Comparing Results of 31 Algorithms from the Black-Box Optimization Benchmarking BBOB-2009
Nikolaus Hansen, Anne Auger, Raymond Ros, Steffen Finck, Petr Posik

To cite this version:
Nikolaus Hansen, Anne Auger, Raymond Ros, Steffen Finck, Petr Posik. Comparing Results of 31 Algorithms from the Black-Box Optimization Benchmarking BBOB-2009. ACM-GECCO Genetic and Evolutionary Computation Conference, Jul 2010, Portland, United States. hal-00545727

HAL Id: hal-00545727
https://hal.archives-ouvertes.fr/hal-00545727
Submitted on 11 Dec 2010
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 Δf = f_target − f_opt is defined. In a single run, an algorithm can either succeed or fail to reach precision Δf. In case of a success, the runtime is the number of function evaluations until Δf 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 is reached) has a success probability of one! Its running time is the number function evaluations until Δf 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, 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. 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 was reached.

The expected value of (the simulated) RT obeys

\[ E(RT(\Delta f)) = \bar{E}(N_{eval}^u) + \frac{1 - \hat{p}_u}{\hat{p}_u} \bar{E}(N_{eval}^u), \]

where \( \bar{E}(N_{eval}^u) \) denotes the average number of function evaluations until \( \Delta f \) is reached from those trials that reached \( \Delta f \); \( \bar{E}(N_{eval}^{u_{opt}}) \) denotes the average number of function evaluation in the remaining (the unsuccessful) trials; \( \hat{p}_u \) denotes the fraction of trials that reached \( \Delta f \). In fact, the true expected runtime of the (truly) restarted algorithm obeys the same formula [2], where \( \bar{E}(N_{eval}^u) \) and \( \bar{E}(N_{eval}^{u_{opt}}) \) are the ex-
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 $\Delta f_t$-value in $\{10^{-1}, 10^{-2}, 10^{-3}, \ldots, 10^{-8}\}$ is used. We write $\Delta f_t \in [10^{2}, 10^{-8}]$ 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-$\Delta f_t$-pair. For convenience, we refer to a function-$\Delta f_t$-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-$\Delta f_t$-pairs), where the $\Delta f_t$-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 $\geq 1$). 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.
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 $\geq 0.8$.

### 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 $\Delta f_i$-values in $[10^{-1}, 10^{-2}]$.

The overall problem difficulty strongly increases with increasing dimension. In 2-D, pure Monte Carlo search (the
worst algorithm) can solve about 40% of the problems (function-∆f-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 $\Delta f \in [100, 10^{-1}]$. The algorithms perform overall better. Nevertheless, more often than not, their individual performance coincides with the one for the more difficult targets.

When the $\Delta f$-values are set to the $\Delta f$-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 LSminbnd (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 $10^{10}$) which is smoothly locally deformed. 12 algorithms show acceptable performance.
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 IPopo-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 IPopo-SEP-CMA-ES and BIPOP-CMA-ES; on the multi-modal 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 tested 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 \( f \)-
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
Figure 6: Empirical runtime distributions on function sub-groups with $\Delta f_t \in [100, 10^{-6}]$ in dimension 20.


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
AMAaGM 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
iAMAaGM IDEA: with incremental model building
BayEDAaG: An EDA using incremental model building
parameters of the continuous Gaussian distribution
BFGS: Quasi-Newton method with MATLAB’s fminunc
Cauchy EDA: EDA with isotropic Cauchy sampling distribution
BIP0P-CMA-ES: CMA-ES restarted with budgets for small and large population size
(1+1)-CMA-ES
DASA: Differential Ant-Stigmergy Algorithm using pheromones on the differential graph that represents parameter difference
DEPSO: PSO with Differential Evolution variations
DIRECT: Axis-parallel search space partitioning procedure
EDA-PSO: hybrid of EDA and PSO with adapted probability of applying one or the other
G3-PCX: Generalized Generation Gap with Parent Centric Crossover (local-search intensive variant)
simple GA: binary coded GA
GLOBAL: Sampling, clustering and local search using BFGS or Nelder-Mead
LSminbnd: Axis-parallel line search with MATLAB’s fminbnd univariate search
LStep: Axis-parallel line search with the univariate STEP Select The Easiest Point, based on interval division
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
MCS: Multilevel Coordinate Search like DIRECT with additional local searches by triple search
NELDER (Han): Nelder-Mead downhill simplex restarted using coordinate-wise projections
NELDER (Doe): Nelder-Mead downhill simplex with restarted half-runs
NEWUOA: NEW Unconstraint Optimization Algorithm builds a second order model using 2n + 1 points and with minimal Frobenius norm
full NEWUOA: using $(n^2 + 3n + 2)/2$ points
(1+1)-ES: with 1/5th success rule for step-size adaptation
POEMS: Prototype Optimization with Evolved Improvement Steps using hypermutations and stochastic local search
PSO: standard PSO with swarm size 40 and no restarts
PSO_Bounds: as PSO and diving the search domain based on a concept from PHIL
Monte Carlo: uniform random sampling
Rosenbrock: Local search algorithm maintaining an orthogonal basis for the adaptation of search directions
IPOP-SEP-CMA-ES: CMA-ES restarted with Incremental POPulation size, first run with SEParable CMA
VNS: Variable Neighbourhood Search combining CMA-ES, PBX-α-EA and μCHC

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
AMAaGM 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
iAMAaGM IDEA: with incremental model building
BayEDAaG: An EDA using incremental model building
parameters of the continuous Gaussian distribution
BFGS: Quasi-Newton method with MATLAB’s fminunc
Cauchy EDA: EDA with isotropic Cauchy sampling distribution
BIP0P-CMA-ES: CMA-ES restarted with budgets for small and large population size
(1+1)-CMA-ES
DASA: Differential Ant-Stigmergy Algorithm using pheromones on the differential graph that represents parameter difference
DEPSO: PSO with Differential Evolution variations
DIRECT: Axis-parallel search space partitioning procedure
EDA-PSO: hybrid of EDA and PSO with adapted probability of applying one or the other
G3-PCX: Generalized Generation Gap with Parent Centric Crossover (local-search intensive variant)
simple GA: binary coded GA
GLOBAL: Sampling, clustering and local search using BFGS or Nelder-Mead
LSminbnd: Axis-parallel line search with MATLAB’s fminbnd univariate search
LStep: Axis-parallel line search with the univariate STEP Select The Easiest Point, based on interval division
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
MCS: Multilevel Coordinate Search like DIRECT with additional local searches by triple search
NELDER (Han): Nelder-Mead downhill simplex restarted using coordinate-wise projections
NELDER (Doe): Nelder-Mead downhill simplex with restarted half-runs
NEWUOA: NEW Unconstraint Optimization Algorithm builds a second order model using 2n + 1 points and with minimal Frobenius norm
full NEWUOA: using $(n^2 + 3n + 2)/2$ points
(1+1)-ES: with 1/5th success rule for step-size adaptation
POEMS: Prototype Optimization with Evolved Improvement Steps using hypermutations and stochastic local search
PSO: standard PSO with swarm size 40 and no restarts
PSO_Bounds: as PSO and diving the search domain based on a concept from PHIL
Monte Carlo: uniform random sampling
Rosenbrock: Local search algorithm maintaining an orthogonal basis for the adaptation of search directions
IPOP-SEP-CMA-ES: CMA-ES restarted with Incremental POPulation size, first run with SEParable CMA
VNS: Variable Neighbourhood Search combining CMA-ES, PBX-α-EA and μCHC