Performance Evaluation of Anytime Black Box Optimizers
Black-Box Optimization (Search)

Minimize an objective function (also: cost, loss, error, or fitness function)

\[ f : \mathcal{X} \subset \mathbb{R}^n \rightarrow \mathbb{R}, \quad x \mapsto f(x) \]

in a black-box scenario (direct search, no gradients)

\[ x \rightarrow \boxed{\text{black box}} \rightarrow f(x) \]

where the black box can be

- non-linear, non-convex, discontinuous, dynamic, stochastic
- from milli-seconds to hours to evaluate

Objective:

- convergence to a global essential infimum of \( f \) as fast as possible
- (informally, time-finite) find \( x \in \mathcal{X} \) with small \( f(x) \) value using as few back-box calls (function evaluations) as possible
Why Do We Need to Measure Performance?

• putting algorithms to a *standardized* test
  - simplify judgement
  - simplify comparison
  - regression test/quality check under algorithm changes

• algorithm selection

• understanding of algorithms
How do we measure performance?
We can measure performance on

- **real world problems**
  - expensive
  - comparison is typically limited to certain domains
  - experts have limited interest to publish

- "artificial" **benchmark functions**
  - cheap
  - data acquisition is comparatively easy
  - problem of representativity

- caveat: parameter of algorithms
...empirically...

- **convergence graphs** is all we have to start with

- the **right presentation** cannot be overestimated

  the details are important
Displaying Three Runs (three trials)

not like this (it's unfortunately a common picture)
better like this (shown are the same data), caveat: fails with negative f-values
even better like this: subtract minimum value over all runs
Displaying 51 Runs

don't hesitate to display all data (the appendix is your friend)

☆: final value

observation: three different "modes", which would be difficult to represent or recover in single statistics
Performance Evaluation of Anytime Black Box Optimizers

Nikolaus Hansen

Object Variables (mean, 19-D, popsize~12)

Scaling (All Main Axes)

Standard Deviations in All Coordinates
Which Statistic?

\[ f\text{-offset} = -3.14159265359 + 1e-11 \]
mean/average function value

- tends to emphasize large values
Which Statistic?

geometric average function value \( \exp\left(\text{mean}_i(\log(f_i))\right) = \left(\prod_{i=1}^{N} f_i\right)^{1/N} \)
- reflects "visual" average
- depends on offset
- artefact due to adding 1e-11
average iterations
  • reflects "visual" average
  • here: incomplete
the median is invariant

- unique for uneven number of data
- independent of log-scale, offset...
  \[
  \text{median} (\log(\text{data})) = \log(\text{median} (\text{data}))
  \]
- same when taken over x- or y-direction
Implication

• use the median as summary datum
• more general: use quantiles as summary data

for example out of 15 data: 2nd, 8th, and 14th value represent the 10%, 50%, and 90%-tile

unless there are good reasons for a different statistic
Examples

Comparison of 4 algorithms using the "median run" and the 90% central range of the final value on two different functions (Ellipsoid and Rastrigin)

caveat: this range display with simple error bars fails, if, e.g., 30% of all runs "converge"
Examples: Plotting All Data

Experiments from two algorithms, A1 and A2
Statistical Assessment

• Don't be scared!

1) Assess the meaning/relevance of a difference first (the only difficult part)

2) Apply rank-sum test (Wilcoxon, Mann-Whitney U)
   • only assumption: no equal data values
     as usual: useful even if assumptions do not hold, for categorical data: \( \chi^2 \)-test
   • hypothesis: \( p(x > y) \neq p(x < y) \neq 1/2 \)
   • compares sum of ranks in a combined ranking
   • two-sided 1%-significance \( p \)-value needs only 2x5 data values
   • For the same \( p \)-value, fewer significant data are better

Generally: non-parametric tests, Kolmogorov-Smirnov test for ECDFs, no need to use the t-test
Performance Measure(s)

Runtime
Three Convergence Graphs

recall: convergence graphs is all we have
(recall) Black-Box Optimization

Two objectives:

- Find solution with small(est possible) function value
- With the least possible search costs (number of function evaluations)
- For measuring performance: fix one and measure the other
Two objectives

Convergence graph is a plot in objective space

\[ f_{\text{offset}} = -3.14159265359 + 1e-11 \]
Measuring Performance from Convergence Graphs

fixed-cost versus fixed-target

(best achieved) function value

number of function evaluations (time)

five repetitions

fixed cost

fixed target
Measuring Performance from Convergence Graphs

fixed-cost versus fixed-target

(best achieved) function value

number of function evaluations (time)

five repetitions

fixed target

fixed cost
Measuring Performance from Convergence Graphs

fixed-cost versus fixed-target

(best achieved) function value

number of function evaluations (time)

five repetitions

fixed target

fixed cost

five repetitions
a performance should be

- **quantitative** on the ratio scale (highest possible)
  - “algorithm A is two times better than algorithm B” is a meaningful statement

- can assume a wide range of values

- **meaningful (interpretable)** with regard to the real world possible to transfer from benchmarking to real world

**runtime** or **first hitting time** is the prime candidate (we don't have many choices anyway)
The performance measure we use

Run length or runtime or first hitting time to a given target function value measured in number of fitness function evaluations

equivalent to first hitting time of a sublevel set in search space

How can we deal with "missing values"?
Measuring Performance from Convergence Graphs

fixed-cost versus fixed-target

(best achieved) function value

five repetitions

fixed target

number of function evaluations (time)
Fixed-target: Measuring Runtime

1. Fix a target $f$-value (most difficult part)

2. Compute the success rate $\hat{p}$ as

$$
\hat{p} = \frac{\text{# of successful runs (that reached the target)}}{\text{# of all runs}} \in [0, 1]
$$

$$
\hat{R} = \frac{1 - \hat{p}}{\hat{p}} = \frac{\text{# of unsuccessful runs}}{\text{# of successful runs}} \in [0, \infty]
$$

$\hat{R}$ is the odds ratio to be unsuccessful

$\hat{R}$ is the number of unsuccessful runs observed for each single successful run (i.e. normalized by # of successful runs)
Fixed-target: Measuring Runtime

\[ \hat{R} = \frac{1 - \hat{p}}{\hat{p}} = \frac{\text{# of unsuccessful runs}}{\text{# of successful runs}} \in [0, \infty] \]

\( \hat{R} \) is the odds ratio to be unsuccessful

\( \hat{R} \) is the number of unsuccessful runs observed for each single successful run (i.e. normalized by # of successful runs)

3. Compute "expected runtime" to hit the target

average runtime for a single successful run

\[ \text{ERT} := \overline{RT}_{\text{succ}} + \hat{R} \times \overline{RT}_{\text{unsucc}} \]

average runtime spent in unsuccessful runs to achieve one successful run

\[ \text{SP1} := \overline{RT}_{\text{succ}} + \hat{R} \times \overline{RT}_{\text{succ}} = \overline{RT}_{\text{succ}} (1 + \hat{R}) \]

disregarding runlength of unsuccessful runs

if \( \hat{R} < \infty \), else we can assume \( \text{ERT} \geq \sum RT_{\text{unsucc}} \)
Fixed-target: Measuring Runtime

\( \hat{R} \) is the number of unsuccessful runs observed for each single successful run (i.e. normalized by # of successful runs)

3. Compute "expected runtime" to hit the target

average runtime for a single successful run

\[
ERT := \overbrace{RT_{succ}} + \underbrace{\hat{R} \times RT_{unsucc}}
\]

average runtime spent in unsuccessful runs to achieve one successful run

\[
SP1 := \overbrace{RT_{succ}} + \underbrace{\hat{R} \times RT_{succ}} = RT_{succ}(1 + \hat{R})
\]

disregarding runlength of unsuccessful runs

We can simulate a single runtime by "restarting" until the first success

\[
RT = RT_{succ} + \sum RT_{unsucc}
\]

\( \implies \) distribution of runtimes incorporating unsuccessful runs

\( \implies \) display the distribution or a statistic of it
Break
Summary

- plot carefully
- display all data
- use the median as summary datum
  unless for runtimes or you know exactly what you do
- more general: use quantiles as summary data
- assess a performance difference before to worry about statistical significance
- vertical vs. horizontal view-point
- run"time" RT and
  - ERT (expected RT)
  - runtime ECDF (empirical cumulative distribution fct)
ECDF:
Empirical Cumulative Distribution Function of the Runtime
A Convergence Graph

function value vs \log_{10}(\text{function evaluations})
First hitting time is monotonous

- first hitting time: a monotonous graph
Performance Evaluation of Anytime Black Box Optimizers

- another convergence graph
• another convergence graph with hitting time
• a target value delivers two data points
• a target value delivers two data points
ECDF with four data points
- reconstructing a single run
Performance Evaluation of Anytime Black Box Optimizers

Nikolaus Hansen

50 equally spaced targets
Performance Evaluation of Anytime Black Box Optimizers

Nikolaus Hansen

The graph shows the function value on the y-axis against the logarithm of the number of function evaluations on the x-axis. The data points are marked with red stars and a blue line connects them, indicating a decreasing trend as the number of evaluations increases.
the ECDF recovers the monotonous graph
the ECDF recovers the monotonous graph, discretised and flipped
the ECDF recovers the monotonous graph, discretised and flipped
The ECDF recovers the monotonous graph, discretised and flipped.

The area over the ECDF curve is the average log runtime (or geometric average runtime).
Performance Evaluation of Anytime Black Box Optimizers

Nikolaus Hansen
Performance Evaluation of Anytime Black Box Optimizers

Nikolaus Hansen
the ECDF of run lengths (runtimes)

80% of the runs reached the target
Performance Evaluation of Anytime Black Box Optimizers

Nikolaus Hansen

15 runs
50 targets
Performance Evaluation of Anytime Black Box Optimizers

15 runs
50 targets
15 runs
50 targets
ECDF with 750 steps
15 runs integrated in a single graph.
Target budgets/run-lengths

1) define reference target budgets
Target budgets on the reference algorithm

1) define reference target budgets

2) compute best function value achieved by a reference algorithm

![Graph showing function evaluations vs. function value](image-url)
Target budgets on the reference algorithm

1) define reference target budgets
2) compute best function value achieved by a reference algorithm

The graph shows the function value on the y-axis and the logarithm of the number of function evaluations on the x-axis. The curve indicates the performance of an optimization algorithm over the course of evaluations.
Run-length based target $f$-values

1) define reference target budgets
2) compute best function value achieved by a reference algorithm

=> set of target function values
Run-length based target $f$-values

1) define reference target budgets
2) compute best function value achieved by a reference algorithm

$\Rightarrow$ set of target function values
Example for ECDFs

Empirical cumulative distribution functions (ECDFs) of running lengths (left) and function values (right)

\[ \log_{10} \frac{N_{\text{evals}}}{D} \quad \log_{10}(f_{\text{best}}) \]
Overview results 2012

Results from BROR-2012
Results of 2012 (20-D)

- Proportion of functions
- log10 of (# f-evals / dimension)
Results of 2010 (20-D)
Results of 2009 (20-D)
Empirical Cumulative Distribution Functions

- recover a single convergence graph (and generalize)
- can aggregate over any set of functions and target values

they display a set of run lengths or runtimes (RT)

- for RT on a single problem (function & target value) allow to estimate any statistics of interest from them, like median, expectation (ERT),… in a meaningful way

- AKA data profile [Moré&Wild 2009]

- Performance profile [Dolan&Moré 2002]: ECDFs of run lengths divided by the smallest observed run length
Different Displays of Runtimes
Scaling Behaviour with Dimension

13 Sharp ridge

- slanted grid lines: quadratic scaling
- horizontal lines: linear scaling
- **light brown**: artificial best 2009

\[ \log_{10}(\text{#fevals/dimension}) \]

\[ \log_{10}(f_{\text{target}}) \]

\[ \text{dimension} \]
Example: Scaling Behaviour

- slanted grid lines: quadratic scaling
- horizontal lines: linear scaling
- light brown: artificial best 2009

Experiments in >40-D are more often than not virtually superfluous
ERT scatter plots, all dimensions & targets

- estimated Expected Run Time (ERT), two algorithms
- 2-10 D: first algorithm "dominates"
- 20 & 40 D: second algorithm "dominates"
ERT scatter plots, all dimensions & targets
### 6 Attractive sector

<table>
<thead>
<tr>
<th>Function</th>
<th>ERT\textsubscript{best}/D</th>
<th>(\Delta\text{f}_{\text{target}})</th>
<th>1e+03</th>
<th>1e+02</th>
<th>1e+01</th>
<th>1e+00</th>
<th>1e+01</th>
<th>1e+02</th>
<th>1e+03</th>
<th>1e+04</th>
<th>1e+05</th>
<th>1e+07</th>
<th>(\Delta\text{f}_{\text{target}})</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMalGaM IDEA</td>
<td>26</td>
<td>19</td>
<td>22</td>
<td>22</td>
<td>21</td>
<td>22</td>
<td>22</td>
<td>21</td>
<td>22</td>
<td>22</td>
<td>22</td>
<td>22</td>
<td>AMalGaM IDEA [4]</td>
</tr>
<tr>
<td>avg NEWUOA</td>
<td>2.3</td>
<td>1.1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>avg NEWUOA [31]</td>
</tr>
<tr>
<td>BayEDA+C</td>
<td>46</td>
<td>60e+0/90e5</td>
<td>4.7</td>
<td>4.7</td>
<td>4.7</td>
<td>4.7</td>
<td>4.7</td>
<td>4.7</td>
<td>4.7</td>
<td>4.7</td>
<td>4.7</td>
<td>4.7</td>
<td>BayEDA+C [10]</td>
</tr>
<tr>
<td>BFS5</td>
<td>2.2</td>
<td>2.7</td>
<td>3.6</td>
<td>4.7</td>
<td>4.7</td>
<td>4.7</td>
<td>4.7</td>
<td>4.7</td>
<td>4.7</td>
<td>4.7</td>
<td>4.7</td>
<td>4.7</td>
<td>BFS5 [30]</td>
</tr>
<tr>
<td>Cauchy EDA</td>
<td>6200</td>
<td>1500</td>
<td>1e3</td>
<td>1700</td>
<td>1e3</td>
<td>1700</td>
<td>1e3</td>
<td>1700</td>
<td>1e3</td>
<td>1700</td>
<td>1e3</td>
<td>1700</td>
<td>Cauchy EDA [24]</td>
</tr>
<tr>
<td>BIPOP-CMA-ES</td>
<td>2.9</td>
<td>2.2</td>
<td>1.5</td>
<td>1.7</td>
<td>1.6</td>
<td>1.6</td>
<td>1.6</td>
<td>1.6</td>
<td>1.6</td>
<td>1.6</td>
<td>1.6</td>
<td>1.6</td>
<td>BIPOP-CMA-ES [15]</td>
</tr>
<tr>
<td>(1+1)-CMA-ES</td>
<td>1.0</td>
<td>4.5</td>
<td>13</td>
<td>180</td>
<td>1200</td>
<td>13e-1/1e4</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>(1+1)-CMA-ES [9]</td>
</tr>
<tr>
<td>DASA</td>
<td>12</td>
<td>6.8</td>
<td>9.9</td>
<td>19</td>
<td>25</td>
<td>33</td>
<td>49</td>
<td>68</td>
<td>63</td>
<td>74</td>
<td>.</td>
<td>DASA [19]</td>
<td></td>
</tr>
<tr>
<td>DEPSO</td>
<td>11</td>
<td>7.5</td>
<td>12</td>
<td>64</td>
<td>15e-1/2e3</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>DEPSO [12]</td>
<td></td>
</tr>
<tr>
<td>DIRECT</td>
<td>18</td>
<td>31</td>
<td>40e+0/5e3</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>DIRECT [25]</td>
</tr>
<tr>
<td>EDA-PSO</td>
<td>27</td>
<td>46</td>
<td>40</td>
<td>45</td>
<td>44</td>
<td>44</td>
<td>44</td>
<td>44</td>
<td>44</td>
<td>44</td>
<td>44</td>
<td>44</td>
<td>EDA-PSO [6]</td>
</tr>
<tr>
<td>full NEWUOA</td>
<td>5.1</td>
<td>1.9</td>
<td>1.5</td>
<td>1.4</td>
<td>1.4</td>
<td>1.4</td>
<td>1.4</td>
<td>1.4</td>
<td>1.4</td>
<td>1.4</td>
<td>1.4</td>
<td>1.4</td>
<td>full NEWUOA [31]</td>
</tr>
<tr>
<td>G3-PCX</td>
<td>4.1</td>
<td>4.4</td>
<td>1.4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>G3-PCX [26]</td>
</tr>
<tr>
<td>simple GA</td>
<td>3.20</td>
<td>130</td>
<td>2e3</td>
<td>1e+0/1e5</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>simple GA [22]</td>
</tr>
<tr>
<td>GLOBAL</td>
<td>5.5</td>
<td>2.9</td>
<td>3.6</td>
<td>4.9</td>
<td>8.5</td>
<td>42e-3/2e5</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>GLOBAL [29]</td>
</tr>
<tr>
<td>iAMaLGaM IDEA</td>
<td>5.1</td>
<td>5.6</td>
<td>5.4</td>
<td>6.8</td>
<td>7.1</td>
<td>7.1</td>
<td>7.1</td>
<td>7.1</td>
<td>7.1</td>
<td>7.1</td>
<td>7.1</td>
<td>7.1</td>
<td>iAMaLGaM IDEA [4]</td>
</tr>
<tr>
<td>LSfminbnd</td>
<td>9.3</td>
<td>31</td>
<td>160</td>
<td>790</td>
<td>1100</td>
<td>72e-1/1e4</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>LSfminbnd [28]</td>
</tr>
<tr>
<td>LSetep</td>
<td>140</td>
<td>260</td>
<td>2300</td>
<td>59e+0/1e4</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>LSetep [28]</td>
</tr>
<tr>
<td>MA-LS-Chain</td>
<td>11.3</td>
<td>4.9</td>
<td>7.5</td>
<td>8.0</td>
<td>8</td>
<td>7.7</td>
<td>7.2</td>
<td>6.7</td>
<td>6.5</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>MA-LS-Chain [21]</td>
</tr>
<tr>
<td>MCS (Neum)</td>
<td>1.8</td>
<td>33</td>
<td>42e+0/1e5</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>MCS (Neum) [18]</td>
</tr>
<tr>
<td>NELDER (Hant)</td>
<td>2.2</td>
<td>2.4</td>
<td>2.7</td>
<td>3.3</td>
<td>3.2</td>
<td>3.5</td>
<td>3.5</td>
<td>3.5</td>
<td>4</td>
<td>7.4</td>
<td>7.4</td>
<td>7.4</td>
<td>NELDER (Hant) [16]</td>
</tr>
<tr>
<td>NELDER (Doe)</td>
<td>1.5</td>
<td>2.3</td>
<td>9.1</td>
<td>20</td>
<td>28</td>
<td>65</td>
<td>110</td>
<td>430</td>
<td>46e-5/8e4</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>NELDER (Doe) [6]</td>
</tr>
<tr>
<td>NEWUOA</td>
<td>1.1</td>
<td>1</td>
<td>1</td>
<td>1.3</td>
<td>1.4</td>
<td>1.5</td>
<td>1.6</td>
<td>1.6</td>
<td>1.6</td>
<td>1.7</td>
<td>1.7</td>
<td>1.7</td>
<td>NEWUOA [31]</td>
</tr>
<tr>
<td>(1+1)-ES</td>
<td>2.2</td>
<td>2.2</td>
<td>2.1</td>
<td>2.8</td>
<td>3.9</td>
<td>5.2</td>
<td>6.1</td>
<td>6.1</td>
<td>6.1</td>
<td>6.1</td>
<td>6.1</td>
<td>6.1</td>
<td>(1+1)-ES [1]</td>
</tr>
<tr>
<td>POEMS</td>
<td>89</td>
<td>26</td>
<td>31</td>
<td>37</td>
<td>36</td>
<td>36</td>
<td>36</td>
<td>35</td>
<td>35</td>
<td>35</td>
<td>35</td>
<td>35</td>
<td>POEMS [20]</td>
</tr>
<tr>
<td>PSO</td>
<td>6.1</td>
<td>280</td>
<td>1100</td>
<td>980</td>
<td>820</td>
<td>710</td>
<td>620</td>
<td>570</td>
<td>790</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>PSO [7]</td>
</tr>
<tr>
<td>PSO_Bounds</td>
<td>9.5</td>
<td>45</td>
<td>120</td>
<td>150</td>
<td>140</td>
<td>140</td>
<td>140</td>
<td>180</td>
<td>220</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>PSO_Bounds [8]</td>
</tr>
<tr>
<td>Monte Carlo</td>
<td>2.4e5</td>
<td>18e+1/1e6</td>
<td>76</td>
<td>210</td>
<td>230</td>
<td>810</td>
<td>21e-2/1e4</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>IPop-SEP-CMA-ES</td>
<td>3.2</td>
<td>2.1</td>
<td>1.7</td>
<td>1.9</td>
<td>1.9</td>
<td>1.9</td>
<td>1.9</td>
<td>1.9</td>
<td>1.9</td>
<td>1.9</td>
<td>1.9</td>
<td>1.9</td>
<td>IPop-SEP-CMA-ES [29]</td>
</tr>
<tr>
<td>VNS (Garcia)</td>
<td>5</td>
<td>2.8</td>
<td>1.9</td>
<td>1.9</td>
<td>1.9</td>
<td>1.9</td>
<td>1.9</td>
<td>1.9</td>
<td>1.9</td>
<td>1.9</td>
<td>1.9</td>
<td>1.9</td>
<td>VNS (Garcia) [11]</td>
</tr>
</tbody>
</table>

Table 6: 20-D, running time excess ERT/ERT\textsubscript{best} on \(f_8\), in italics is given the median final function value and the median number of function evaluations to reach this value divided by dimension.
Questions?
Python
• a general-purpose, well-designed, modern high-level programming language

• dynamically-typed, highly object-oriented (not enforced), highly modularized

• for scripting, for programming, for interactive usage

• comes with thousands of packages

• the Python programming language is much better designed than Matlab/Octave

• IPython can replace Matlab/Octave for interactive usage

• (I)Python is free and available on almost every computer
### Popularity of Programming Languages (TIOBE)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>=</td>
<td>C</td>
<td>18.729%</td>
<td>+1.38%</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>=</td>
<td>Java</td>
<td>16.914%</td>
<td>+0.31%</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>↑</td>
<td>Objective-C</td>
<td>10.428%</td>
<td>+2.12%</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>↓</td>
<td>C++</td>
<td>9.198%</td>
<td>-0.63%</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>=</td>
<td>C#</td>
<td>6.119%</td>
<td>-0.70%</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>=</td>
<td>PHP</td>
<td>5.784%</td>
<td>+0.07%</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>=</td>
<td>(Visual) Basic</td>
<td>4.656%</td>
<td>-0.80%</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>=</td>
<td>Python</td>
<td>4.322%</td>
<td>+0.50%</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>=</td>
<td>Perl</td>
<td>2.276%</td>
<td>-0.53%</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td>↑</td>
<td>Ruby</td>
<td>1.670%</td>
<td>+0.22%</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>↓</td>
<td>JavaScript</td>
<td>1.536%</td>
<td>-0.60%</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>=</td>
<td>Visual Basic .NET</td>
<td>1.131%</td>
<td>-0.14%</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td>MATLAB</td>
<td>0.563%</td>
<td>0.00%</td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
<td>R</td>
<td>0.480%</td>
<td></td>
</tr>
</tbody>
</table>

Nikolaus Hansen

Performance Evaluation of Anytime Black Box Optimizers
BBOB with COCO in practice (for dummies)

COCO (COmparing Continuous Optimizers): a tool for black-box optimization benchmarking
This is the COCO download page.

Last release: **30/05/2012 v11.06**

**BBOB (5MB)** is all that is needed to run the benchmarking experiments and compile a template paper (gathering post-processed results).

**BBOB (35MB)** contains all files, as listed below.

- **CODE:**
  - tar code in Matlab/Octave to run experiments
  - tar code in C to run experiments
  - tar code in Java to run experiments
  - tar code in Python to run experiments and post-processing and latex templates (3MB)
  - tar R package to run experiments

- **DOCS:**
  - pdf description of experimental procedure
  - pdf (12MB) noiseless functions documentation with figures
  - pdf noiseless functions documentation, version without figures
  - pdf (19MB) noisy function documentation with figures
  - pdf noisy function documentation, version without figures
  - pdf software user documentation
  - html online post-processing package documentation

**BUGS for older versions:**

- Bugs in version 11.05:
BBOB in practice
Matlab script (*exampleexperimwnt.m*):

```matlab
dimensions = [2, 3, 5, 10, 20, 40]; % small dimensions first, for CPU reasons.
functions = benchmarks('FunctionIndices'); % or benchmarksnoisy(...)
instances = [1:5, 31:40]; % 15 function instances

for dim = dimensions
    for ifun = functions
        for iinstance = instances
            fgeneric('initialize', ifun, iinstance, datapath, opt);
            MY_OPTIMIZER('fgeneric', dim, fgeneric('ftarget'), eval(maxfunevals) - f);
            disp(sprintf([' f%0d in %0d-D, instance %0d: FEs=%0d with %0d restarts, fbest:
                          fgeneric('finalize');
            end
            disp([' date and time: ' num2str(clock, ' %0f')]);
            end
            disp(sprintf('---- dimension %0d-D done ----', dim));
        end
    end
end
```
Running the experiment at an OS shell:

```
$ nohup nice octave < exampleexperiment.m > output.txt &
$ less output.txt
```

GNU Octave, version 3.6.3
Copyright (C) 2012 John W. Eaton and others.
This is free software; see the source code for copying conditions.
[...
Read http://www.octave.org/bugs.html to learn how to submit bug reports.

For information about changes from previous versions, type `news'.

```
f1 in 2-D, instance 1: FEs=242, fbest-ftarget=-8.1485e-10, elapsed time [h]: 0.00
f1 in 2-D, instance 2: FEs=278, fbest-ftarget=-6.0931e-09, elapsed time [h]: 0.00
f1 in 2-D, instance 3: FEs=242, fbest-ftarget=-9.2281e-09, elapsed time [h]: 0.00
f1 in 2-D, instance 4: FEs=302, fbest-ftarget=-4.5997e-09, elapsed time [h]: 0.00
f1 in 2-D, instance 5: FEs=230, fbest-ftarget=-9.8350e-09, elapsed time [h]: 0.00
f1 in 2-D, instance 6: FEs=284, fbest-ftarget=-7.0829e-09, elapsed time [h]: 0.00
f1 in 2-D, instance 7: FEs=278, fbest-ftarget=-6.5999e-09, elapsed time [h]: 0.00
f1 in 2-D, instance 8: FEs=272, fbest-ftarget=-8.7044e-09, elapsed time [h]: 0.00
f1 in 2-D, instance 9: FEs=248, fbest-ftarget=-2.6316e-09, elapsed time [h]: 0.00
f1 in 2-D, instance 10: FEs=302, fbest-ftarget=-4.6779e-09, elapsed time [h]: 0.00
f1 in 2-D, instance 11: FEs=272, fbest-ftarget=-5.1499e-09, elapsed time [h]: 0.00
f1 in 2-D, instance 12: FEs=260, fbest-ftarget=-8.8635e-09, elapsed time [h]: 0.00
f1 in 2-D, instance 13: FEs=266, fbest-ftarget=-2.5484e-09, elapsed time [h]: 0.00
f1 in 2-D, instance 14: FEs=218, fbest-ftarget=-9.9961e-09, elapsed time [h]: 0.00
f1 in 2-D, instance 15: FEs=248, fbest-ftarget=-7.5842e-09, elapsed time [h]: 0.00
date and time: 2013 3 29 19 59 26
f2 in 2-D, instance 1: FEs=824, fbest-ftarget=-7.0206e-09, elapsed time [h]: 0.00
f2 in 2-D, instance 2: FEs=572, fbest-ftarget=-9.2822e-09, elapsed time [h]: 0.00
[...]
```
Post-processing at the OS shell:

$ python codepath/bbob_pproc/rungeneric.py datapath

[...]

$ pdflatex templateACMArticle.tex

[...]
Post-processing at the OS shell:

```
$ python codepath/bbob_pproc/rungeneric.py datapath
```

```
$ pdflatex templateACMarticle.tex
```

---

**BBOB in practice**

---

**Post-processing at the OS shell:**

```
$ python codepath/bbob_pproc/rungeneric.py datapath
```

```
$ pdflatex templateACMarticle.tex
```

---
Performance Evaluation of Anytime Black Box Optimizers

Nikolaus Hansen
Performance Evaluation of Anytime Black Box Optimizers

Nikolaus Hansen
Test Functions
Test Functions

Test functions

- define the "scientific question"
  the relevance can hardly be overestimated
- should represent "reality"
- are often too simple?
  remind separability
- a number of testbeds are around
1 Separable functions
1.1 Sphere Function
1.2 Ellipsoidal Function
1.3 Rastrigin Function
1.4 Büche-Rastrigin Function
1.5 Linear Slope

2 Functions with low or moderate conditioning
2.6 Attractive Sector Function
2.7 Step Ellipsoidal Function
2.8 Rosenbrock Function, original
2.9 Rosenbrock Function, rotated

3 Functions with high conditioning and unimodal
3.10 Ellipsoidal Function
3.11 Discus Function
3.12 Bent Cigar Function
3.13 Sharp Ridge Function
3.14 Different Powers Function

4 Multi-modal functions with adequate global structure
4.15 Rastrigin Function
4.16 Weierstrass Function
4.17 Schaffers F7 Function
4.18 Schaffers F7 Function, moderately ill-conditioned
4.19 Composite Griewank-Rosenbrock Function F8F2

5 Multi-modal functions with weak global structure
5.20 Schwefel Function
5.21 Gallagher’s Gaussian 101-me Peaks Function
5.22 Gallagher’s Gaussian 21-hi Peaks Function
5.23 Katsuura Function
5.24 Lunacek bi-Rastrigin Function
Questions?