

# Benchmarking and analyzing iterative optimization heuristics with

# IOH Profiler



Carola Doerr, Hao Wang, Diederick Vermetten, Thomas Bäck, Jacob de Nobel, Furong Ye



**Universiteit  
Leiden**  
The Netherlands



# Introduction

- Benchmarking is a key component in the field of optimization algorithms
- Need data to judge effectiveness of a new algorithm relative to state-of-the-art
- But benchmarking is not just a number showing algorithm A is better than algorithm B!



# Benchmarking



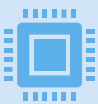
Benchmarking can be used to gain important insights about algorithms and problems



Highlights interplay between problem and algorithm



Differences in performance show potential for new developments

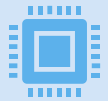


Data-driven studies of algorithm selection, configuration, dynamic switching...

# Requirements for benchmarking



Ease of access



Flexible to the specific requirements of the user

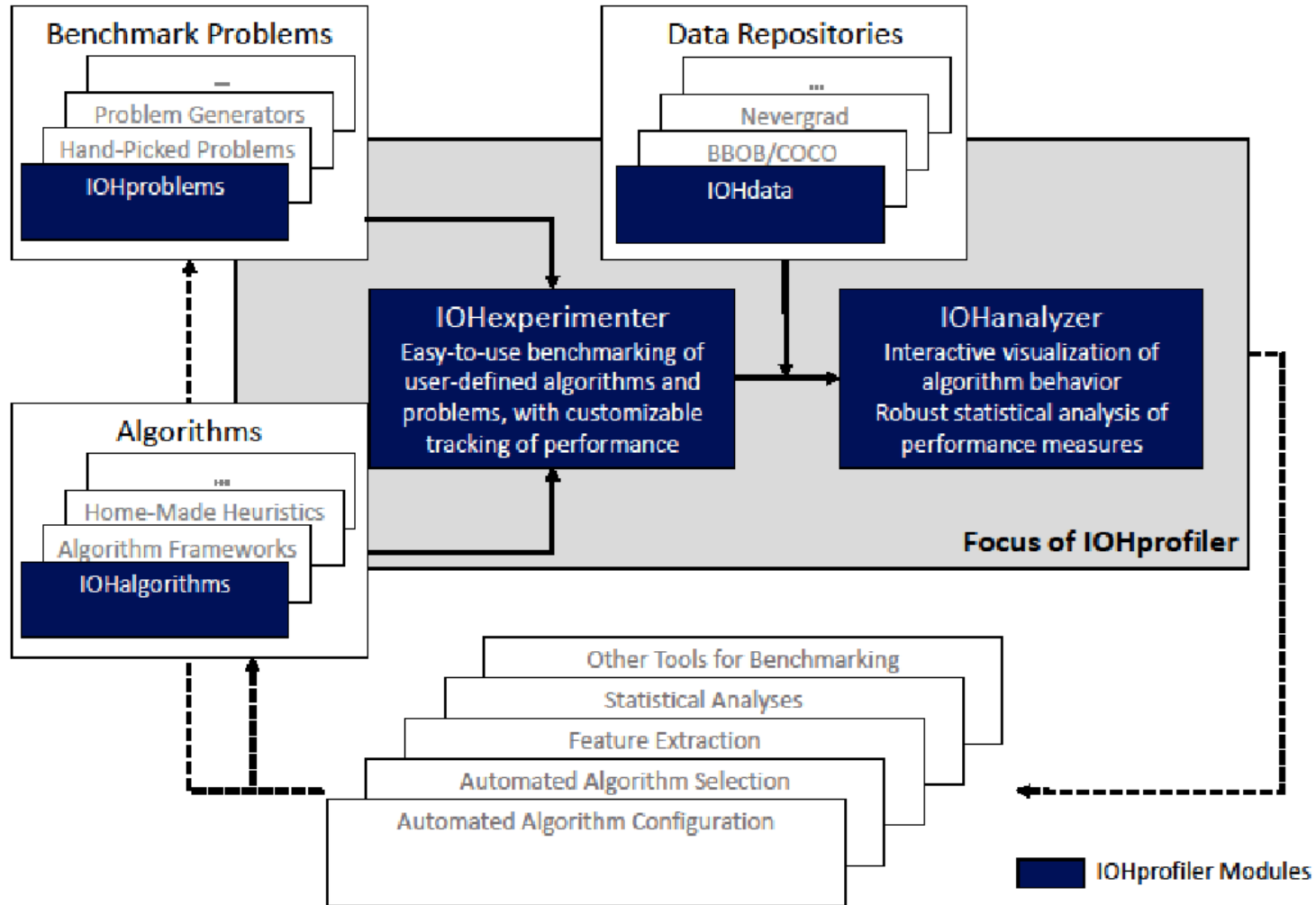


Interoperability with other common tools



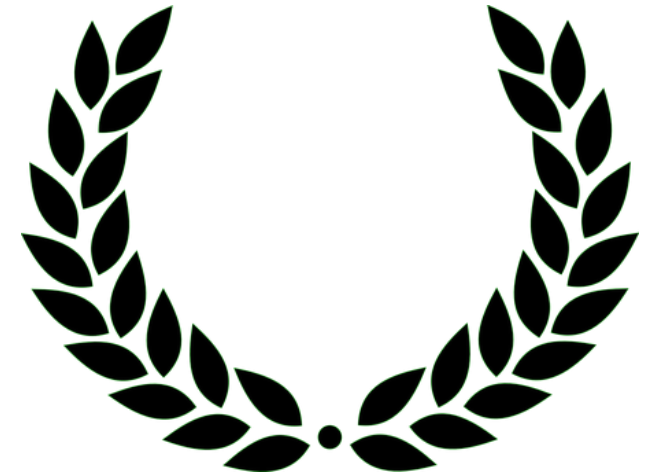
Standardized to ensure reproducibility

# IOHprofiler Architecture Overview



# IOHanalyzer overview

- Performance can not be captured in a single number
- Analysis of performance on a benchmark can be very context-dependent
- To a large extent influenced by the specific requirements of the user
- As such, flexibility is key





DEMO

IOHanalyzer

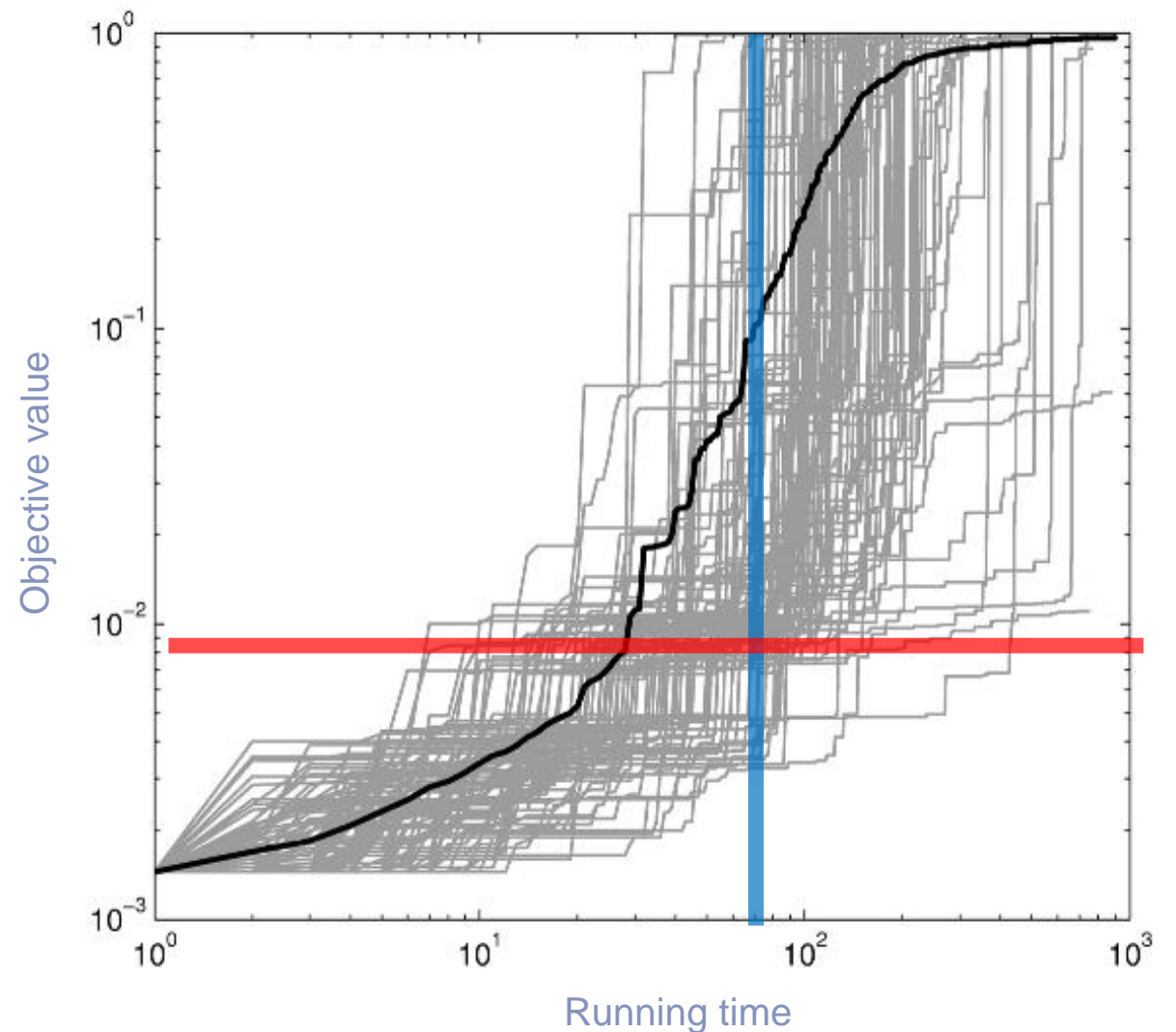
# IOHanalyzer overview

- What perspective to consider?

*Fixed-budget:*  
objective samples given  
runtime budget

*Fixed-target:* runtime samples  
given target value

*Success rate:* some runs might  
not hit this line





# Fixed-target Analysis

- Running time – random variable

- Parameterized by a *target value*

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

- The number of runs –  $r$

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

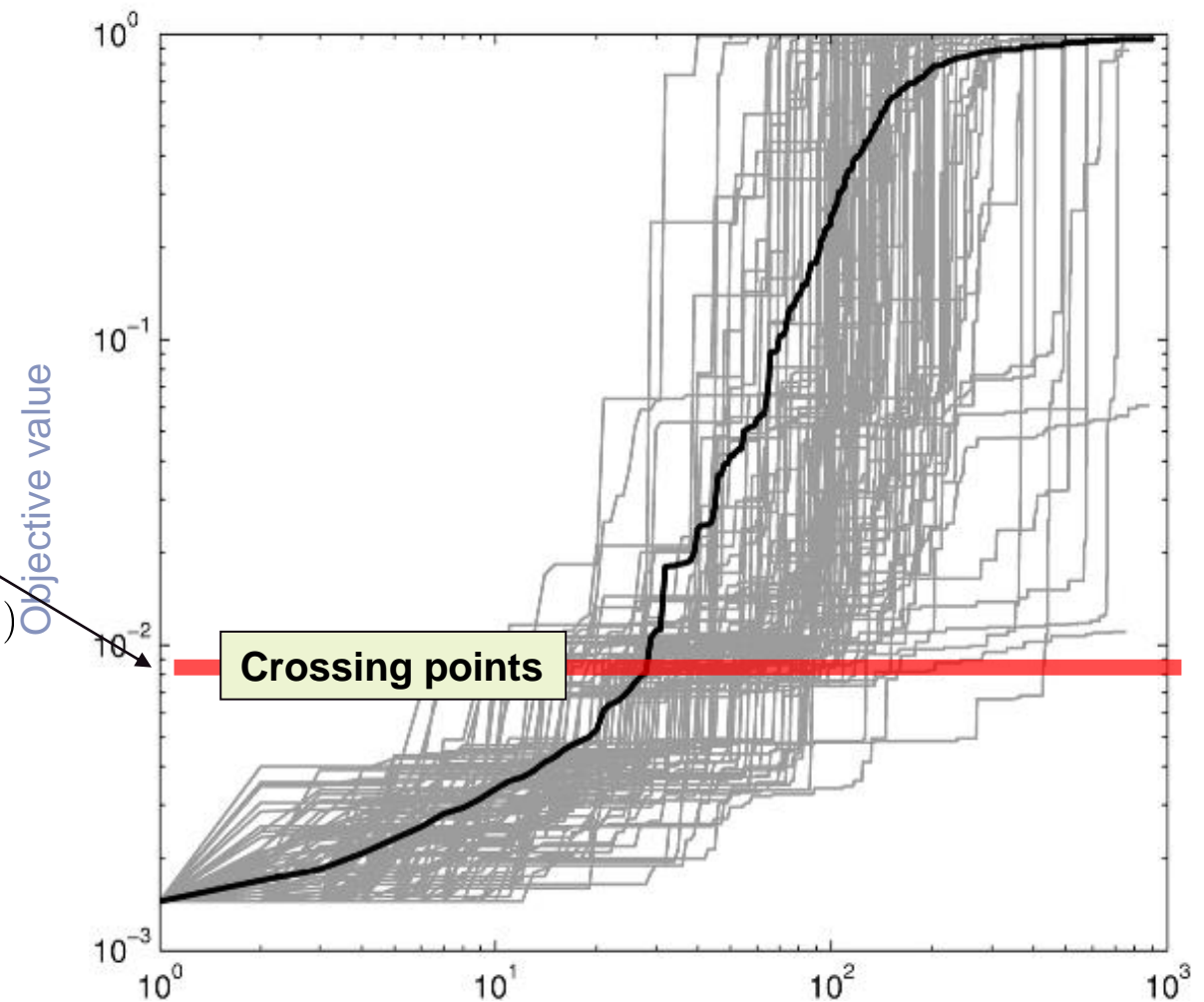
- Successful?  $t_i(f_{\text{target}}) < \infty$

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

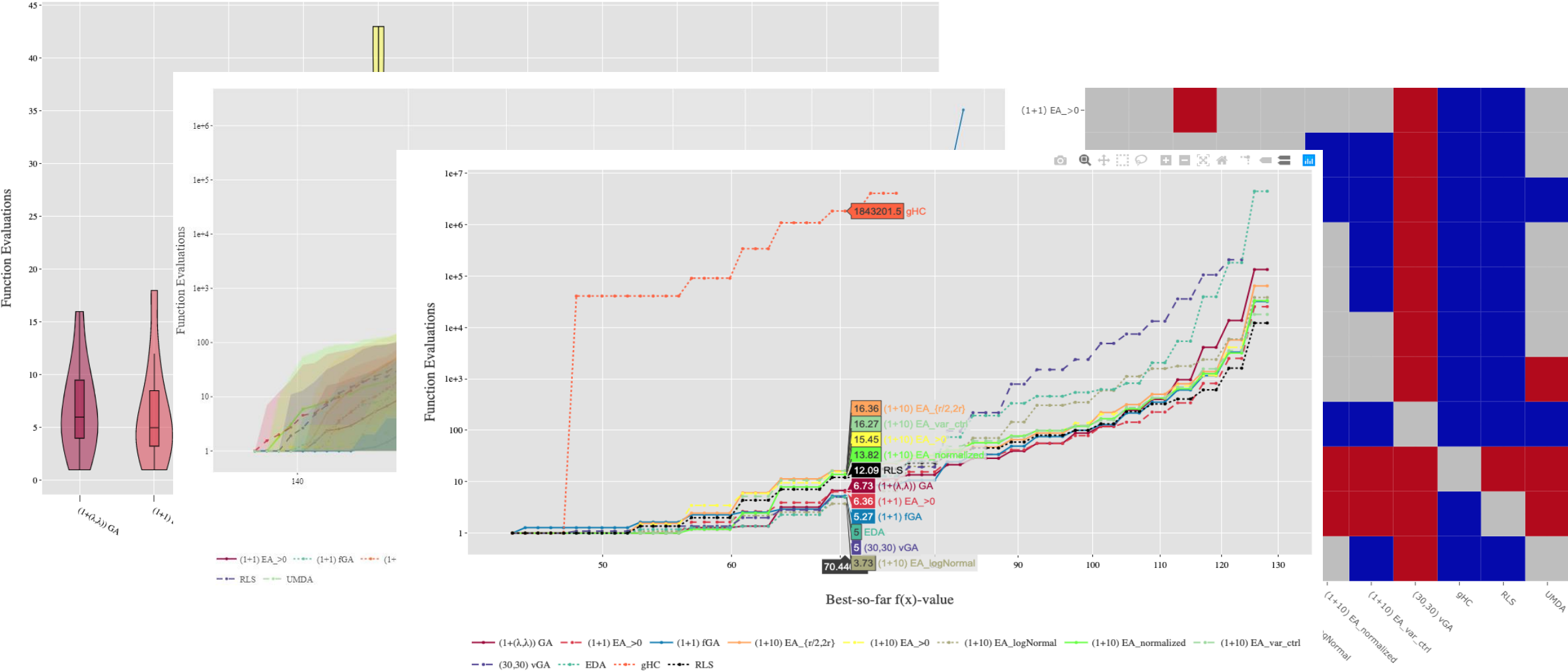
- Number of successes  $N_{\text{succ}} = \sum_{i=1}^r \mathbf{1}(t_i(f_{\text{target}}) < \infty)$

- Performance measures:

- Expected Running Time
- Penalized Average Running Time
- Average Hitting time
- Confidence intervals of hitting time
- ...



# Fixed-target Analysis



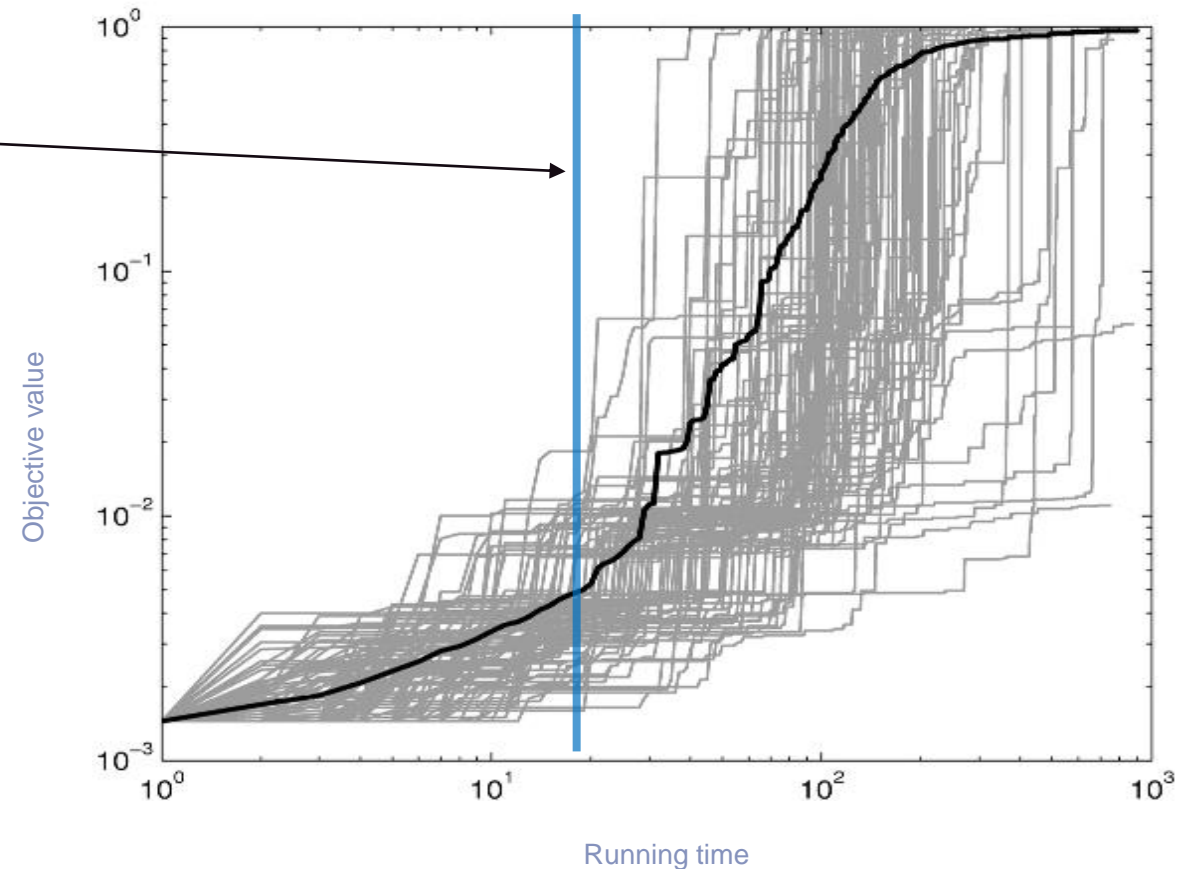
# Fixed-budget analysis

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

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

- The number of runs –  $r$

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



# Single-function Analysis

| ID | runtime              | runs   | mean | median | sd  | 2%   | 5%  | 10% | 25% | 50% | 75% | 90% | 95% | 98% |
|----|----------------------|--------|------|--------|-----|------|-----|-----|-----|-----|-----|-----|-----|-----|
| 1  | (1+1) EA_>0          | 100000 | 11   | 312    | 312 | 0    | 312 | 312 | 312 | 312 | 312 | 312 | 312 | 312 |
| 2  | gHC                  | 100000 | 11   | 312    | 312 | 0    | 312 | 312 | 312 | 312 | 312 | 312 | 312 | 312 |
| 3  | (1+10) EA_{r/2,2r}   | 100000 | 11   | 312    | 312 | 0    | 312 | 312 | 312 | 312 | 312 | 312 | 312 | 312 |
| 4  | (1+10) EA_>0         | 100000 | 11   | 312    | 312 | 0    | 312 | 312 | 312 | 312 | 312 | 312 | 312 | 312 |
| 5  | (1+10) EA_logNormal  | 100000 | 11   | 312    | 312 | 0    | 312 | 312 | 312 | 312 | 312 | 312 | 312 | 312 |
| 6  | (1+10) EA_normalized | 100000 | 11   | 312    | 312 | 0    | 312 | 312 | 312 | 312 | 312 | 312 | 312 | 312 |
| 7  | (1+10) EA_var_ctrl   | 100000 | 11   | 312    | 312 | 0    | 312 | 312 | 312 | 312 | 312 | 312 | 312 | 312 |
| 8  | UMDA                 | 100000 | 11   | 312    | 312 | 0    | 312 | 312 | 312 | 312 | 312 | 312 | 312 | 312 |
| 9  | (1+1) fGA            | 100000 | 11   | 312    | 312 | 0    | 312 | 312 | 312 | 312 | 312 | 312 | 312 | 312 |
| 10 | (30,30) vGA          | 100000 | 11   | 293    | 293 | 3.81 | 287 | 288 | 289 | 291 | 293 | 294 | 297 | 298 |
| 11 | RLS                  | 100000 | 11   | 312    | 312 | 0    | 312 | 312 | 312 | 312 | 312 | 312 | 312 | 312 |

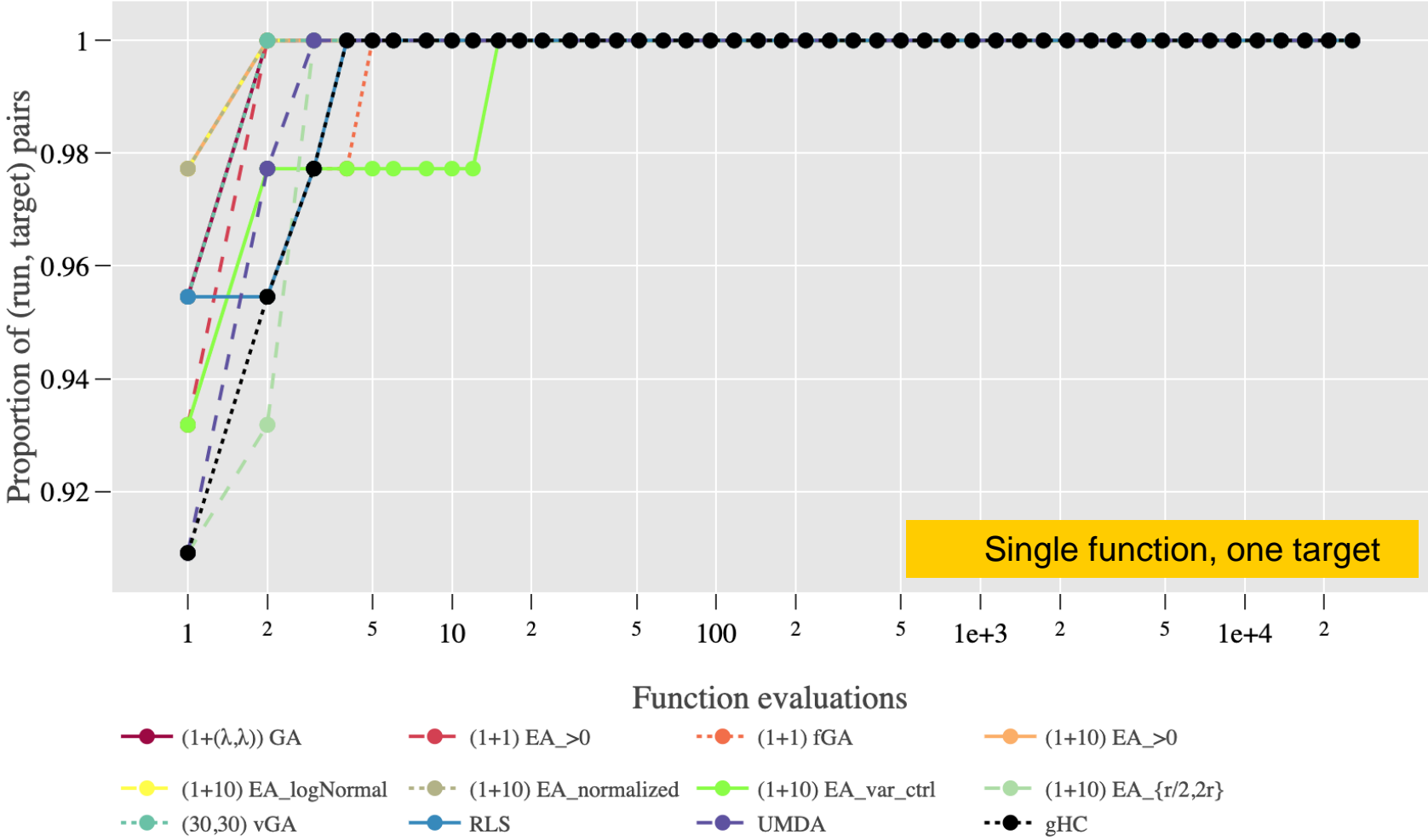
Showing 1 to 11 of 11 entries

Previous

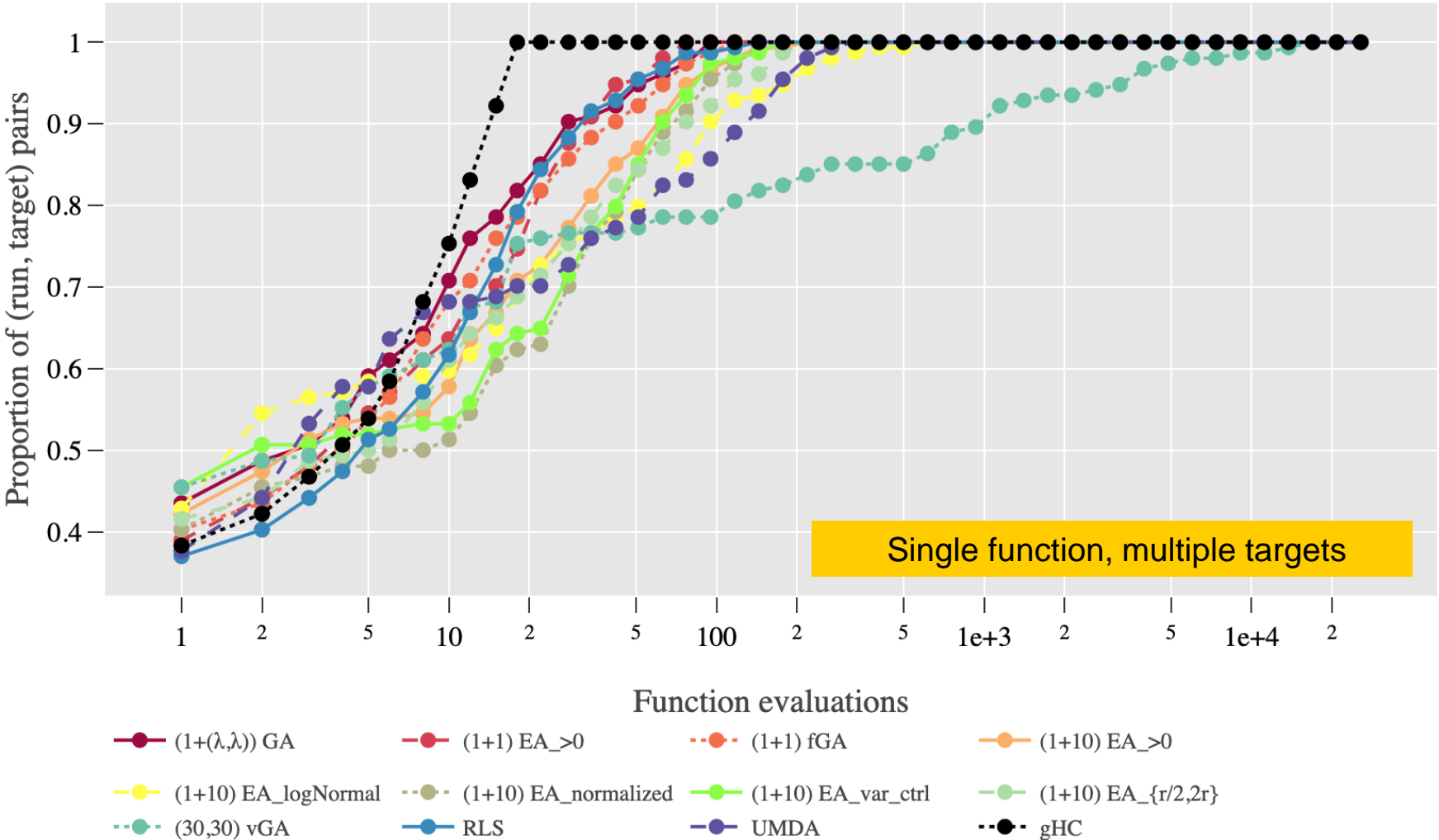
1

Next

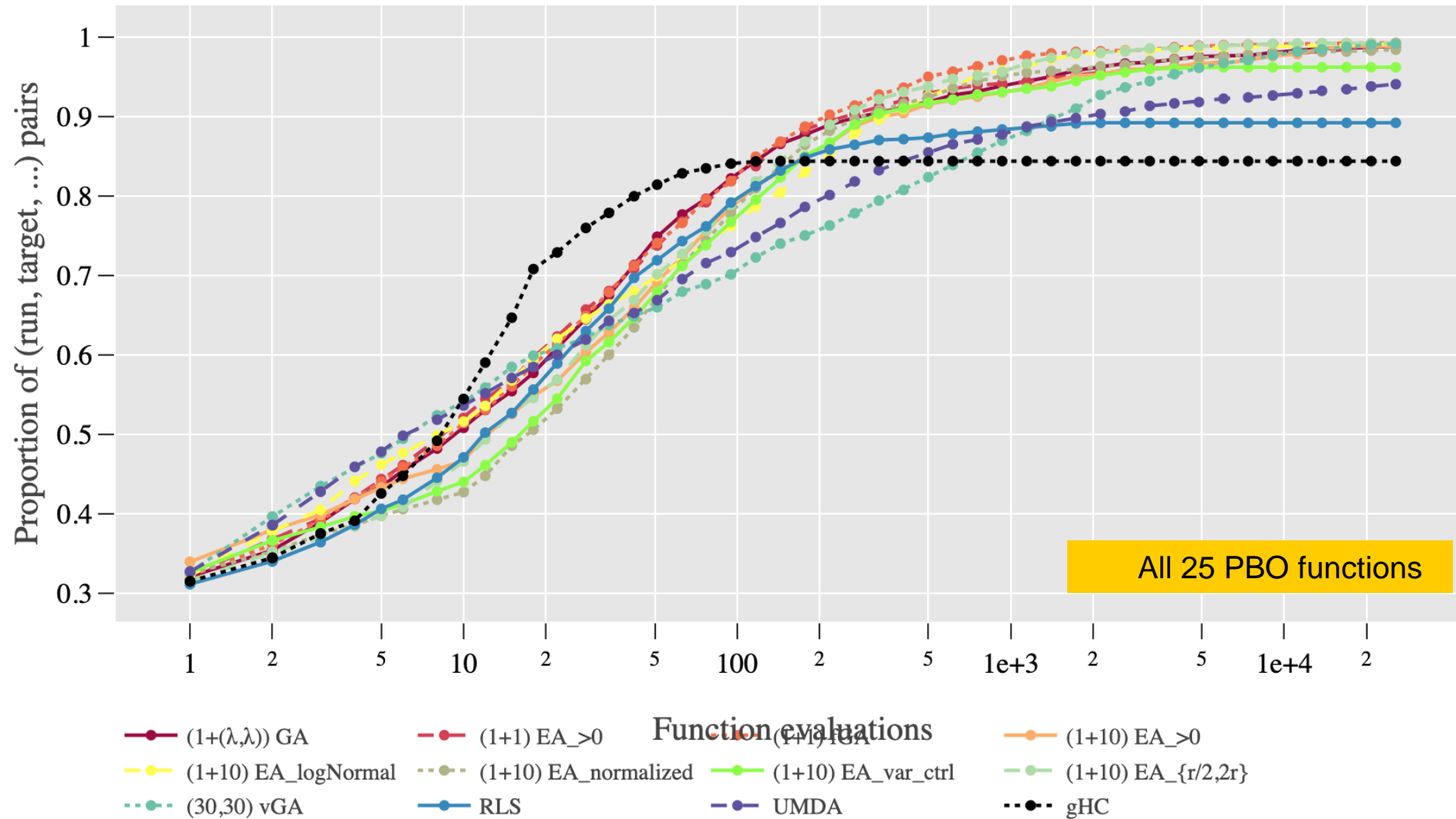
# Empirical Cumulative Density Functions



# Empirical Cumulative Density Functions

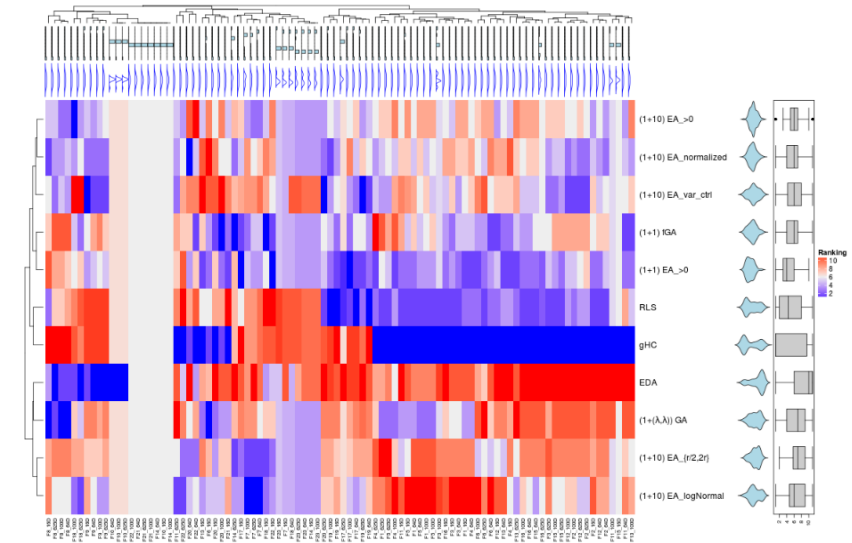


# Empirical Cumulative Density Functions



# Other functionality

- Aggregated rankings
  - Based on better mean per function
  - Glicko2 ranking based on per-run comparisons
- Portfolio contributions (Based on combined ECDF)
- Deep Statistical Comparison [1]
- Parameter Analysis



[1] Eftimov, T., Korošec, P., & Seljak, B. K. (2017). A novel approach to statistical comparison of meta-heuristic stochastic optimization algorithms using deep statistics. *Information Sciences*, 417, 186-215.



# IOHexperimenter overview

- The experimentation module of IOHprofiler
  - Easy to use, customizable and expandable
- Consists of **benchmark problems** and **loggers**
  - Benchmarks of discrete and continuous optimizations are available, and **customized** problems can be easily wrapped.
  - Default logger tracks algorithm performance and dynamic parameters by storing into csv files, and **customized** ones can be easily created for specific tasks.
- Current collaborations and extensions.
  - Algorithm frameworks: modular CMA-ES [2], DE [3] and modular GA [4]
  - More problems: submodular optimization competition [5], MK-Landscapes problems.
  - Other projects: integration with ParadisEO [6] and Nevergrad [7] for automated algorithm configuration.

[2] <https://github.com/IOHprofiler/ModularCMAES>

[3] <https://github.com/Dvermetten/ModDE>

[4] <https://github.com/IOHprofiler/IOHalgorithm>

[5] <https://cs.adelaide.edu.au/~optlog/CompetitionESO2023.php>

[6] <https://github.com/nojhan/paradiseo>

[7] <https://github.com/facebookresearch/nevergrad>



DEMO

IOHexperimenter

# Benchmark Problems

- 25 Pseudo-Boolean optimization problems
  - OneMax
  - LeadingOnes
  - ....
- 24 BBOB continuous optimization problems
  - Sphere
  - ...
- W-model extensions
  - Discrete optimization problems can be configured with customized properties.
- Submodular Problems (including constraints)
- MK-landscape problems
- Star-discrepancy problems (discrete and continuous formulations)
- Wrapper supporting any discrete or continuous problems

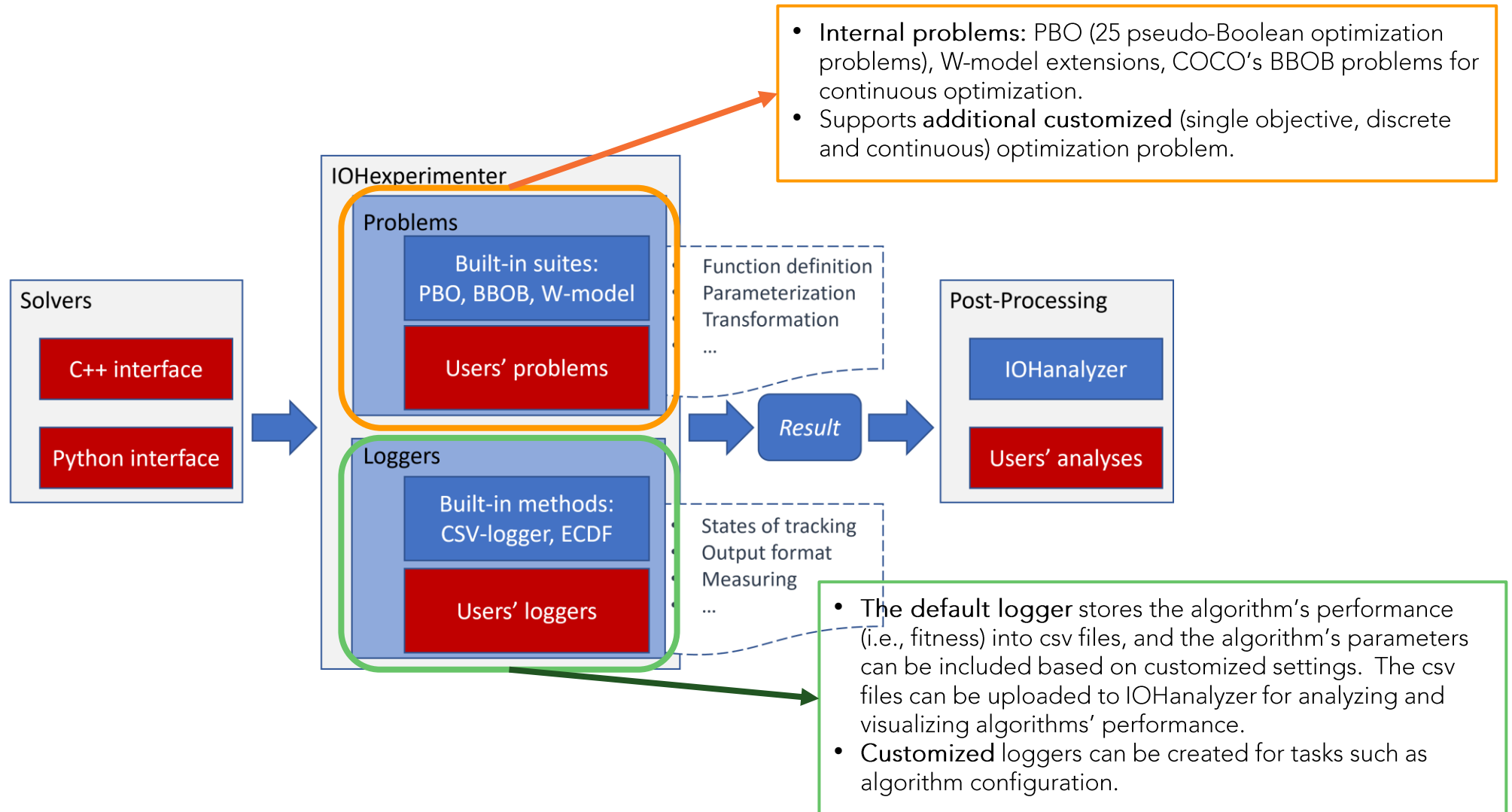


# Logger

- Logger generating data that can be accessible by IOAnalyzer
- Modular design
  - Customized trigger to determine when to store the data
  - Tracking arbitrary parameters of the algorithm
- CSV logger
- In-memory
- Loggers computing statistics on the fly



# Workflow of IOHexperimenter



# What can I do with IOHexperimenter?

- I want to test my algorithms on IOHproblems without modifying much of the current implementation.
  - Include the IOH header files for C++ or import the IOH package for Python and replace your fitness functions with the ones defined in IOHexperimenter.
- I want to check how IOH algorithms perform on my own benchmark problems.
  - Wrapper functions are available for C++ and Python. Define your problems following the examples and apply the existing algorithm implementations for the new problems by replacing the function evaluating fitness.
- I want to compare my algorithms to other state-of-the-art algorithms.
  - Implement your algorithms integrating with the *problem* and *logger* classes of IOHexperimenter and upload the generated data to IOHanalyzer, where data of many other algorithms are accessible for comparison.
- I want to use IOHanalyzer to analyze the performance of my algorithms.
  - Apply the logger of IOHexperimenter to track the performance of your algorithms by attaching the *logger* to the *problem* class. If you plan to test on your own problems, we suggest wrapping the problems into IOHexperimenter, which can be easily done.
- How do I contribute my work to IOHexperimenter?
  - Any contributions, e.g., reporting bugs, suggesting new problems, customized loggers, etc., are appreciated. A PR workflow is available on Github.

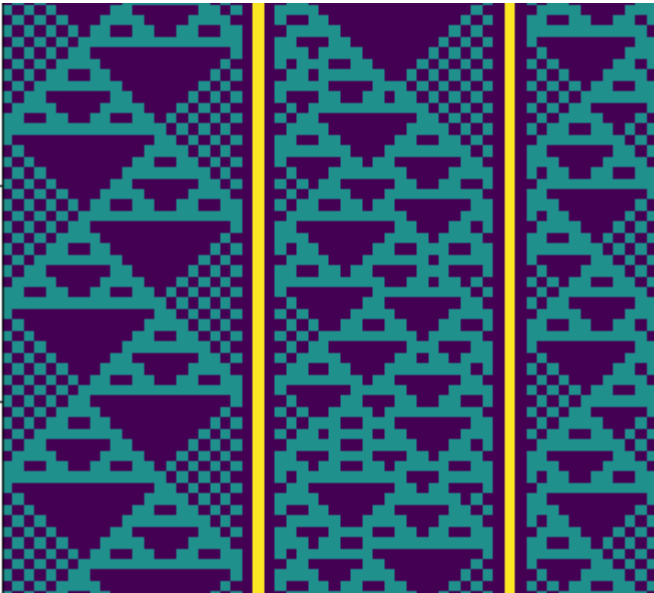
# Teaching

- Benchmarking is a core component of any study of optimization algorithms
- So, should be accessible to students
- Introducing benchmarking concepts to students can be challenging
- IOHexperimenter helps by providing a fixed framework with pre-existing examples
- Interactivity of IOHanalyzer enables non-expert programmers to contribute to understanding of algorithm performance



# Teaching

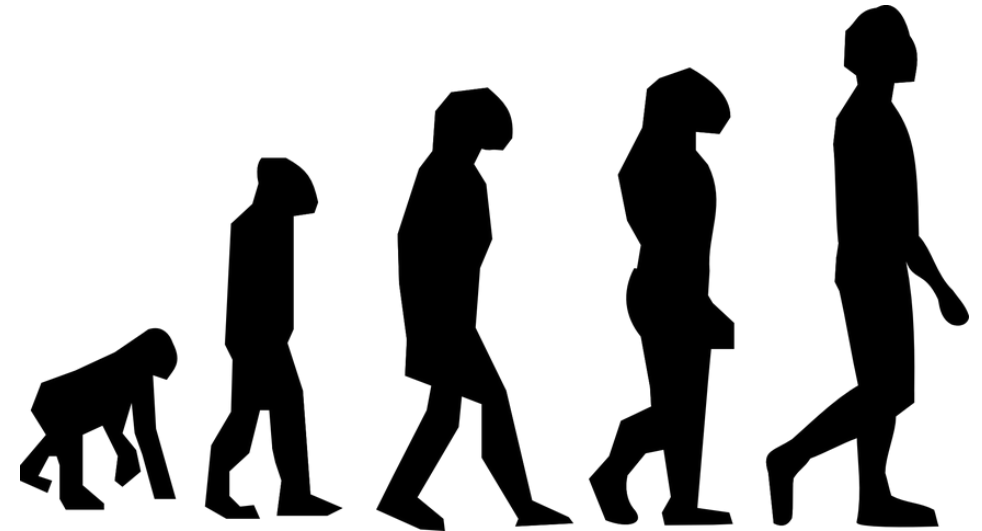
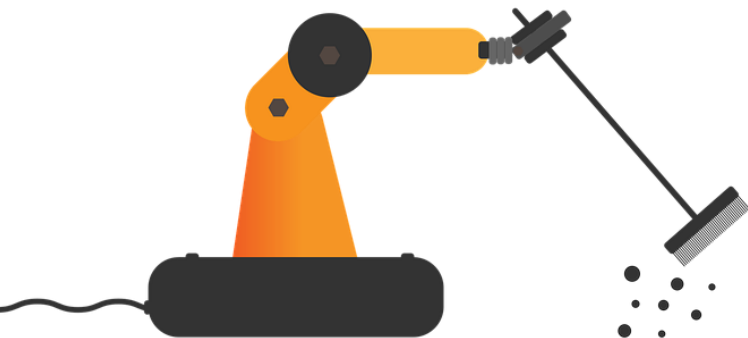
- Some examples of benefits we found while teaching:
  - Easily extending objective functions enables more wide variety of types of problems (e.g. inverting rules of cellular automata for a bachelor course on natural computing)
  - The common data format and many-algorithm analysis options enables for some degree of competitive algorithm design (e.g. a bonus point for the top 10% algorithms based on area under aggregated ECDF)
  - Common structure allows easy reproducibility of submissions + enables the creation of simple test-functions so students can check their algorithm for minimal acceptable performance





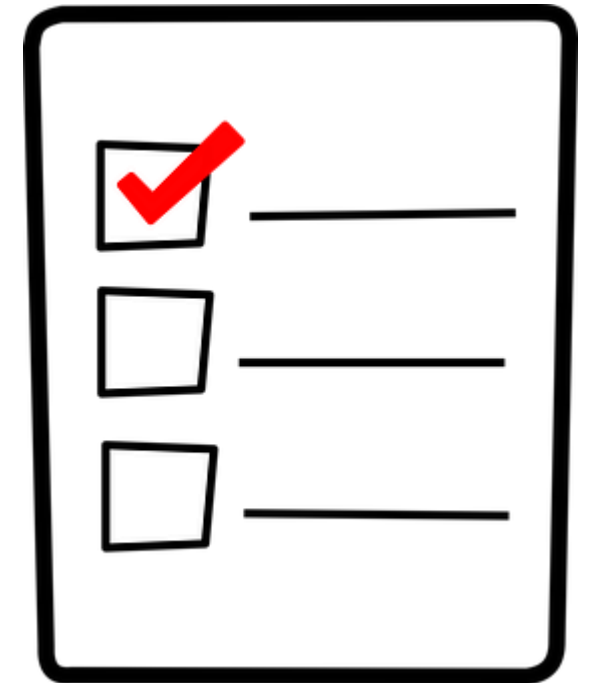
# Student Projects

- Within our group, IOHprofiler is commonly used in both Bsc. and Msc. thesis projects
- Significantly reduces the overhead of setting up correct benchmarking practices and common visualizations
- Makes transitioning to scientific papers easier, since the benchmarking pipeline is correct



# Research Benefits

- Straightforward data collection used in variety of contexts
- Algorithm selection: combine ioh and nevergrad to get data for 13 algorithms on 24 functions
- Dynamic Algorithm Selection: tracking of position information and dynamic attributes to determine switching points [9]
- Supporting algorithm development [10]:
  - Tables downloadable in LaTeX
  - Plots as pdf
  - Statistical testing

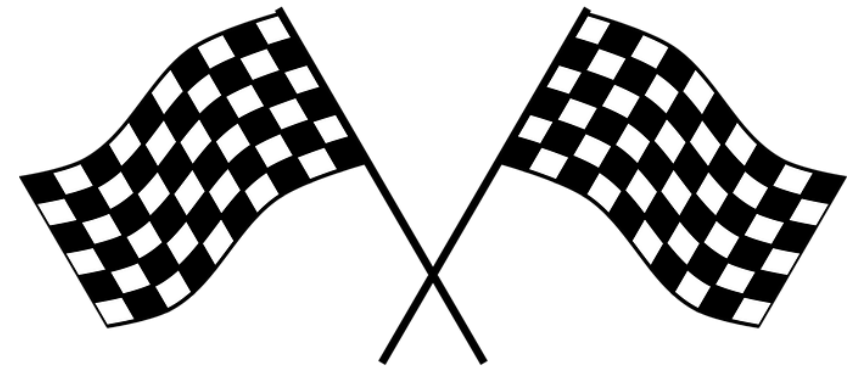


[9] Kostovska, A., Jankovic, A., Vermetten, D., de Nobel, J., Wang, H., Eftimov, T., & Doerr, C. (2022, August). Per-run algorithm selection with warm-starting using trajectory-based features. In Parallel Problem Solving from Nature–PPSN XVII: 17th International Conference, PPSN 2022

[10] Vermetten, D., Caraffini, F., Kononova, A. V., & Bäck, T. (2023). Modular Differential Evolution. arXiv preprint arXiv:2304.09524.

# Competitions / Workshops

- Integrate problem in IOHexperiments:
  - Directly in the C++ backend
  - Using the wrapping functionality
- Make baselines directly available in IOHanalyzer
- Set up template for drop-in tables / figures
- Easy comparison of all combined results
  
- Currently active:
  - Submodular Optimization Competition [11]
  - Star-discrepancy Optimization Competition [12]
  - Strict box-constrained optimization workshop [13]



[11] Neumann, F., Neumann, A., Qian, C., Do, V. A., de Nobel, J., Vermetten, D., ... & Bäck, T. (2023). Benchmarking Algorithms for Submodular Optimization Problems Using IOHProfiler. arXiv preprint arXiv:2302.01464.

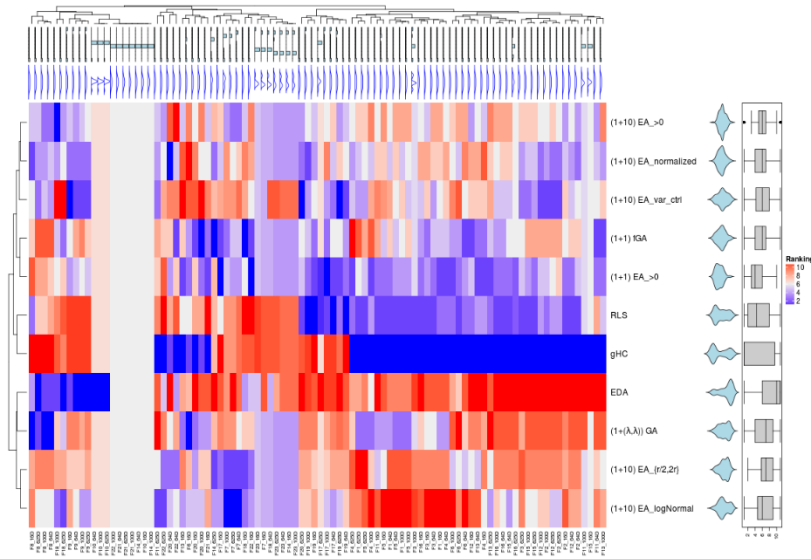
[12] Clement, F., Vermetten, D., de Nobel, J., Jesus, A., Paquete, L., & Doerr, C. Computing Star Discrepancies with Numerical Black-Box Optimization Algorithms.

[13] Kononova, A. et al. SBOX-COST — Strict box-constraint optimization studies Workshop, GECCO 2023

# Collaborations



- Integration IOHexperimenter with ParadisEO (Johann Dréo)
- GUI for DSCtool in IOHanalyzer (Tome Eftimov)
- Implementation of W-model in IOHexperimenter (Thomas Weise)
- Integration of IOHexperimenter problems into Nevergrad
- We are always open to new collaborations!



- 1 original bit string (here: variable-length)  
 $x$  0101 0110 0000 1110 1000 0
- 2 Introduction of Neutrality  
 $\mu=2$  01 01 01 10 00 00 11 10 10 00 0  
 $u_r(g)$  1 1 1 1 0 0 1 1 1 0  $\times$
- 3 Introduction of Epistasis  
 $v=4$  1111 0011 10 insufficient bits, at the end, use  $\eta=2$  instead of  $\eta=4$   
 $e_d$  1110 0110 11
- 4 Multi-Objectivity  
 $m=2, n=6$  1 1 1 0 0 1 1 0 1 1 padding:  $x^*[5]=0$   
 $(x_1, x_2)$  110110 101010  
 $x_1$   $x_2$
- 5 Objective Values  
 $n=6$  110110 101010  
 $f(x_1)=3$   $f(x_2)=6$
- 6 Introduction of Ruggedness  
 $\gamma=12, n=6$   $\gamma'=9$   $f(x_1)=3$   $f(x_2)=6$   
 $r_{12}[f(x_1)]=3$   $r_{12}[f(x_2)]=5$

# OPTION Ontology

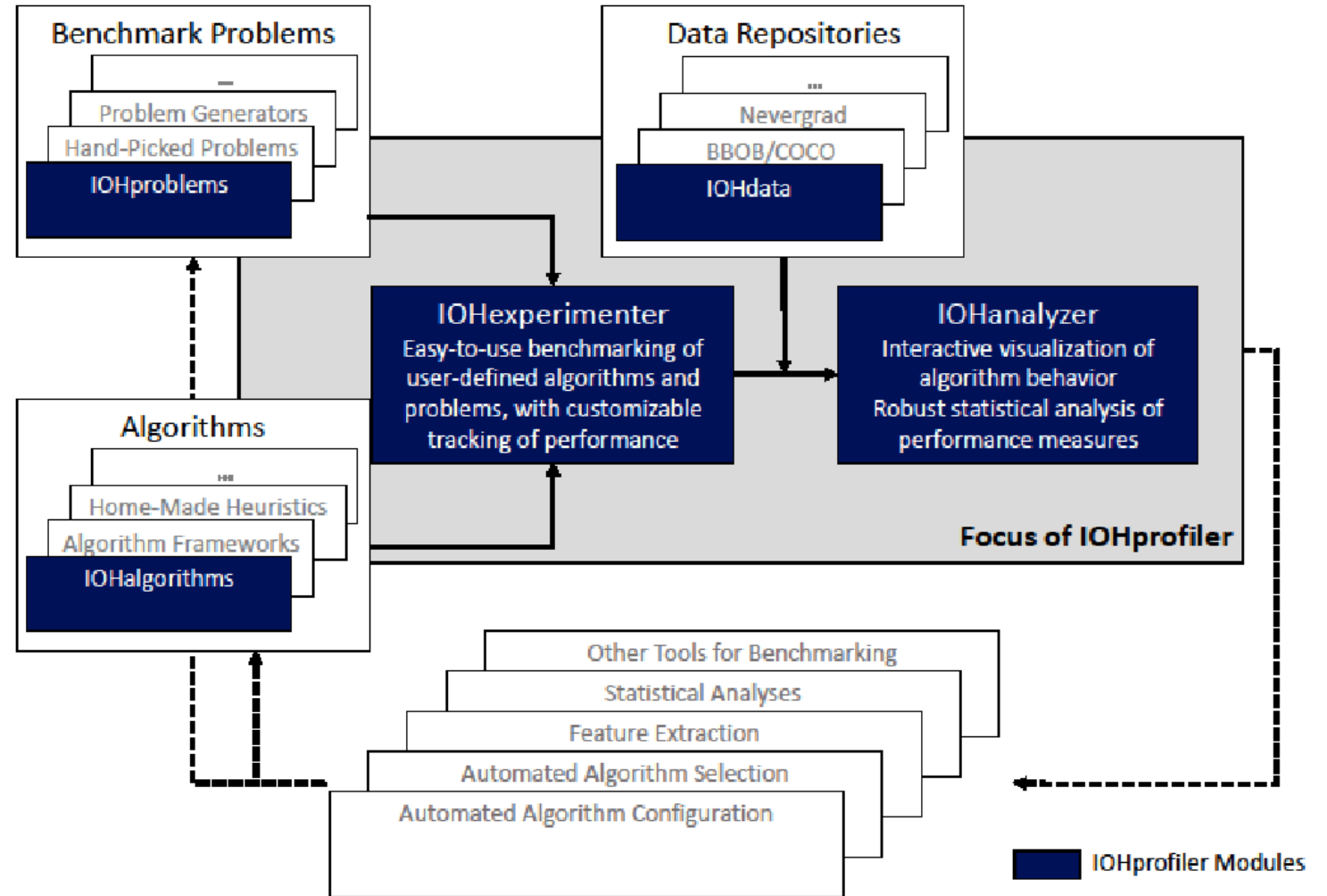
- Benchmark data contains a lot of information
- But not always easy to extract
- Collaboration with OPTION: data ontology for iterative optimization [14]
- Annotated data supports wide variety of queries
- Current prototype interface enables loading data by study name
- Significant potential to implement many more easy to use query types



[14] A. Kostovska, D. Vermetten, C. Doerr, S. Džeroski, Panče Panov and T. Eftimov, "OPTION: OPTImization Algorithm Benchmarking ONtology," in IEEE Transactions on Evolutionary Computation, doi: 10.1109/TEVC.2022.3232844.

# What is Next?

- Extend repository of algorithms + data
- Add interfaces to more tools
- Extensions to mixed-integer and multi-objective problems
- Add more benchmarks for constrained optimization
- More ways of analyzing the data



# Discussion

- How can benchmarking tools like IOHprofiler aid in reproducibility?
- How can we make robust benchmarking easier to do?
- What features would make IOHprofiler fit your use-case better?
- How can we collaborate to make benchmarking as easy and useful as possible?

