Benchmarking and analyzing iterative optimization heuristics with IOHprofiler

Wang, H.; Doerr, C.; Shir, O.M.; Bäck, T.H.W.

Benchmarking and Analyzing Iterative Optimization Heuristics with IOHprofiler

Hao Wang[1], Diederick Vermetten[3], Carola Doerr[1], Ofer M. Shir[2], Thomas Bäck[3]

[1]Sorbonne Université, CNRS, LIP6, Paris, France
[2]Tel-Hai College and Migal Institute, Upper Galilee, Israel
[3]LIACS, Leiden University, The Netherlands

http://gecco-2020.sigevo.org/

GECCO '20 Companion, Cancun, Mexico

https://doi.org/10.1145/3377929.3389879

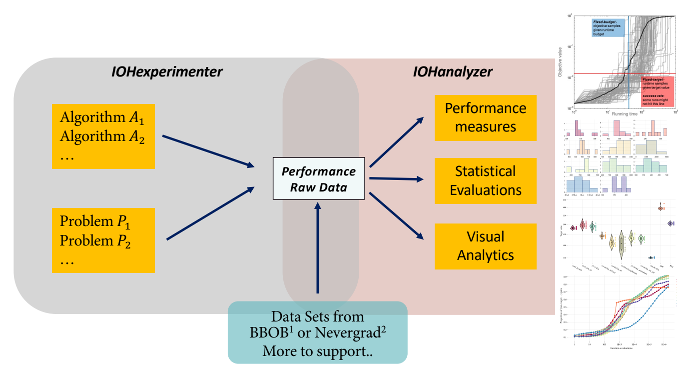
1

INTRODUCTION

Iterative Optimization Heuristic Profiler

- For **benchmarking** and **analyzing** iterative optimization heuristics (IOHs)
- **IOHprofiler** = IOHexperimenter + IOHanalyzer + IOHdata + IOHalgorithm
- **IOHexperimenter** – standardized experimental environment
- **IOHanalyzer** – ready-to-use, interactive analyses of experimental data
- **IOHdata** – a public, cloud-based data repository [in progress]
- **IOHalgorithm** – efficient implementation of some selected IOHs [in progress]

IOHprofiler: Workflow



[1]https://github.com/numbbo/coco
[2]https://github.com/facebookresearch/nevergrad

IOHprofiler - Philosophy

User to choose

- ✓ Which benchmark problems and algorithms to use
  - ✓ Easy to add new benchmark functions and algorithms
- ✓ Which performance measures to use
  - ✓ Wide range of performance statistics:
    - ✓ Fixed-target runtimes (+distributions)
    - ✓ Fixed-budget results (+distributions)
    - ✓ Probabilities of success, ECDF curves, etc.
- ✓ Which statistical procedures to use [in progress]
  - ✓ Kolmogorov-Smirnov, Friedman Test, Deep Statistical Comparison[1], etc.
- ✓ How to aggregate results
  - ✓ Over target values, functions, and dimensions..
  - ✓ Highly interactive

[1] Eftimov, Tome, Peter Korošec, and Barbara Koroušić Seljak. "A novel approach to statistical comparison of meta-heuristic stochastic optimization algorithms using deep statistics." *Information Sciences* 417 (2017): 186-215.

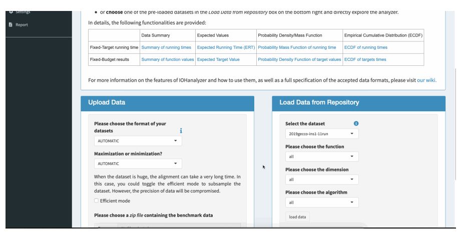
IOHexperimenter

- A general-purpose benchmarking framework – instantiates to discrete, continuous, and mixed search spaces [in progress]
- Pre-installed problem sets
  - **Pseudo-Boolean Optimization (PBO)** Suite[1]
  - Integration of **Black-Box Optimization Benchmarking (BBOB)** Suite
- Easy to add / define new problems and suites
- Backbone implemented in C++ for speed; Python/R interfaces available
- Automatic data logging
  - Running time, best-so-far objective value, etc.
  - **User-chosen dynamic parameters of IOHs**
  - IOHexperimenter data format

[1] https://iohprofiler.github.io/Suites/PBO/

IOHanalyzer

- A web-based interface to analyze and visualize the performance of IOHs
- Interactive plotting
- Statistical evaluation
- Built on R and C++
- Data formats:
  - IOHexperimenter
  - BBOB
  - Facebook's Nevergrad
  - Generic two-column formats*



IOHprofiler - Availability

- ✓ The source code:
  - ✓ **IOHexperimenter:** https://github.com/IOHprofiler/IOHexperimenter
  - ✓ **IOHanalyzer:** https://github.com/IOHprofiler/IOHanalyzer
  - ✓ **IOHdata**: https://github.com/IOHprofiler/IOHdata
  - ✓ **Reference algorithms:** https://github.com/IOHprofiler/IOHalgorithm
- ✓ **IOHanalyzer** is read-to-use online: http://iohprofiler.liacs.nl/
- ✓ **Wiki page**: https://iohprofiler.github.io/
- ✓ **Documentation:** https://arxiv.org/abs/2007.03953
- ✓ Exercises and data sets of this tutorial: https://github.com/IOHprofiler/GECCO20-Tutorial

2

BENCHMARKING

Benchmarking

- Run each pair of (A, f, d) for several times independently
  - Estimate the empirical distribution function:

$$\{v_1, v_2, \ldots, v_r\} \rightarrow \hat{F}_V \qquad \{t_1, t_2, \ldots, t_r\} \rightarrow \hat{F}_T$$

  - Stopping Criteria? e.g., a budget on the function evaluation, or a target function value to hit
  - How to determine the number of runs/repetitions r? – to obtain enough data/evidence
  - How to determine which test functions to use?

IOHexperimenter

- Implemented in **C++** (1/14), non-trivial architecture (DPs)
- Effective wrappers in R and Python
- **Pseudo-Boolean Optimization**[1] (**PBO**) suite: $f: \{0,1\}^d \rightarrow \mathbb{R}$
  - Currently, **25** built-in combinatorial test-functions
  - OneMax, LeadingOnes feat. W-model[2]
  - Ising models, N-Queens problem, Maximum Independent Set, Low Autocorrelation Binary Sequences, Concatenated Trap, NK landscape
  - Extendible to any test functions following the interface

[1] Carola Doerr, Furong Ye, Naama Horesh, Hao Wang, Ofer M. Shir, and Thomas Bäck. "Benchmarking discrete optimization heuristics with IOHprofiler." *Applied Soft Computing* 88 (2020): 106027.

[2] Weise, Thomas, and Zijun Wu. "Difficult features of combinatorial optimization problems and the tunable W-model benchmark problem for simulating them." In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 1769-1776. 2018.

IOHexperimenter

- Integration of **Black-Box Optimization Benchmarking** (**BBOB**) suite[1]: $f: S \subseteq \mathbb{R}^d \rightarrow \mathbb{R}$
  - **24** built-in continuous test functions
- **12** optimization algorithms implemented in C++
  - https://github.com/IOHprofiler/IOHalgorithm
  - **Python** and **R** interface to those algorithm [in progress]
- Benchmark data on those 12 algorithms
  - https://github.com/IOHprofiler/IOHdata
  - All 25 test problems
  - **11** runs of each algorithms
  - $d \in \{16, 100, 625\}$

[1] https://github.com/numbbo/coco

12 Reference Algorithms

(more to come, your contributions most appreciated!)

- gHC: Greedy Hill Climber: flip 1 bit/iteration
- RLS: Randomized Local Search: flips 1 random bit/iteration
- $(1+\lambda)$ EAs with standard bit mutation
- $(1+\lambda)$ EAs with normalized bit mutation and self-adjusting means [YeDB'19]
- $(1+\lambda)$ EAs with normalized bit mutation and self-adjusting means and variance control [YeDB'19]
- $(1+\lambda)$ EAs with log-normal self-adaptive mutation rates [BäckS96]
- fast GA from [DoerrLMN GECCO'17]
- self-adjusting 2-rate $(1+\lambda)$ EA from [DoerrGWY GECCO'17]
- self-adjusting $(1+(\lambda, \lambda))$ GA from [DoerrD Algorithmica'18]
- (μ, λ) "vanilla"/textbook GA
- UMDA

Strong focus on "textbook" algorithms and recent developments from the theory community
→ your contributions are highly welcome!

Pseudo-Boolean Optimization (PBO) suite

- F1: OneMax
- F2: LeadingOnes
- F3: Harmonic Linear Function
- F4-F10: W-Model[1] on top of OneMax (W-model allows to scale effective dimension, epistasis, neutrality, ruggedness, fitness plateaus, etc.)
- F5-17: W-Model[1] on top of LeadingOnes
- F18: LABS: Low Autocorrelation Binary Sequences
- F19: Ising Model: Ring
- F20: Ising Model: Torus
- F21: Ising Model: Triangular (Isometric 2D Grid)
- F22: MIVS: Maximum Independent Vertex Set
- F23: N-Queens Problem
- F24: Concatenated Trap
- F25: NK landscape

Thousands of combinations are possible, our implementation covers the whole W model

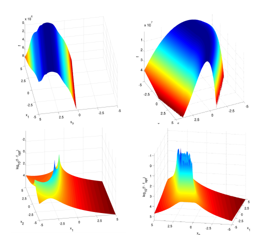
[1] Weise, Thomas, and Zijun Wu. "Difficult features of combinatorial optimization problems and the tunable W-model benchmark problem for simulating them." In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 1769-1776. 2018.

PBO: W-model

- F4 – F17: OneMax transformed by the so-called W-model
  - **Dummy variables:** $(x_1, \ldots, x_d) \rightarrow (x_{i_1}, \ldots x_{i_m}),\ m < d$
    $i_1, \ldots i_m$ are random chosen from $[1..d]$
  - **Neutrality:** $(\underbrace{x_1, \ldots, x_\mu}_{x'_1}, \underbrace{x_{\mu+1}, \ldots, x_{2\mu}}_{x'_2}, \ldots, \underbrace{x_{d-\mu+1}, \ldots, x_d}_{x'_{d/\mu}})$
    x'_i's are the majority vote in each block
  - **Epistasis:** local permutations
  - **Fitness perturbation**

Black-Box Optimization Benchmarking (BBOB)

- Single-objective, continuous, black-box, (noisy) optimization tasks
- Widely used by many scientists and conference, e.g. GECCO Workshop for Real-Parameter Optimization
- 24 test functions
  - 3 unimodal, separable (e.g., sphere)
  - 7 unimodal, non-separable (e.g., bent cigar)
  - 2 multimodal, separable (e.g., Rastrigin)
  - 12 multimodal, non-separable (e.g., Schwefel)
- https://coco.gforge.inria.fr/
- https://github.com/numbbo/coco



IOHexperimenter: Problem Instances

- **PBO** – **multiplicative**, **additive**, and **XOR** shifting; permutation of variables

$$\tilde{f} := af(\sigma(\mathbf{x} \oplus \mathbf{z})) + b$$

- **BBOB** – each test function is modified 'slightly' by:
  - Translation and rotation of the search space[1]

$$\tilde{f} := f(\mathbf{R}(\mathbf{x} + \mathbf{x}_{\text{offset}})) + y_{\text{offset}}$$

  - Small irregularities on the function landscape[1]

[1]Hansen, Nikolaus, Steffen Finck, Raymond Ros, and Anne Auger. "Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions." (2009).

IOHexperimenter – C Interface

- A simple interface..

```
#include "src/IOHprofiler_experimenter.hpp"

void _run_experiment() {
  std::string configName = "./configuration.ini";
  /// An example for PBO suite.
  IOHprofiler_experimenter<int> experimenter(configName,evolutionary_algorithm);

  /// An exmaple for BBOB suite.
  /// IOHprofiler_experimenter<double> experimenter(configName, random_search);
  experimenter._set_independent_runs(10);
  experimenter._run();
}
```


IOHexperimenter – C Interface

- Configuration file

```
[suite]
suite_name = PBO
problem_id = 1-5
instance_id = 1
dimension = 16

[logger]
output_directory = ./
result_folder = Experiment
algorithm_name = (1+1)_EA
algorithm_info = An_EA_algorithm

[observer]
complete_triggers = false
update_triggers = true
number_interval_triggers = 0
number_target_triggers = 0
base_evaluation_triggers = 1
```

Specify which pre-installed suite to use

Select a subset of problems

Control when to register a data record

IOHexperimenter – R interface

- Wrappers enable modern *look-n-feel* in scripting languages
- To install the **R** interface:

```
R> install.packages("IOHexperimenter")
R> library(IOHexperimenter)
```

- Compilation might take a while (we are working on pre-compiled binary packages for the common platform)

IOHexperimenter – R interface

- Create an experiment:

```
R> set.seed(42) # to create reproducible results
R> experimenter <- IOHexperimenter(
+   suite = "PBO",
+   dims = c(16, 100),
+   functions = c(1, 2, 19, 23),
+   instances = 1:5, algorithm.name = 'RLS',
+   data.dir = './data'
+ )
```

- **instances**: 1) translation, and 2) permutation of binary string
- **data.dir**: The path to the folder where (raw) benchmark data are stored

IOHexperimenter – R interface

- Iterate over the test problems:

```
R> problem <- next_problem(experimenter)
R> print(problem)
IOHproblem (Suite PBO: Instance 1 of function OneMax 16D)
```

- And use it for your algorithms:

```
while (!is.null(p)) {
    algorithm(p$dimension, p$obj_func, budget = 1e3 * p$dimension,
              target_hit = p$target_hit)
    p <- next_problem(experimenter)
}
```

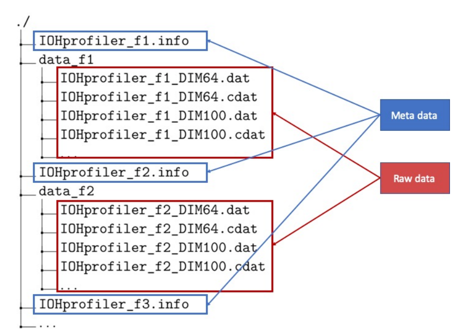
- We will cover the Python interface in the hands-on session!

3

DATA FORMAT

IOHprofiler Data Format

- File structure – text files,
  - *Meta data* – summary of a test problem/function.
  - *Raw data* – running times/function values for each function and dimension.



IOHprofiler Data Format

Meta data – naming convention, e.g., IOHprofiler_f1_OneMax.info

IOHprofiler_f<**function ID**>[_**function name**].info

- Here, <**function ID**> is required. [_**function name**] is optional.
- **function ID** can take either an integer or a string as its value.
- **function name** is only *optional.*

```
suite = 'PBO', funcId = 10, DIM = 100, algId = '(1+1) fGA'
%
data_f10/IOHprofiler_f10_DIM625.dat, 1:1953125|5.59000e+02,
1:1953125|5.59000e+02, 1:1953125|5.59000e+02, 1:1953125|5.54000e+02,
1:1953125|5.59000e+02, 1:1953125|5.64000e+02, 1:1953125|5.54000e+02,
1:1953125|5.59000e+02, 1:1953125|5.49000e+02, 1:1953125|5.54000e+02,
1:1953125|5.49000e+02
suite = 'PBO', funcId = 10, DIM = 625, algId = '(1+1) fGA'
%
data_f10/IOHprofiler_f10_DIM625.dat, 1:1953125|5.59000e+02,
1:1953125|5.59000e+02, 1:1953125|5.59000e+02, 1:1953125|5.54000e+02,
1:1953125|5.59000e+02, 1:1953125|5.64000e+02, 1:1953125|5.54000e+02,
1:1953125|5.59000e+02, 1:1953125|5.49000e+02, 1:1953125|5.54000e+02,
```


IOHprofiler Data Format

Raw data – naming convention, e.g., IOHprofiler_f1_OneMax.info

IOHprofiler_f<**function ID**>_DIM_<**dimension**>.[**t**|**i**|**c**]dat

- Using **IOHexperimenter**, you could produce four types of raw data files: *.dat, *.idat, *.tdat, and *.cdat, which share exactly the same format and only differ in the logging events at which data are recorded.

```
"function evaluation"  "current f(x)" "best-so-far f(x)"  "current af(x)+b" "best af(x)+b" "parameter name"
1  +2.95000e+02  +2.95000e+02  +2.95000e+02  +2.95000e+02  0.000000  ...
2  +2.96000e+02  +2.96000e+02  +2.96000e+02  +2.96000e+02  0.001600  ...
4  +3.07000e+02  +3.07000e+02  +3.07000e+02  +3.07000e+02  0.219200  ...
9  +3.11000e+02  +3.11000e+02  +3.11000e+02  +3.11000e+02  0.006400  ...
12  +3.12000e+02  +3.12000e+02  +3.12000e+02  +3.12000e+02  0.001600  ...
16  +3.16000e+02  +3.16000e+02  +3.16000e+02  +3.16000e+02  0.006400  ...
20  +3.17000e+02  +3.17000e+02  +3.17000e+02  +3.17000e+02  0.001600  ...
23  +3.28000e+02  +3.28000e+02  +3.28000e+02  +3.28000e+02  0.027200  ...
27  +3.39000e+02  +3.39000e+02  +3.39000e+02  +3.39000e+02  0.059200  ...
"function evaluation"  "current f(x)" "best-so-far f(x)"  "current af(x)+b" "best af(x)+b" "parameter name"
1   +3.20000e+02  +3.20000e+02  +3.20000e+02  +3.20000e+02  1.000000  ...
24  +3.44000e+02  +3.44000e+02  +3.44000e+02  +3.44000e+02  2.000000  ...
60  +3.64000e+02  +3.64000e+02  +3.64000e+02  +3.64000e+02  3.000000  ...
"function evaluation"  "current f(x)" "best-so-far f(x)"  "current af(x)+b" "best af(x)+b" "parameter name"
...  ... ... ... ...  ...  ...
```


IOHprofiler Data Format

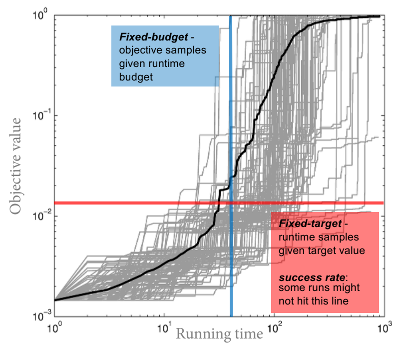
Logging Events

- [**Mandatory**] *Target-based tracking* (*.dat): a record is collected if an improvement on the "best-so-far f(x)" value is observed.
- *Interval tracking* (*.idat): a record is collected every τ-th function evaluation, where the interval τ can be controlled by the user.
- *Time-based tracking* (*.tdat): a record is collected when the user-specified checkpoints on the running time are reached. These checkpoints are evenly spaced in the $\log_{10}$ scale.
- *Complete tracking* (*.cdat): a record is collected for every function evaluation.

4

PERFORMANCE ANALYSES

IOHanalyzer: How to assess Empirical Performance?



30

IOHanalyzer: fixed-target running times

- Running time – random variable
  - Parameterized by ***a target value***

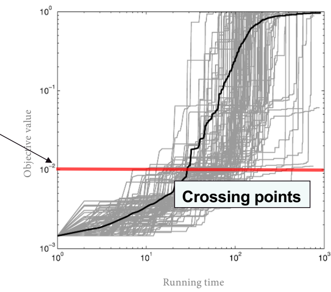
$$T(f_{\text{target}}) \in \mathbb{N}_{>0}$$

- The number of runs – r

$$\{t_1(f_{\text{target}}), \ldots, t_r(f_{\text{target}})\}$$

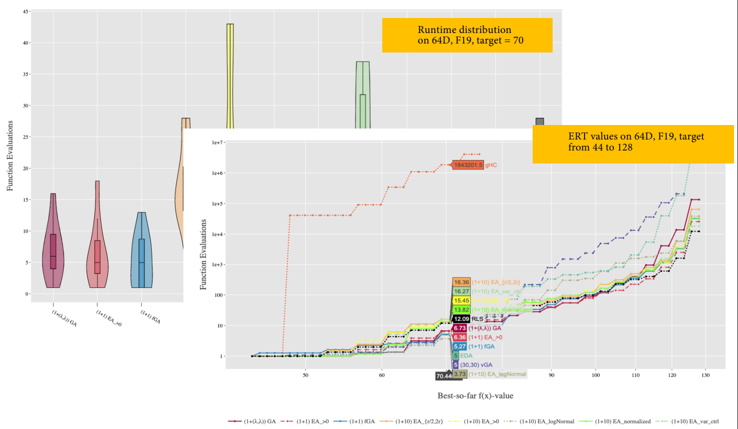
- Successful? $t_i(f_{\text{target}}) < \infty$
- Unsuccessful? $t_i(f_{\text{target}}) = \infty$

- Number of successes $N_{\text{succ}} = \sum_{i=1}^{r} \mathbf{1}(t_i(f_{\text{target}}) < \infty)$



31

IOHanalyzer: fixed-target running times



32

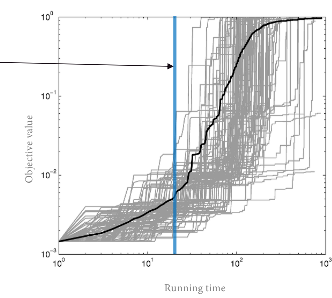
IOHanalyzer: fixed-budget results

- Function value – random variable
  - Parameterized by ***a budget value***

$$V(b) \in \mathbb{R}, b \leq B$$

- The number of runs – r

$$\{v_1(b), \ldots, t_r(b)\}$$



33

IOHanalyzer: Descriptive Statistics

- Sample mean, median, standard deviation…
- Sample quantiles, e.g., 2%, 5%, …, 98%
- Empirical success rate
- **Expected Running Time (ERT)**
- **Empirical Cumulative Distribution Functions (ECDFs)**

IOHanalyzer: Descriptive Statistics

	algId	target	mean	median	sd	2%	5%	10%	25%	50%	75%	90%	95%	98%	ERT	runs	ps
	All	All	All	All	All	All	All	All	All	All	All	All	All	All	All	A	A
1	RLS	1184	1	1	0	1	1	1	1	1	1	1	1	1	1	11	1
2	RLS	1330.22	61	58	16.77	35	35	35	44	58	76	78	78	81	61	11	1
3	RLS	1476.44	198.18	199	23.29	166	166	166	180	199	200	225	225	246	198.18	11	1
4	RLS	1622.66	416.18	405	69.35	335	335	335	366	405	412	523	523	552	416.18	11	1
5	RLS	1768.88	716.64	704	85.05	568	568	568	643	704	755	820	820	864	716.64	11	1
6	RLS	1915.1	1181.64	1137	133.01	1013	1013	1013	1077	1137	1247	1361	1361	1406	1181.64	11	1
7	RLS	2061.32	1906.18	1911	214.23	1430	1430	1430	1760	1911	2088	2110	2110	2122	1906.18	11	1
8	RLS	2207.54	3576.18	3496	622.67	2563	2563	2563	3267	3496	3587	4453	4453	4731	3576.18	11	1
9	RLS	2353.76	14338.64	15885	6119.17	6430	6430	6430	8898	15885	18850	21707	21707	22417	14338.64	11	1
10	RLS	2499.98	42210.14	43953	14424.28	23409	23409	23409	27858	43953	49470	53472	63202	63202	1158281.57	7	0.64

Showing 1 to 10 of 11 entries

Previous 1 2 Next

IOHanalyzer: ECDFs

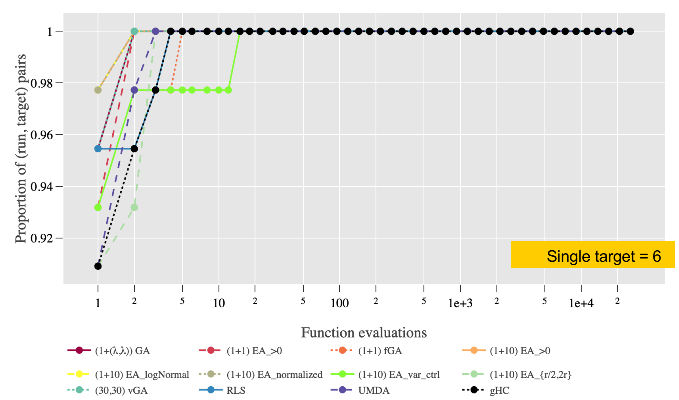
- Aggregate ECDFs
  - Over multiple **targets**: $\mathcal{V} = \{v_1, v_2, \ldots\}$

$$\widehat{F}_T(t\,;\,A, f, d, \mathcal{V}) = \frac{1}{r|\mathcal{V}|}\sum_{v\in\mathcal{V}}\sum_{i=1}^{r}\mathbb{1}(t_i(A, f, d, v) \leq t).$$

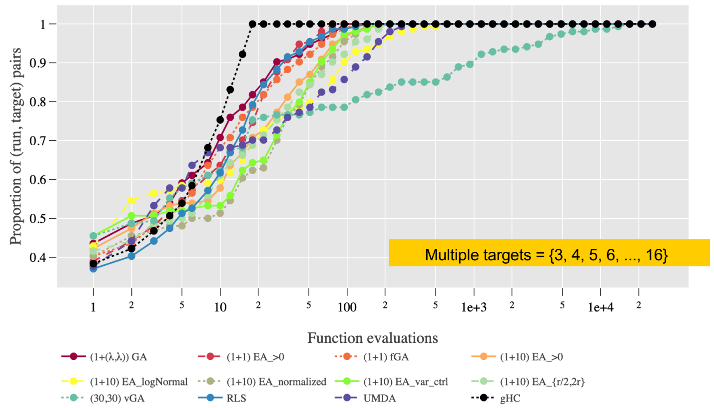
  - Over multiple **functions**: $\mathcal{F} = \{f_1, f_2, \ldots\}$

$$\widehat{F}_T(t\,;\,A, \mathcal{F}, d, \mathcal{V}) = \frac{1}{r|\mathcal{V}||\mathcal{F}|}\sum_{f\in\mathcal{F}}\sum_{v\in\mathcal{V}}\sum_{i=1}^{r}\mathbb{1}(t_i(A, f, d, v) \leq t).$$

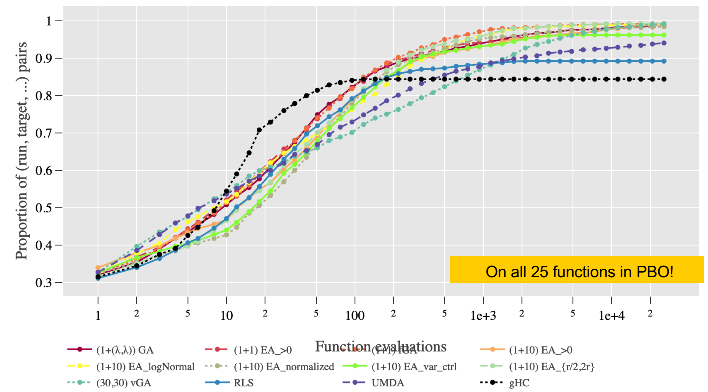
IOHanalyzer: ECDFs



IOHanalyzer: ECDFs



IOHanalyzer: ECDFs



IOHANALYZER DEMO

Hands-on Session

- ✓ The best way to get familiar with a software
- ✓ Online version for this tutorial: http://iohprofiler.liacs.nl/
- ✓ The material of the hands-on session can be found here: http://liacs.leidenuniv.nl/~csnaco/GECCO2020/
- ✓ https://colab.research.google.com/drive/1c_mozIXHQ7YxvlC0UOY9gi6gIrj3vBe6?usp=sharing
- ✓ Some example data set can be download here: https://github.com/IOHprofiler/IOHdata

Hands-on Session

If you want to play with **IOHanalyzer** locally, you need a R installation first..

- ✓ to install R: https://cran.r-project.org/
- ✓ to install IOHanalyzer:

```
R> install.packages('IOHanalyzer')
```

- ✓ to start the web-based server locally:

```
R> library('IOHanalyzer')
R> runServer()
Loading required package: shiny

Listening on http://127.0.0.1:3943
```


IOHprofiler – Examples of Use-Cases

Journal papers:

1. Hao Wang and Diederick Vermetten and Furong Ye and Carola Doerr and Thomas Bäck: IOHanalyzer: Performance Analysis for Iterative Optimization Heuristic. arXiv, arXiv:2007.03953
2. Carola Doerr, Furong Ye, Naama Horesh, Hao Wang, Ofer M. Shir, Thomas Bäck: Benchmarking discrete optimization heuristics with IOHprofiler. Appl. Soft Comput. 88: 106027 (2020)

PPSN full papers:

1. Furong Ye, Hao Wang, Carola Doerr, Thomas Bäck: Benchmarking a (μ+λ) Genetic Algorithm with Configurable Crossover Probability. To appear at PPSN'20

CEC full papers:

1. Furong Ye, Carola Doerr, and Thomas Bäck. Interpolating Local and Global Search by Controlling the Variance of Standard Bit Mutation. CEC'19

GECCO full papers:

1. Diederick Vermetten, Hao Wang, Carola Doerr, Thomas Bäck: Towards Dynamic Algorithm Selection for Numerical Black-Box Optimization: Investigating BBOB as a Use Case. To appear at GECCO'20
2. Diederick Vermetten, Hao Wang, Carola Doerr and Thomas Bäck. Integrated vs. Sequential Approaches for Selecting and Tuning CMA-ES Variants. To appear at GECCO'20
3. Naama Horesh, Thomas Bäck, Ofer M. Shir. Predict or Screen Your Expensive Assay? DoE vs. Surrogates in Experimental Combinatorial Optimization. GECCO'19
4. Nguyen Dang, Carola Doerr. Hyper-Parameter Tuning for the (1+(λ, λ)) GA. GECCO'19
5. Carola Doerr, Furong Ye, Sander van Rijn, Hao Wang, Thomas Bäck. Towards a theory-guided benchmarking suite for discrete black-box optimization heuristics: profiling (1 + λ) EA variants on OneMax and LeadingOnes. GECCO'18

IOHprofiler – Examples of Use-Cases

GECCO workshop papers:

1. Carola Doerr, Furong Ye, Naama Horesh, Hao Wang, Ofer M. Shir, and Thomas Bäck. Benchmarking Discrete Optimization Heuristics with IOHprofiler. GECCO'19
2. Borja Calvo, Ofer M. Shir, Josu Ceberio, Carola Doerr, Hao Wang, Thomas Bäck, and Jose A. Lozano. Bayesian Performance Analysis for Black-Box Optimization Benchmarking. GECCO'19
3. Ivan Ignashov, Arina Buzdalova, Maxim Buzdalov, and Carola Doerr. Illustrating the Trade-Off between Time, Quality, and Success Probability in Heuristic Search. GECCO'19
4. Nathan Buskulic and Carola Doerr. Maximizing Drift is Not Optimal for Solving OneMax. GECCO'19

Usage in teaching:

1. Natural Computing at LIACS, 3rd year bachelor Computer Science
2. Evolutionary Algorithms at LIACS, 1st year master Computer Science

Work in progress

- ✓ Extension of benchmark problems and reference algorithms
- ✓ Constant improvements of the User-Experience (UX)
  - ✓ Efficiency
  - ✓ Report generation [in progress]
  - ✓ Additional visualizations and statistical analysis tools
- ✓ Support for mixed-integer and noisy problems

Suggestions and contributions are very welcome!

Want to get involved?

You can contribute by providing (or suggesting)
- Feedback
- benchmark problems
- Reference algorithms
- Performance statistics
- Ideas how to combine performance analysis with landscape analysis
- Other features that would improve IOHprofiler

… and by simply using IOHprofiler, exporting its output for your presentation/manuscript, and sharing your user-experience!

Acknowledgments

We thank the following colleagues for contributions and feedback on IOHprofiler and/or discussions around the topic of benchmarking IOHs:
- Furong Ye, Diederick Vermetten, Sander van Rijn from Leiden University, NL
- Arina Buzdalova, Maxim Buzdalov, and students from ITMO University, Russia
- Borja Calvo, Josu Ceberio, and Jose A. Lozano from San Sebastián, Spain
- Thomas Weise from Heifei University, China
- Naama Horesh and Assaf Israeli from Migal Institute, Israel
- Dirk Sudholt from Sheffield University, UK
- Anne Auger, Dimo Brockhoff, and Niko Hansen from INRIA, France

We acknowledge financial support from
- Chinese scholarship council (CSC No. 201706310143)
- ANR-11-LABX-0056-LMH,
- Paris Ile-de-France Region
- COST Action CA15140 on Improving Applicability of Nature-Inspired Optimisation by Joining Theory and Practice (ImAppNIO)