Comparing Results of 31 Algorithms from the Black-Box Optimization Benchmarking BBOB-2009

Nikolaus Hansen
INRIA Saclay, TAO team
LRI, Bat 490 Univ. Paris-Sud
91405 Orsay Cedex France

Anne Auger
INRIA Saclay, TAO team
LRI, Bat 490 Univ. Paris-Sud
91405 Orsay Cedex France

Raymond Ros
INRIA Saclay, TAO team
LRI, Bat 490 Univ. Paris-Sud
91405 Orsay Cedex France

Steffen Finck
Research Center PPE, Univ.
of Applied Science Vorarlberg
Hochschulstrasse 1
6850 Dornbirn, Austria

Petr Pošík
Czech Technical University in
Prague, Dept. of Cybernetics
Technická 2
166 27 Prague 6, Czech Rep.

ABSTRACT

This paper presents results of the BBOB-2009 benchmarking of 31 search algorithms on 24 noiseless functions in a black-box optimization scenario in continuous domain. The runtime of the algorithms, measured in number of function evaluations, is investigated and a connection between a single convergence graph and the runtime distribution is uncovered. Performance is investigated for different dimensions up to 40-D, for different target precision values, and in different subgroups of functions. Searching in larger dimension and multi-modal functions appears to be more difficult. The choice of the best algorithm also depends remarkably on the available budget of function evaluations.

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, performance

Keywords

Benchmarking, black-box optimization

1. INTRODUCTION AND METHODS

This paper presents running time results from BBOB-2009—the *Black-Box Optimization Benchmarking* workshop at the *Genetic and Evolutionary Computation Conference (GECCO)* 2009. 31 real-parameter optimization algorithms (see Appendix) have been tested in a black-box scenario on 24 noiseless benchmark functions. The experimental procedure is detailed in [16], the functions are presented in [10, 17]. Performance results for each algorithm on each function can be found in the original publications. Tables with results of all algorithms on each single function are available at `http://coco.gforge.inria.fr/doku.php?id=bbob-2009-results`. In the following, we present summarizing results and results on function groups.

The performance measure adopted in this paper is the **runtime (RT)**. For measuring a runtime, a target precision value $\Delta f_\mathrm{t} = f_\mathrm{target} - f_\mathrm{opt}$ is defined. In a single run, an algorithm can either succeed or fail to reach precision Δf_t. In case of a success, the runtime is the number of function evaluations until Δf_t was reached. In case of a failure we can restart the algorithm. Assuming a positive success probability in a single run (a mild assumption for a stochastic search algorithm) the repeatedly restarted algorithm (that terminates, if Δf_t is reached) has a success probability of one! Its running time is the number function evaluations until Δf_t was reached.

In this paper, *simulated runtime instances* of the virtually restarted algorithm are displayed. We obtain a simulated runtime instance from a *set of given trials* (from the BBOB-2009 data) of the algorithm on a given function: if not a single trial in the set reached Δf_t, we set RT to infinity; otherwise, we draw trials uniformly at random with replacement until a trial is found that reached the target precision Δf_t. The runtime instance is then computed as the sum of function evaluations from all trials drawn. For the last trial only those function evaluations are taken into account that were executed until Δf_t was reached.

The expected value of (the simulated) RT obeys

$$E(\mathrm{RT}(\Delta f_\mathrm{t})) = \widehat{E}(N^\mathrm{s}_\mathrm{eval}) + \frac{1-\widehat{p}_\mathrm{s}}{\widehat{p}_\mathrm{s}}\,\widehat{E}(N^\mathrm{u}_\mathrm{eval}) \ , \qquad (1)$$

where $\widehat{E}(N^\mathrm{s}_\mathrm{eval})$ denotes the average number of function evaluations until Δf_t is reached from those trials that reached Δf_t; $\widehat{E}(N^\mathrm{u}_\mathrm{eval})$ denotes the average number of function evaluation in the remaining (the unsuccessful) trials; $\widehat{p}_\mathrm{s}$ denotes the fraction of trials that reached Δf_t. In fact, the true expected runtime of the (truly) restarted algorithm obeys the same formula [2], where $\widehat{E}(N^\mathrm{s}_\mathrm{eval})$ and $\widehat{E}(N^\mathrm{u}_\mathrm{eval})$ are the ex-

GECCO'10, July 7–11, 2010, Portland, Oregon, USA.

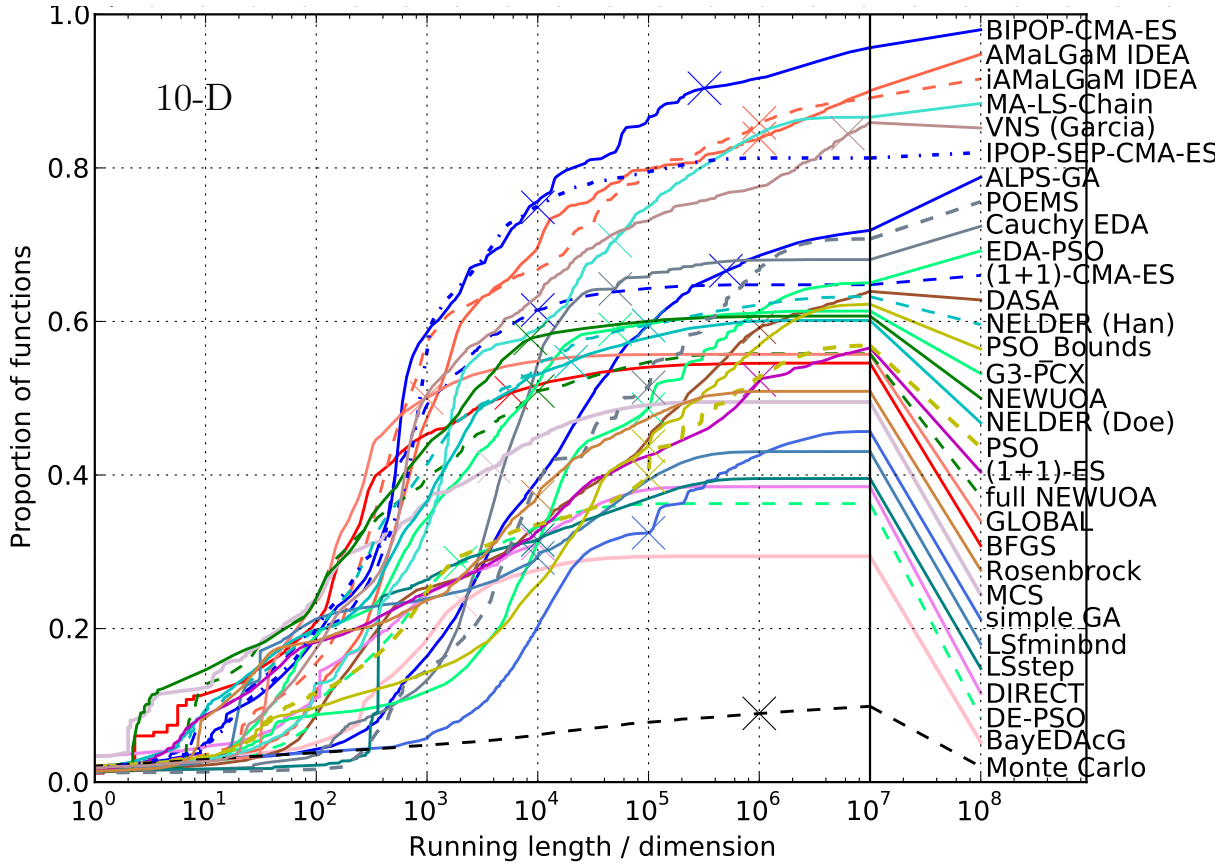

Figure 1: Empirical runtime distributions (runtime in number of function evaluations divided by dimension) on all functions with $\Delta f_\mathrm{t} \in]100, 10^{-8}]$ in dimension 10. The cross indicates the maximum number of function evaluations. A decline in steepness right after the cross (e.g. for IPOP-SEP-CMA-ES) indicates that the maximum number of function evaluations should have been chosen larger. A steep increase right after the cross (e.g. for simple GA) indicates that a restart should have been invoked earlier

pected number of evaluations for successful runs (terminated when Δf_t is reached) and unsuccessful runs respectively, and $\widehat{p}_\mathrm{s}$ is the probability of success.

2. RESULTS

In general, *summarizing results never tell the full story*: even if one algorithm solves more functions much faster than others, it does not necessarily perform superior on *each and every* function.

How to read the figures. Each graph in Figure 1 depicts the empirical cumulative distribution of RT of the annotated algorithm on all functions f_1–f_{24}, in dimension 10. For each function, each Δf_t-value in $\{10^{1.8}, 10^{1.6}, 10^{1.4}, \ldots, 10^{-8}\}$ is used. We write $\Delta f_\mathrm{t} \in]10^2, 10^{-8}]$ for this case and use an analogous notation for other cases. Here and in the following 100 instances of RT are generated (using the method described in Section 1) for each function-Δf_t-pair. For convenience, we refer to a *function-Δf_t-pair* also as a *problem*.

The x-value in the figure shows a given budget, that is, a given number of function evaluations, divided by dimension. The y-value gives the proportion of problems (function-Δf_t-pairs), where the Δf_t-value was reached within the given budget. The graphs are monotonous by definition. Crosses indicate the maximum number of function evaluations observed for the respective algorithm. Results to the right of a cross are only comparable between algorithms with similar maximum number of function evaluations. The limit value to the right indicates the ratio of solved problems.

For any given budget (x-value), the proportion of solved problems (y-value) is a useful performance criterion. Even more useful is the horizontal distance between graphs, revealing a difference in runtime for solving the same proportion of problems. The area between two graphs, up to a given y-value, is the average runtime difference (averaged on the log scale), arguably the most useful aggregated performance measure. The best algorithm covers the largest area under its graph.

Discussion of Figure 1. Overall, the functions are not easy to solve. Within a budget of $100 \times D$ function evaluations, even the best algorithms can only solve 25% of the problems (20% of the problems have a target precision of ≥ 1). The worst algorithms need 100 times larger a budget to solve 25% of the problems and the diversity of results becomes more pronounced for larger budgets.

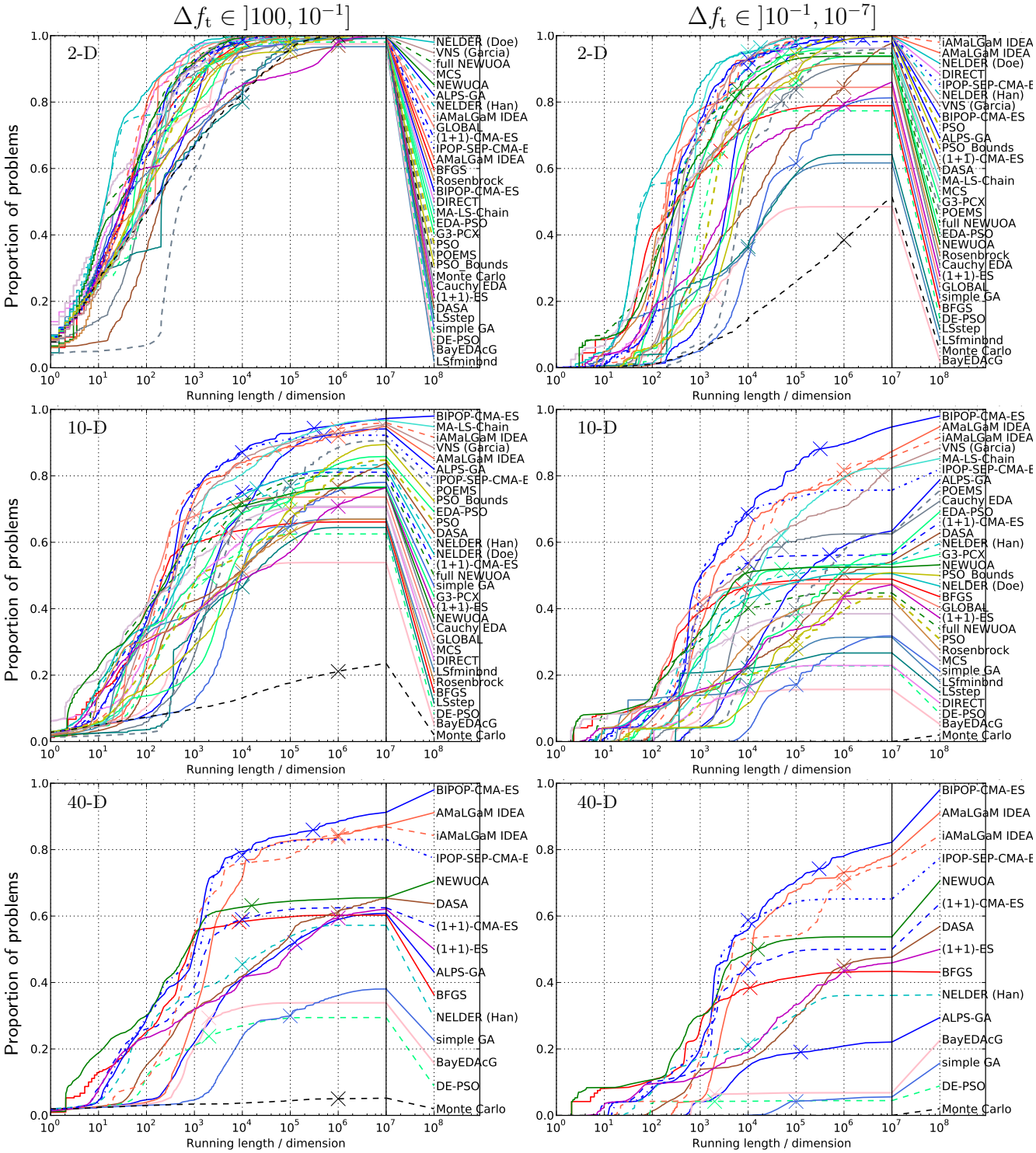

Figure 2: Empirical runtime distributions on all functions with $\Delta f_t \in]100, 10^{-1}]$ (left) and $\Delta f_t \in]10^{-1}, 10^{-7}]$ (right) in dimension 2, 10, 40 (top to bottom)

For budgets below $500D$ function evaluations, the best performance achieve NEWUOA, MCS and GLOBAL. For larger budgets, BIPOP-CMA-ES and IPOP-SEP-CMA-ES become superior. The latter sample in each iteration step several solutions from a multivariate Gaussian distribution like all algorithms with a final success ratio ≥ 0.8.

2.1 Search Space Dimensionality

Figure 2 shows the empirical cumulative distribution of RT from all functions f_1–f_{24} in 2-D, 10-D and 40-D. The right column uses Δf_t-values in $]10^{-1}, 10^{-7}]$.

The overall problem difficulty strongly increases with increasing dimension. In 2-D, pure Monte Carlo search (the

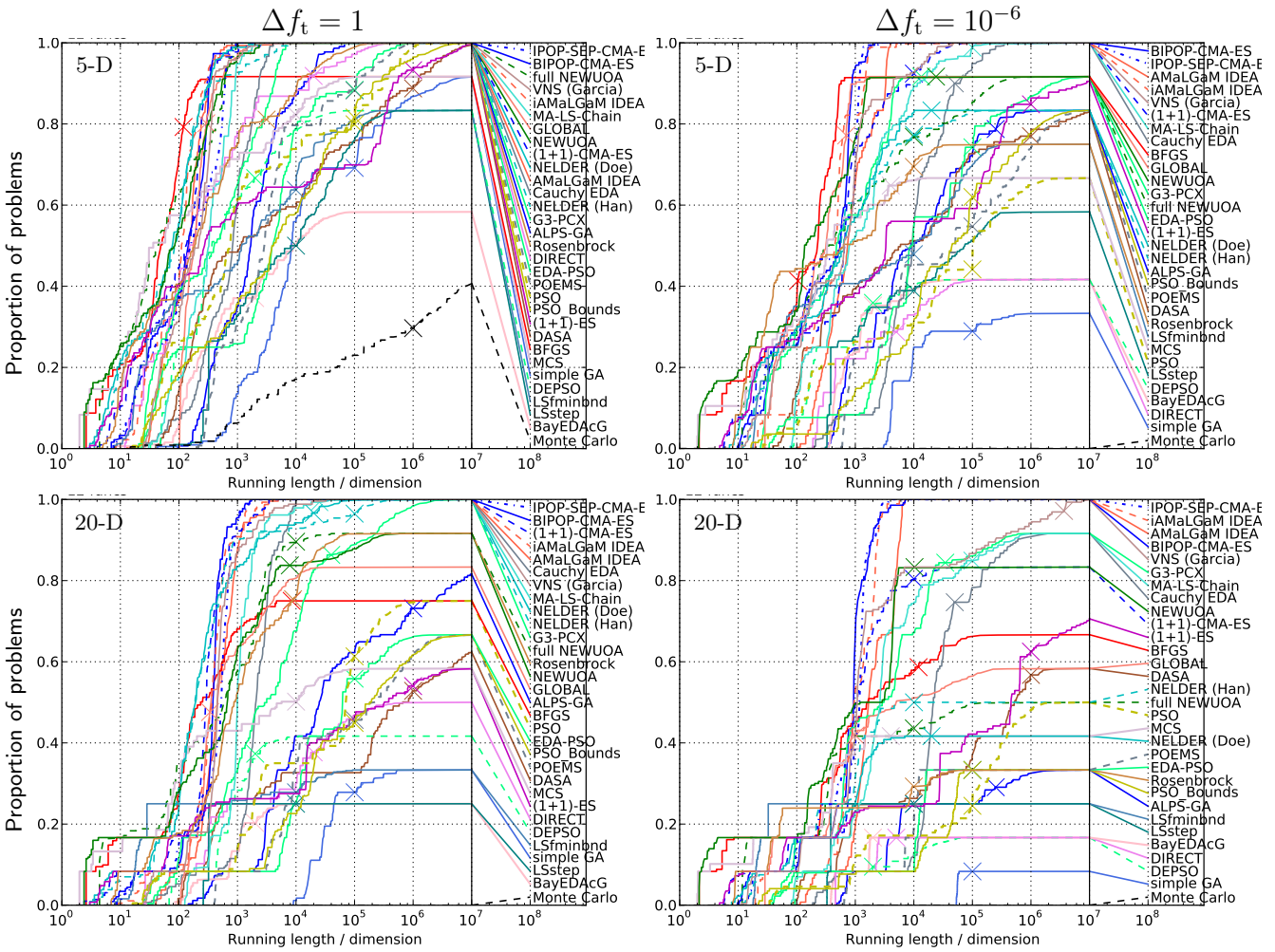

Figure 3: Empirical runtime distributions on 12 unimodal functions f_1, f_2, f_5–f_{14} with target precision $\Delta f_t = 1$ (left) and $\Delta f_t = 10^{-6}$ (right) in dimension 5 (above) and 20 (below)

worst algorithm) can solve about 40% of the problems (function-Δf_t-pairs) in about $10^6 \times D$ function evaluations. In 10-D, the fraction of solved problems becomes invisible for Monte Carlo and half of all algorithms drop below 40%. The spread between the best and the worst algorithms widens remarkably with increasing dimension.

- In **2-D**, NELDER (Doe) is overall clearly the best algorithm. Only for tiny budgets of less than $20D = 40$ function evaluations, it does not solve the most problems. In 3-D, it still performs very well (not shown), while in 5-D other algorithms take over (cp. Fig. 5).
- In **larger dimension**, the picture is more diverse. The best performance depends more significantly on the given budget, as already discussed in Fig. 1.

The left column of Fig. 2 shows data with the more easy target precision values $\Delta f_t \in]100, 10^{-1}]$. The algorithms perform overall better. Nevertheless, more often than not, their individual performance coincides with the one for the more difficult targets.

When the Δf_t-values are set to the Δf-values reached by the best algorithm within D function evaluations, MCS clearly performs best in 20-D, suggesting that MCS has implemented its initial procedures most carefully (not shown).

2.2 Essentially Unimodal Functions

Figure 3 shows results on 12 functions, most of which are unimodal, or they have otherwise an attraction region of the global optimum $\gg 50\%$ (i.e. f_8 and f_9). For target precision 10^{-6} the performance spread is quite pronounced. The above-mentioned set of well-performing algorithms is complemented by BFGS (5-D), full NEWUOA (target precision 1), Rosenbrock (5-D, target precision 10^{-6}), NELDER (Doe) (20-D, target precision 1), and LSfminbnd (20-D).

Figure 4 shows results on three single functions:

f_6 **Attractive Sector** function, a highly asymmetric function, where the optimum lies at the tip of a cone. 15 algorithms show acceptable performance with a performance loss of mostly less than a factor of hundred (horizontal distance) compared to the best algorithm.

f_8 **Rosenbrock** function, a classical test function which has one non-global optimum with an attraction region of smaller than 50%. 15 algorithms show acceptable performance.

f_{10} **Ellipsoid** function, a globally quadratic, ill-conditioned function (condition number 10^6) which is smoothly locally deformed. 12 algorithms show acceptable performance.

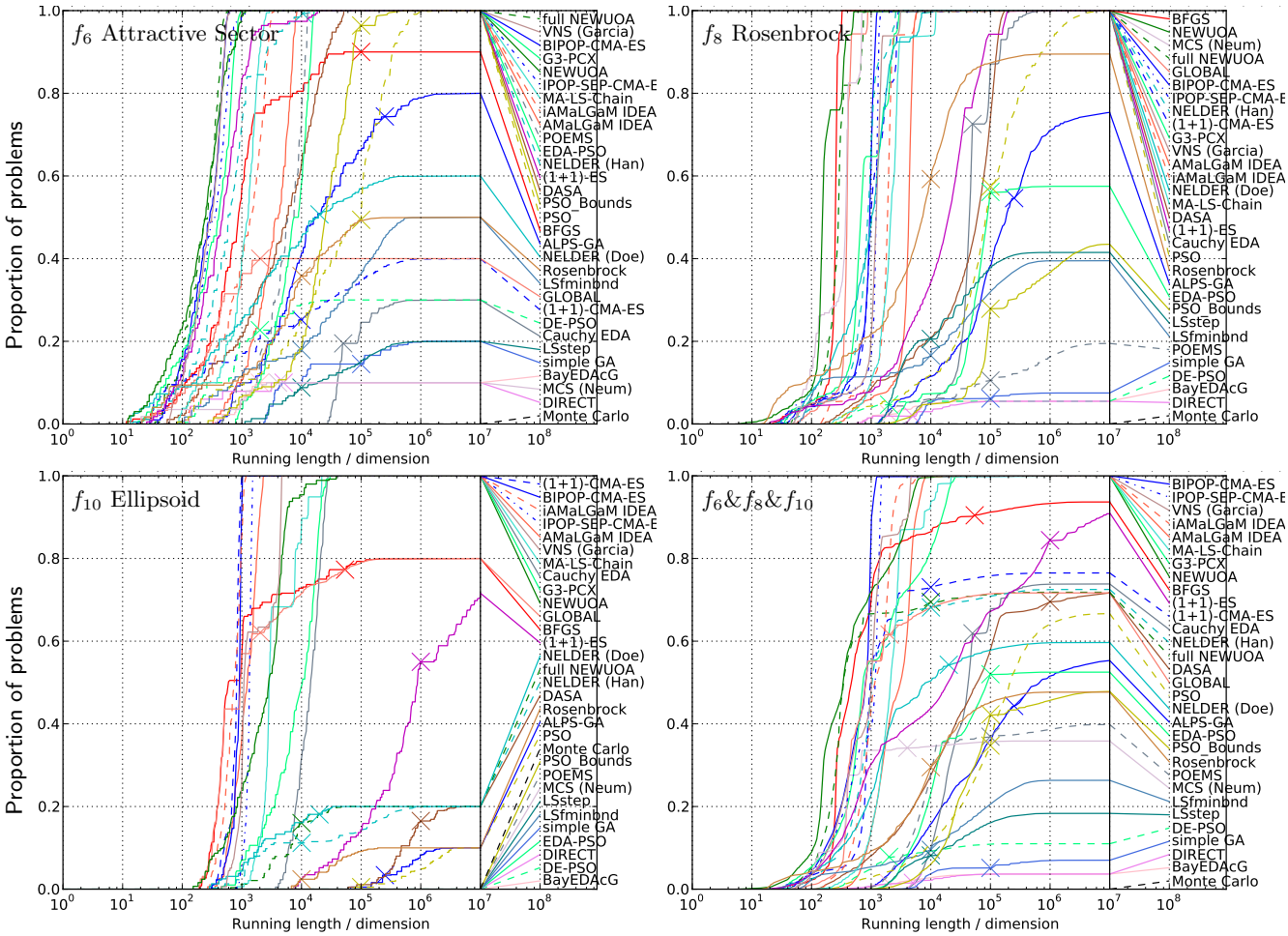

Figure 4: Empirical runtime distributions on single functions with $\Delta f_t \in]10^2, 10^{-8}]$ in dimension 20. For a single trial, the graphs would show a single convergence graph, upside down, with $\Delta f = 10^{-10y+2}$, where $y = -0.1(\log_{10}(\Delta f) - 2) \in [0, 1]$ is the annotated y-value. The lower right figure combines the three other figures

The lower right subfigure combines the convergence data from the three functions. The first nine algorithms listed top (down to BFGS) stay within a performance loss factor of ten (horizontal distance) to the best algorithm up to a y-value of 0.8.

2.3 Multimodal Functions

Figure 5 shows running times on the 12 multimodal functions in dimension 5 and 20 (right column) compared to the unimodal functions (left column). The multimodal functions pose a considerably stronger challenge also with a stronger decline with increasing dimension.

On multimodal functions in 20-D with larger budgets, BIPOP-CMA-ES clearly outperforms all algorithms but IPOP-SEP-CMA-ES, which becomes incomparable for budgets $\gg 10^4 D$. AMaLGaM IDEA outperforms the remaining algorithms for budgets larger than $10^4 D$.

2.4 Function Subgroups

Figure 6 shows results for six subgroups of functions. The following algorithms perform particularly well up to their individual maximum number of function evaluations, forming more than 10% of the left envelope of the set of graphs: on separable functions NEWUOA, LSfminbnd and LSstep; on moderate functions NEWUOA and IPOP-SEP-CMA-ES; on ill-conditioned functions GLOBAL, iAMaLGaM and BIPOP-CMA-ES; on the multi-modal structured functions IPOP-SEP-CMA-ES and BIPOP-CMA-ES; on the multimodal weakly structured functions GLOBAL and BIPOP-CMA-ES; on non-smooth functions iAMaLGaM and BIPOP-CMA-ES.

The IDEA and *POP*-CMA variants show a quite similar performance characteristics over the subgroups.

3. CONCLUSIONS

We draw some summarizing conclusions on the BBOB-2009 data set.

Benchmarks. The benchmark function testbed is comparatively difficult. In dimension 20, within $10^5 D$ function evaluations, the best algorithm can solve about 75% of the functions up to a precision of 10^{-6}, the median algorithm solves about 30%. For the multimodal functions the rate is about 50% (median below 20%).

Empirical run time distributions (cf. Fig. 4). A single convergence graph—plotting the best achieved f-value against time—can be interpreted, when plotted upside down, as a cumulative runtime distribution for the set of all f-

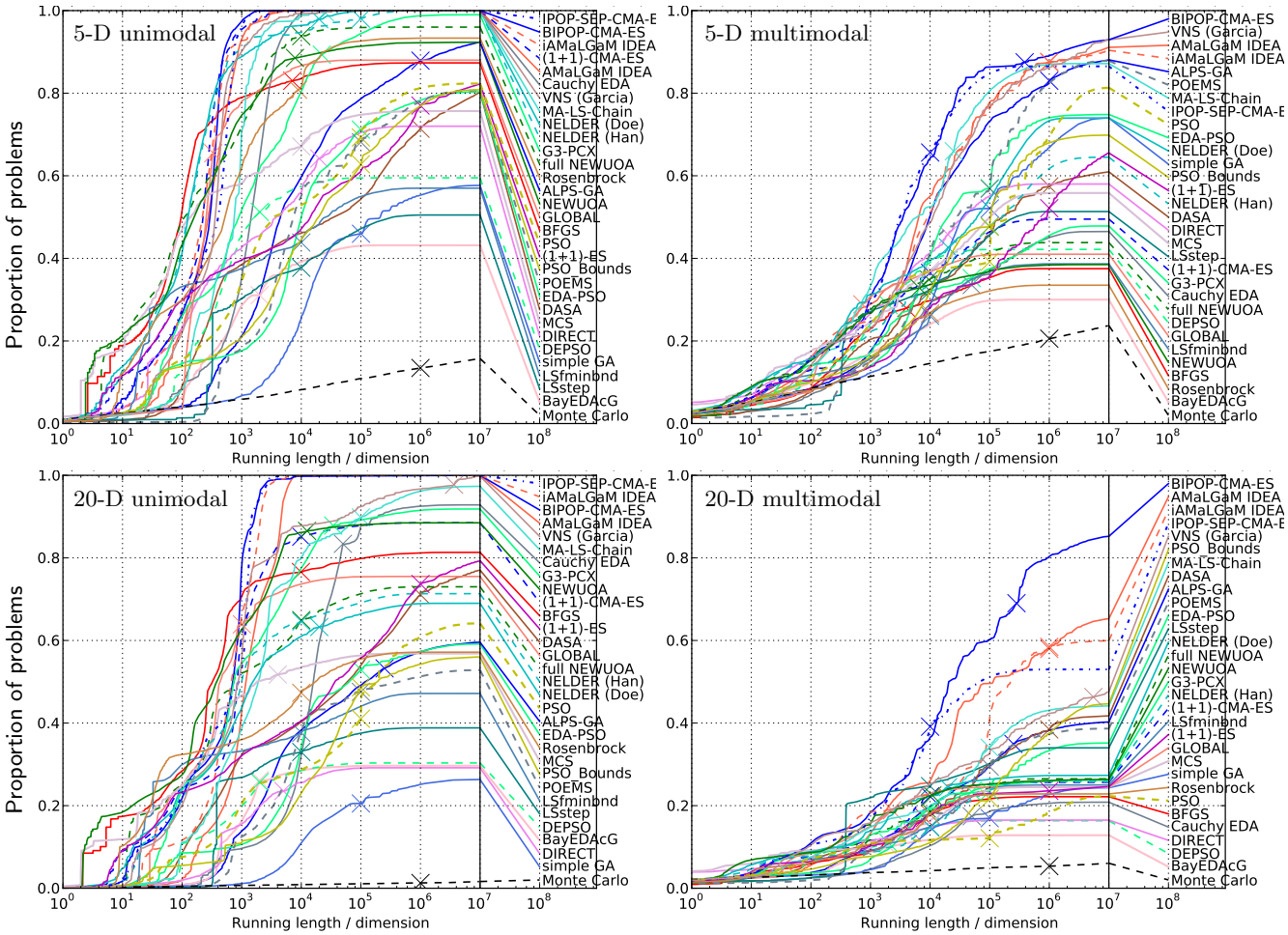

Figure 5: Empirical runtime distributions on unimodal (left) and multimodal (right) functions with $\Delta f_t \in]10^2, 10^{-8}]$ in dimension 5 and 20. The step observed for LSstep in the lower right is due to the separable functions (see Fig. 6)

values. Exploiting this interpretation, convergence data from several trials can be combined into a single graph. Even data from various functions can be merged into a single graph. During this integration only the labels of single data points to individual trials and functions are lost.

Impact on performance. A strong impact on the function difficulty can be found from dimensionality, multi-modality, and non-smoothness. Also different constraints for the time budget (number of function evaluations) have a great impact on which algorithms perform best.

Algorithms. For very low dimension, NELDER (Doe) was superior. For lower budgets NEWUOA, MCS and GLOBAL were the best algorithms. For difficult functions and larger budgets, variants of CMA-ES performed best, followed by the AMaLGaM-IDEA variants. The results can provide a clear guideline for the choice of an algorithm or of an ensemble of algorithms in an appropriate way to solve an unknown black-box optimization problem.

4. REFERENCES

[1] A. Auger. Benchmarking the (1+1) evolution strategy with one-fifth success rule on the BBOB-2009 function testbed. In Rothlauf (2009, [34]), pages 2447–2452.

[2] A. Auger and N. Hansen. Performance evaluation of an advanced local search evolutionary algorithm. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2005)*, pages 1777–1784, 2005.

[3] A. Auger and N. Hansen. Benchmarking the (1+1)-CMA-ES on the BBOB-2009 function testbed. In Rothlauf (2009, [34]), pages 2459–2466.

[4] A. Auger and R. Ros. Benchmarking the pure random search on the BBOB-2009 testbed. In Rothlauf (2009, [34]), pages 2479–2484.

[5] P. A. N. Bosman, J. Grahl, and D. Thierens. AMaLGaM IDEAs in noiseless black-box optimization benchmarking. In Rothlauf (2009, [34]), pages 2247–2254.

[6] B. Doerr, M. Fouz, M. Schmidt, and M. Wahlström. BBOB: Nelder-Mead with resize and halfruns. In Rothlauf (2009, [34]), pages 2239–2246.

[7] M. El-Abd and M. S. Kamel. Black-box optimization benchmarking for noiseless function testbed using an EDA and PSO hybrid. In Rothlauf (2009, [34]), pages 2263–2268.

[8] M. El-Abd and M. S. Kamel. Black-box optimization benchmarking for noiseless function testbed using particle swarm optimization. In Rothlauf (2009, [34]), pages 2269–2274.

[9] M. El-Abd and M. S. Kamel. Black-box optimization benchmarking for noiseless function testbed using PSO_Bounds. In Rothlauf (2009, [34]), pages 2275–2280.

[10] S. Finck, N. Hansen, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2009: Presentation of

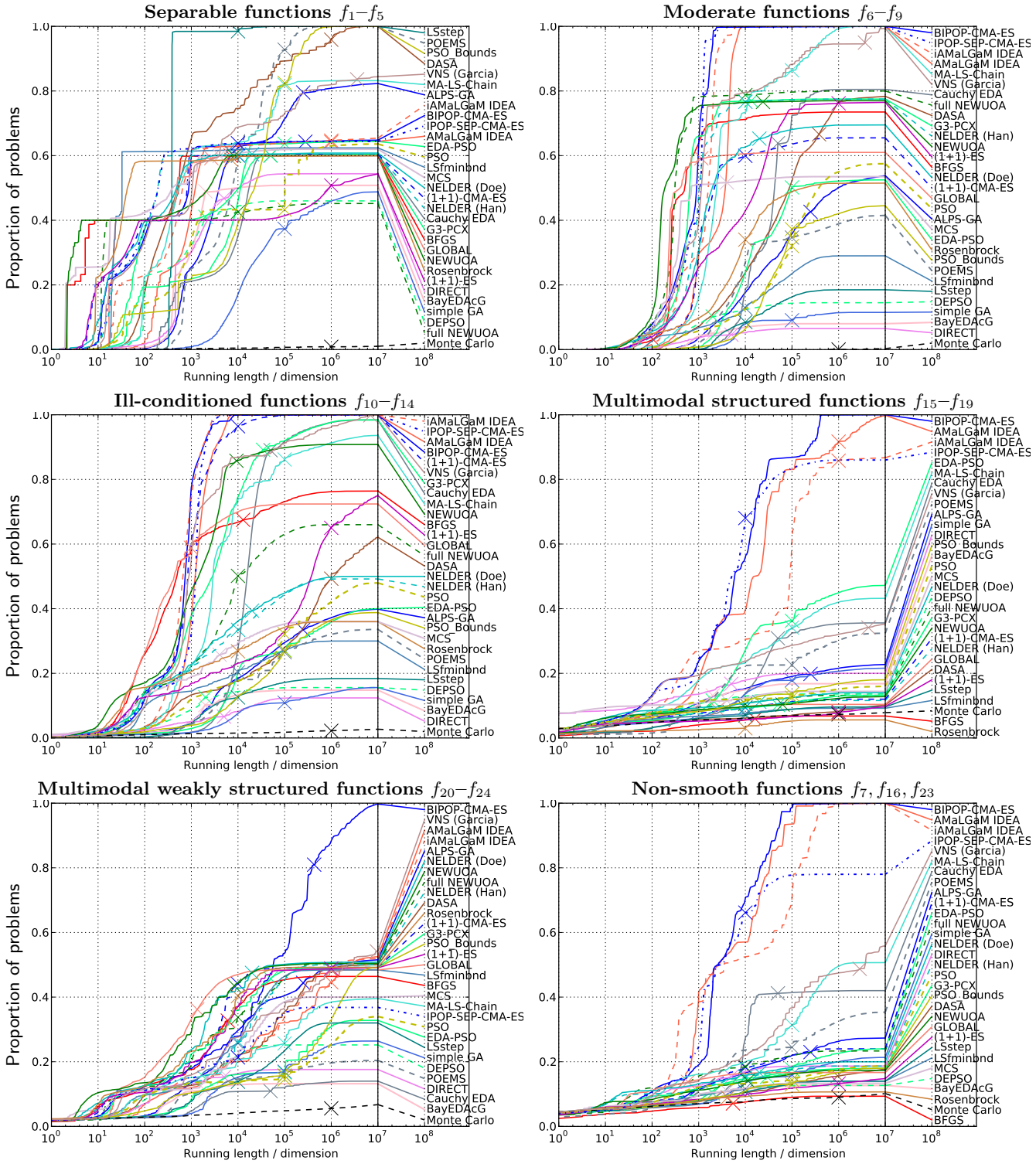

Figure 6: Empirical runtime distributions on function sub-groups with $\Delta f_t \in]100, 10^{-8}]$ in dimension 20

the noiseless functions. Technical Report 2009/20, Research Center PPE, 2009.

[11] M. Gallagher. Black-box optimization benchmarking: results for the BayEDAcG algorithm on the noiseless function testbed. In Rothlauf (2009, [34]), pages 2281–2286.

[12] C. García-Martínez and M. Lozano. A continuous variable neighbourhood search based on specialised EAs: application to the noiseless BBO-benchmark 2009. In Rothlauf (2009, [34]), pages 2287–2294.

[13] J. García-Nieto, E. Alba, and J. Apolloni. Noiseless functions black-box optimization: evaluation of a hybrid

particle swarm with differential operators. In Rothlauf (2009, [34]), pages 2231–2238.

[14] N. Hansen. Benchmarking a BI-population CMA-ES on the BBOB-2009 function testbed. In Rothlauf (2009, [34]), pages 2389–2396.

[15] N. Hansen. Benchmarking the Nelder-Mead downhill simplex algorithm with many local restarts. In Rothlauf (2009, [34]), pages 2403–2408.

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

[17] 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.

[18] G. S. Hornby. The age-layered population structure (ALPS) evolutionary algorithm. `http://coco.gforge.inria.fr/doku.php?id=bbob-2009-results`, July 2009. Noiseless testbed.

[19] G. S. Hornby. Steady-state ALPS for real-valued problems. In *GECCO '09: Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 795–802, New York, NY, USA, 2009. ACM.

[20] W. Huyer and A. Neumaier. Benchmarking of MCS on the noiseless function testbed. `http://www.mat.univie.ac.at/~neum/papers.html`, 2009. P. 989.

[21] P. Korosec and J. Silc. A stigmergy-based algorithm for black-box optimization: noiseless function testbed. In Rothlauf (2009, [34]), pages 2295–2302.

[22] J. Kubalik. Black-box optimization benchmarking of prototype optimization with evolved improvement steps for noiseless function testbed. In Rothlauf (2009, [34]), pages 2303–2308.

[23] D. Molina, M. Lozano, and F. Herrera. A memetic algorithm using local search chaining for black-box optimization benchmarking 2009 for noise free functions. In Rothlauf (2009, [34]), pages 2255–2262.

[24] M. Nicolau. Application of a simple binary genetic algorithm to a noiseless testbed benchmark. In Rothlauf (2009, [34]), pages 2473–2478.

[25] L. Pál, T. Csendes, M. C. Markót, and A. Neumaier. BBO-benchmarking of the GLOBAL method for the noiseless function testbed. `http://www.mat.univie.ac.at/~neum/papers.html`, 2009. P. 986.

[26] P. Pošík. BBOB-benchmarking a simple estimation of distribution algorithm with Cauchy distribution. In Rothlauf (2009, [34]), pages 2309–2314.

[27] P. Pošík. BBOB-benchmarking the DIRECT global optimization algorithm. In Rothlauf (2009, [34]), pages 2315–2320.

[28] P. Pošík. BBOB-benchmarking the generalized generation gap model with parent centric crossover. In Rothlauf (2009, [34]), pages 2321–2328.

[29] P. Pošík. BBOB-benchmarking the Rosenbrock's local search algorithm. In Rothlauf (2009, [34]), pages 2337–2342.

[30] P. Pošík. BBOB-benchmarking two variants of the line-search algorithm. In Rothlauf (2009, [34]), pages 2329–2336.

[31] R. Ros. Benchmarking sep-CMA-ES on the BBOB-2009 function testbed. In Rothlauf (2009, [34]), pages 2435–2440.

[32] R. Ros. Benchmarking the BFGS algorithm on the BBOB-2009 function testbed. In Rothlauf (2009, [34]), pages 2409–2414.

[33] R. Ros. Benchmarking the NEWUOA on the BBOB-2009 function testbed. In Rothlauf (2009, [34]), pages 2421–2428.

[34] F. Rothlauf, editor. *Genetic and Evolutionary Computation Conference, GECCO 2009, Proceedings, Montreal, Québec, Canada, July 8-12, 2009, Companion Material*. ACM, 2009.

APPENDIX

Used Acronyms

GA/EA: Genetic Algorithm / Evolutionary Algorithm
EDA: Estimation of Distribution Algorithm
CMA: Covariance Matrix Adaptation
ES: Evolution Strategy
PSO: Particle Swarm Optimization

Algorithms

ALPS-GA: Age-Layered Population Structure running a standard GA in 12 layers	[19, 18]
AMaLGaM IDEA: Adapted Maximum-Likelihood Gaussian Model Iterated Density Estimation Algorithm with no-improvement stretch, anticipated mean shift and interlaced restarts with one large or several small populations	[5]
iAMaLGaM IDEA: with incremental model building	[5]
BayEDAcG: An EDA using Bayesian inference to learn the parameters of the continuous Gaussian distribution	[11]
BFGS: Quasi-Newton method with MATLABs `fminunc`	[32]
Cauchy EDA: EDA with isotropic Cauchy sampling distribution	[26]
BIPOP-CMA-ES: CMA-ES restarted with budgets for small and large population size	[14]
(1+1)-CMA-ES	[3]
DASA: Differential Ant-Stigmergy Algorithm using pheromones on the differential graph that represents parameter difference	[21]
DEPSO: PSO with Differential Evolution variations	[13]
DIRECT: Axis-parallel search space partitioning procedure	[27]
EDA-PSO: hybrid of EDA and PSO with adapted probability of applying one or the other	[7]
G3-PCX: Generalized Generation Gap with Parent Centric Crossover (local-search intensive variant)	[28]
simple GA: binary coded GA	[24]
GLOBAL: Sampling, clustering and local search using BFGS or Nelder-Mead	[25]
LSfminbnd: Axis-parallel line search with MATLAB `fminbnd` univariate search	[30]
LSstep: Axis-parallel line search with the univariate STEP Select The Easiest Point, based on interval division	[30]
MA-LS-Chain: Memetic Algorithm with Local Search Chaining using a steady-state GA and CMA-ES for local search with a fixed local/global search ratio	[23]
MCS: Multilevel Coordinate Search like DIRECT with additional local searches by triple search	[20]
NELDER (Han): Nelder-Mead downhill simplex restarted using coordinate-wise projections	[15]
NELDER (Doe): Nelder-Mead downhill simplex with restarted half-runs	[6]
NEWUOA: NEW Unconstraint Optimization Algorithm builds a second order model using $2n + 1$ points and with minimal Frobenius norm	[33]
full NEWUOA: using $(n^2 + 3n + 2)/2$ points	[33]
(1+1)-ES: with 1/5th success rule for step-size adaptation	[1]
POEMS: Prototype Optimization with Evolved Improvement Steps using hypermutations and stochastic local search	[22]
PSO: standard PSO with swarm size 40 and no restarts	[8]
PSO_Bounds: as PSO and diving the search domain based on a concept from PBIL	[9]
Monte Carlo: uniform random sampling	[4]
Rosenbrock: Local search algorithm maintaining an orthogonal basis for the adaptation of search directions	[29]
IPOP-SEP-CMA-ES: CMA-ES restarted with Increa- sing POPulation size, first run with SEParable CMA	[31]
VNS: Variable Neighbourhood Search combining CMA-ES, PBX-α-EA and μCHC	[12]