.8e5	2.2e5
1e−1	8	4.9e5	3.1e5	7.5e5	2.7e5
1e−3	8	4.9e5	3.2e5	7.7e5	2.7e5
1e−5	8	4.9e5	3.2e5	7.6e5	2.7e5
1e−8	8	5.0e5	3.3e5	8.0e5	2.8e5
f_{21} in 20-D, N=15, mFE=2.00e6					
10	9	1.4e6	9.6e5	1.8e6	1.3e6
1	0	*67e−1*	*20e−1*	*32e+0*	2.0e5
1e−1	.	.	.	.	.
1e−3	.	.	.	.	.
1e−5	.	.	.	.	.
1e−8	.	.	.	.	.
f_{22} in 5-D, N=15, mFE=500001					
10	15	3.3e4	1.5e3	6.5e4	3.3e4
1	10	4.1e5	2.7e5	6.2e5	2.4e5
1e−1	5	1.2e6	7.8e5	2.3e6	3.5e5
1e−3	5	1.2e6	7.9e5	2.3e6	3.5e5
1e−5	5	1.2e6	7.8e5	2.3e6	3.5e5
1e−8	5	1.2e6	7.9e5	2.3e6	3.6e5
f_{22} in 20-D, N=15, mFE=2.00e6					
10	10	1.1e6	5.5e5	1.8e6	4.6e5
1	1	2.8e7	1.3e7	>3e7	2.0e6
1e−1	0	*51e−1*	*20e−1*	*43e+0*	2.0e5
1e−3	.	.	.	.	.
1e−5	.	.	.	.	.
1e−8	.	.	.	.	.
f_{23} in 5-D, N=15, mFE=500001					
10	15	4.0e1	2.4e1	5.8e1	4.0e1
1	15	1.2e4	9.7e3	1.4e4	1.2e4
1e−1	9	3.6e5	2.2e5	5.5e5	2.4e5
1e−3	6	8.0e5	5.0e5	1.4e6	2.8e5
1e−5	5	1.1e6	6.7e5	2.1e6	3.3e5
1e−8	5	1.1e6	6.9e5	2.1e6	3.4e5
f_{23} in 20-D, N=15, mFE=2.00e6					
10	15	7.4e1	3.6e1	1.2e2	7.4e1
1	15	6.7e4	5.8e4	7.6e4	6.7e4
1e−1	11	8.6e5	4.8e5	1.3e6	6.5e5
1e−3	0	*59e−3*	*35e−3*	*16e−2*	1.4e6
1e−5	.	.	.	.	.
1e−8	.	.	.	.	.
f_{24} in 5-D, N=15, mFE=500001					
10	14	7.6e4	3.2e4	1.3e5	7.5e4
1	0	*70e−1*	*24e−1*	*87e−1*	7.1e4
1e−1	.	.	.	.	.
1e−3	.	.	.	.	.
1e−5	.	.	.	.	.
1e−8	.	.	.	.	.
f_{24} in 20-D, N=15, mFE=2.00e6					
10	0	*46e+0*	*29e+0*	*71e+0*	2.8e5
1	.	.	.	.	.
1e−1	.	.	.	.	.
1e−3	.	.	.	.	.
1e−5	.	.	.	.	.
1e−8	.	.	.	.	.

Table 1: Shown are, for a given target difference to the optimal function value Δf: the number of successful trials (#); the expected running time to surpass $f_{\text{opt}}+\Delta f$ (ERT, see Figure 3); the 10%-tile and 90%-tile of the bootstrap distribution of ERT; the average number of function evaluations in successful trials or, if none was successful, as last entry the median number of function evaluations to reach the best function value (RT_{succ}). If $f_{\text{opt}}+\Delta f$ was never reached, figures in *italics* denote the best achieved Δf-value of the median trial and the 10% and 90%-tile trial. Furthermore, N denotes the number of trials, and mFE denotes the maximum of number of function evaluations executed in one trial. See Figure 3 for the names of functions.

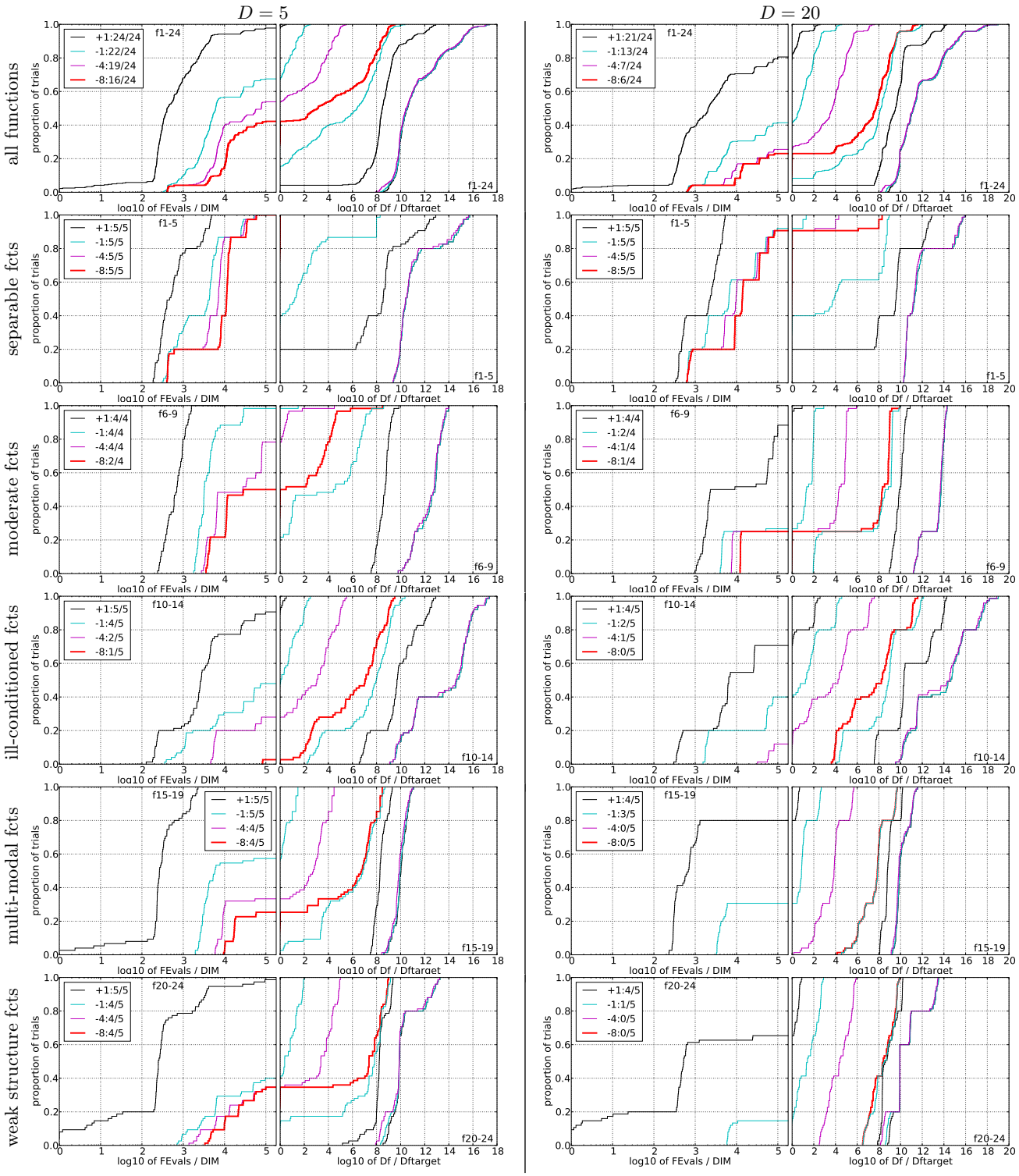

Figure 4: Empirical cumulative distribution functions (ECDFs), plotting the fraction of trials versus running time (left) or Δf. Left subplots: ECDF of the running time (number of function evaluations), divided by search space dimension D, to fall below $f_{\mathrm{opt}} + \Delta f$ with $\Delta f = 10^k$, where k is the first value in the legend. Right subplots: ECDF of the best achieved Δf divided by 10^k (upper left lines in continuation of the left subplot), and 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). Top row: all results from all functions; second row: separable functions; third row: misc. moderate functions; fourth row: ill-conditioned functions; fifth row: multi-modal functions with adequate structure; last row: multi-modal functions with weak structure. The legends indicate the number of functions that were solved in at least one trial. FEvals denotes number of function evaluations, D and DIM denote search space dimension, and Δf and Df denote the difference to the optimal function value.