Pratical Assignment

Evolutionary Algorithms Course, LIACS, 2023-2024

General Info

- Work in a team of up to 2.
- Enrol in the same group with your teammate in Brightspace

- You need to submit
  - Source code (Python) with requried format
    - We will run your codes with a script, so please make sure you program is compatible with the requirement.
    - Please submit the version which is consistent with the result in your report
  - A Scientific report
    - We provide the template
    - Exercise for writing scientific articles

- Pactical Assignment:
  - PA ddl: Dec. 7, 23:59
  - Every week late: 0.5 pts grade degradation

Solving the F18 and F19 problems from *Pseudo-Boolean Optimization* (PBO) problem set using GA and ES

Low Autocorrelation Binary Sequences (LABS)

F18: Low Autocorrelation Binary Sequences (LABS)

The Low Autocorrelation Binary Sequences (LABS) problem poses a non-linear objective function over a binary sequence space, with the goal to maximize the reciprocal of the sequence's autocorrelation: $x \in \{0,1\}^n$

$$\mathrm{LABS} : x \mapsto \frac{n^2}{2\sum_{k=1}^{n-1}\left(\sum_{i=1}^{n-k} s_i s_{i+k}\right)^2}, \text{ where } s_i = 2x_i - 1.$$

1. Doerr, C., Ye, F., Horesh, N., Wang, H., Shir, O. M., & Bäck, T. (2019, July). Benchmarking discrete optimization heuristics with IOHprofiler. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (pp. 1798-1806).
2. https://iohprofiler.github.io/IOHproblem/PBO

Example Problem: LABS

- Low Autocorrelation of Binary Sequences
- Autocorrelation function on $\{-1,1\}^n$
- Important applications
  - Telecommunications
  - Radar
  - Sonar
- Transformation of variables:
  - $\{0,1\} \rightarrow \{-1,1\}$

The Objective Function

- Search space: $\{0,1\}^n$
- Goal: Find $\mathbf{x} \in \{0,1\}^n$ such that

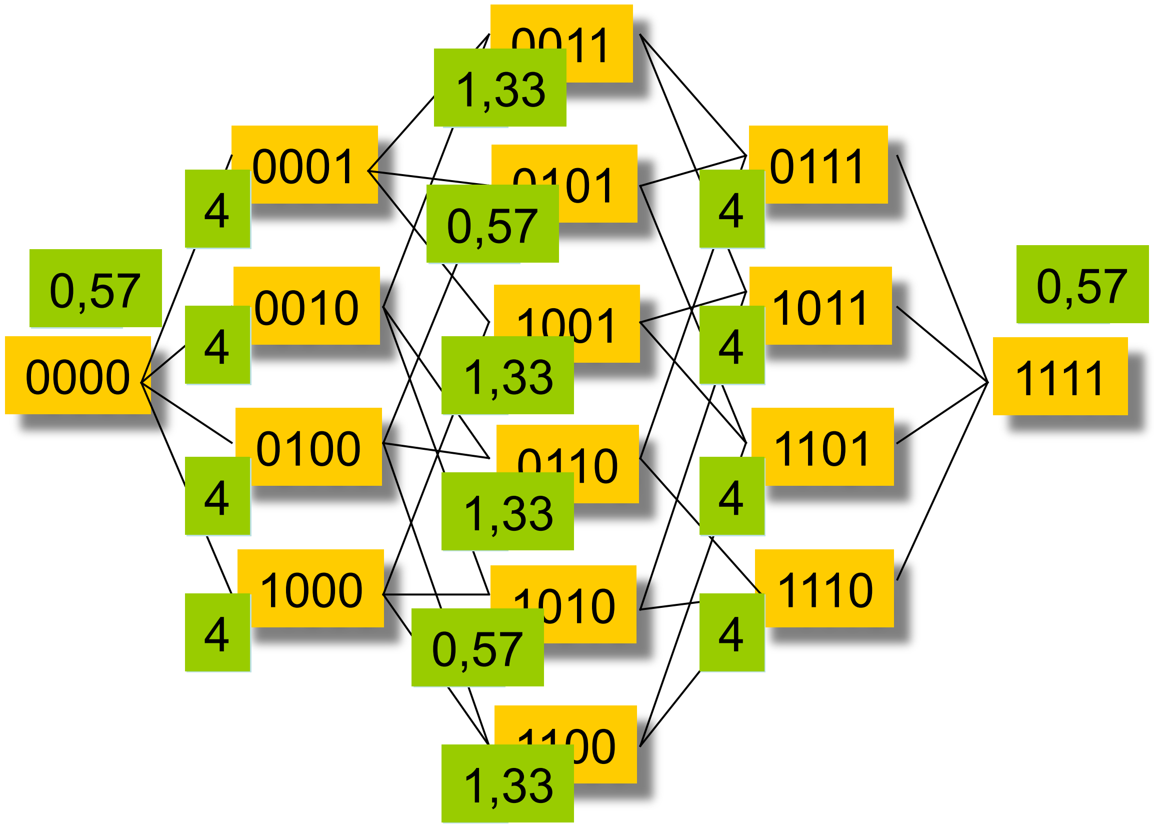
$$E(x) = \sum_{k=1}^{n-1}(\sum_{i=1}^{n-k} y_i \cdot y_{i+k})^2 \rightarrow \min$$

$$y_i = 2x_i - 1$$

- Alternative formulation (merit factor):

$$F(\mathbf{x}) = \frac{n^2}{2E(\mathbf{x})} \rightarrow \max$$

Example: n=4



$$E(\mathbf{x}) = (y_1y_2 + y_2y_3 + y_3y_4)^2 + (y_1y_3 + y_2y_4)^2 + (y_1y_4)^2$$

Some Values

- Theory indicates that

$$\lim_{n\to\infty} \arg\max F(\mathbf{x}) \approx 12.32$$

- See table for some known records
  - Values in bold are not confirmed to be best possible
  - Most optimizers get stuck around 7

n	Best value of f
20	7.6923
50	8.1699
100	**8.6505**
199	**7.5835**
200	**7.4738**
201	**7.5263**
202	**7.3787**
203	**7.5613**
219	**7.2122**
220	**7.0145**
221	**7.2207**
222	**7.0426**

Some Values

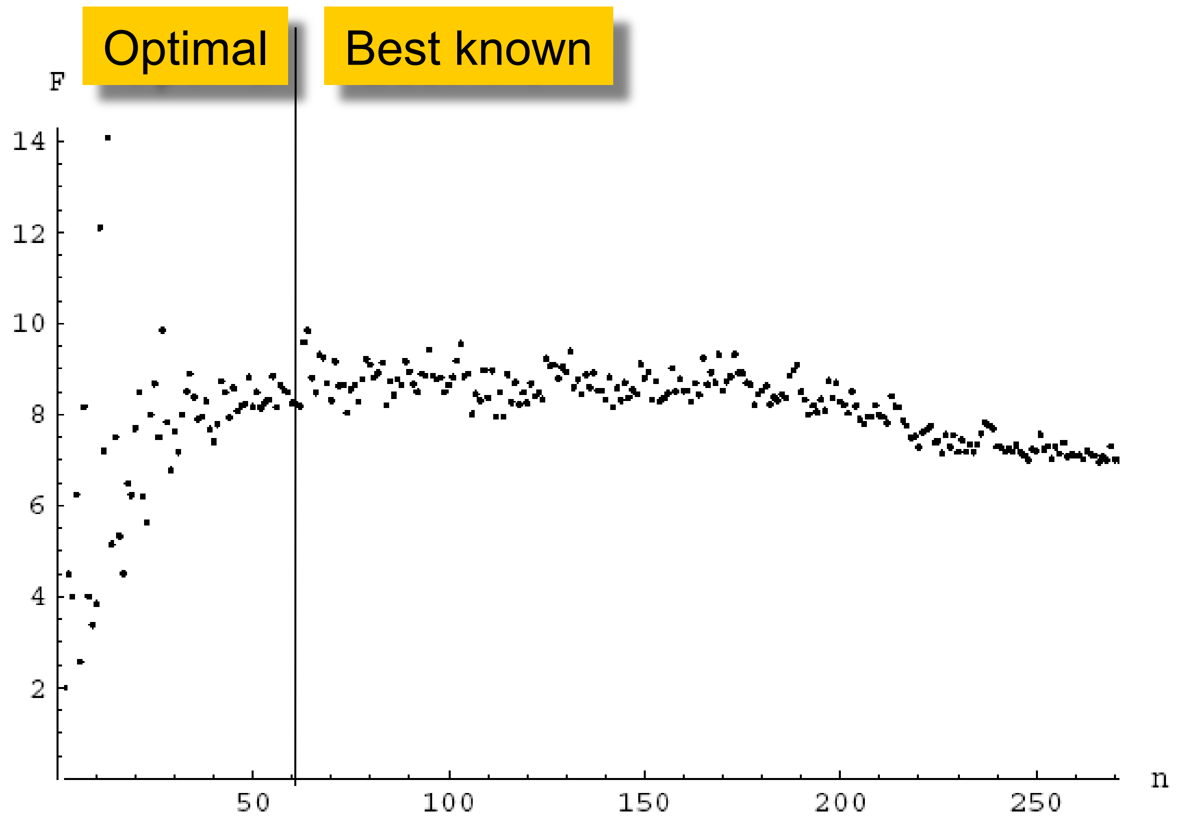


Figure 1: The optimal merit factor (for $2 \le n \le 60$) and the best known merit factor (for $61 \le n \le 271$) for binary sequences of length n.

The ISING Model

F19-F21: The Ising Model

The classical Ising model considers a set of spins placed on a regular lattice $G = ([n], E)$, where each edge $(i, j) \in E$ is associated with an interaction strength J_{ij}. Given a configuration of n spins, $S := (s_1, \ldots, s_n) \in \{-1,1\}^n$, this problem poses a quadratic function, representing the system's energy and depending on its structure J_{ij}. Assuming zero external magnetic fields and using $x_i = (s_i + 1)/2$ and a unit interaction strength, we obtain the following pseudo-Boolean maximization problem:

$$\text{ISING} : x \mapsto \sum_{i,j \in E} \left[x_i x_j + (1 - x_i)\left(1 - x_j\right) \right].$$

In PBO, we consider three instances of lattices: the one-dimensional *ring* (F19), the two-dimensional *torus* (F20), and the two-dimensional *triangular lattice* (F21).

1. Doerr, C., Ye, F., Horesh, N., Wang, H., Shir, O. M., & Bäck, T. (2019, July). Benchmarking discrete optimization heuristics with IOHprofiler. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (pp. 1798-1806).
2. https://iohprofiler.github.io/IOHproblem/PBO

Implementation Details

Requirements

- Implement a genetic algorithm solving the F18 and F19 problems
- Implement an evolution strategy solving the F18 and F19 problems
- Submit at least two files '*studentnumber1_studentnumber2_GA.py*' and '*studentnumber1_studentnumber2_ES.py*'.
- Additional files of other functions are allowed. Please make sure we can get results by running '*python *_GA.py*' and '*python *_ES.py*' without additional arguments.
- Submit a report introducing your algorithms and presenting the experimental results.

Task 1: Genetic Algorithm

Some hints:

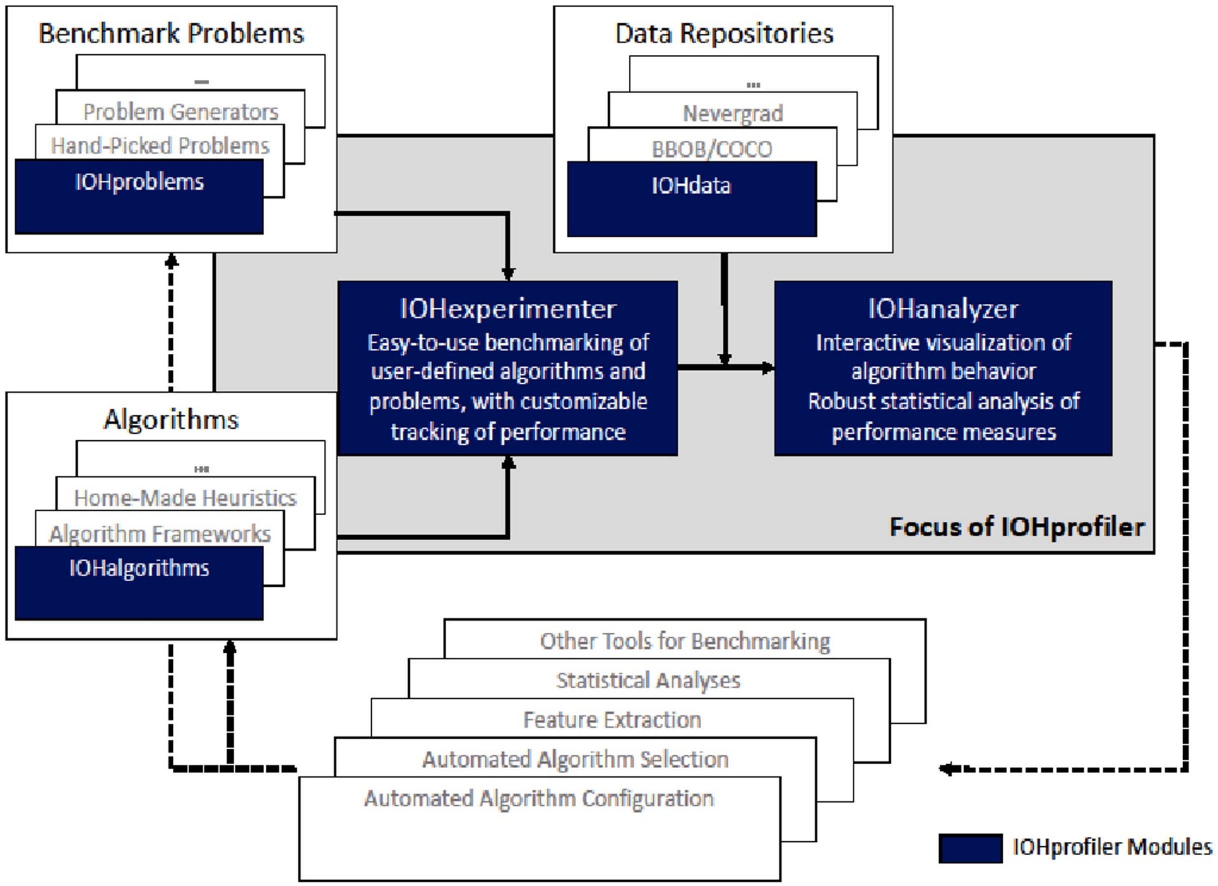
- Which variators (i.e., mutation and crossover) and selection operators will you use?
- What's your suggestion for the parameter settings (e.g., population size, mutation rate, etc.)?

Task 2: Evolution Strategy

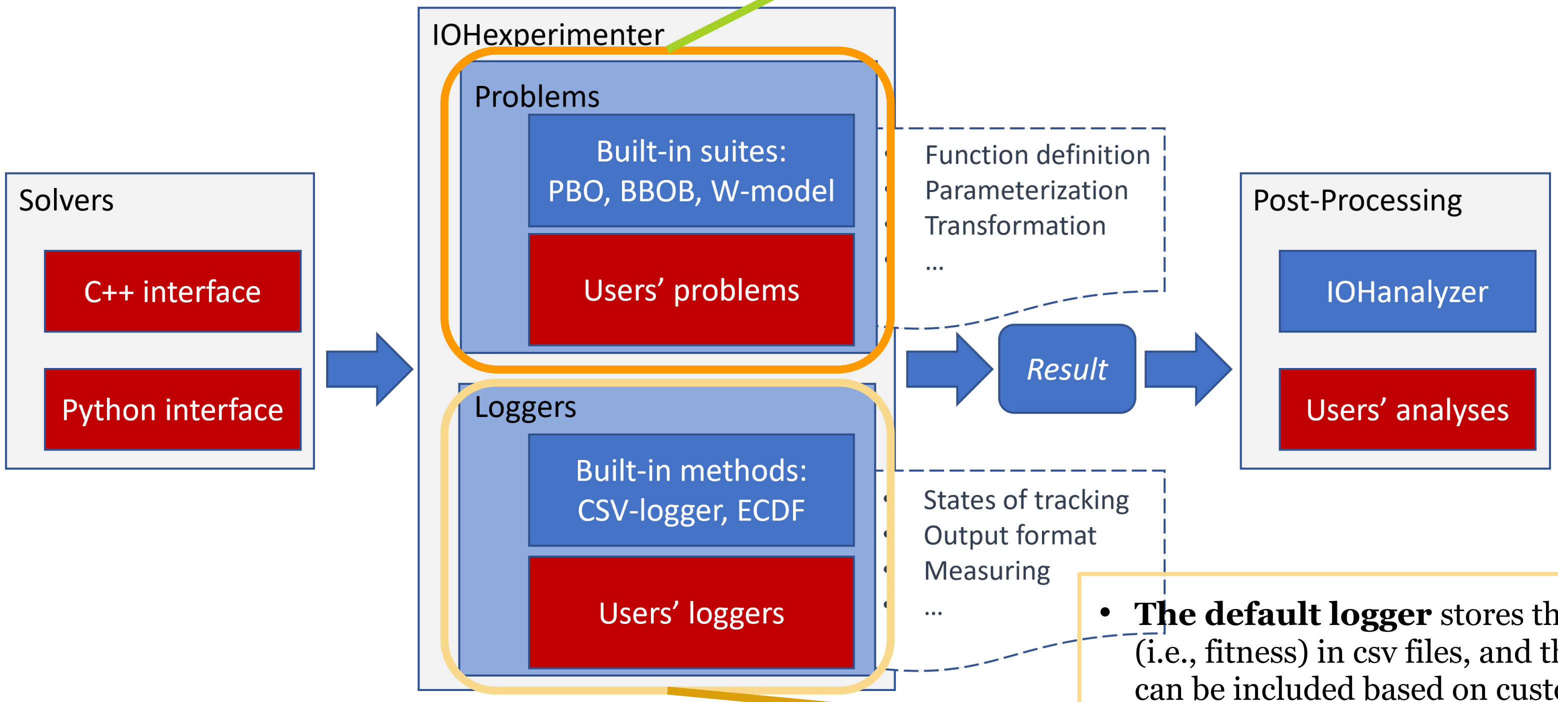
- How to handle the representation of binary/discrete variables while applying evolution strategies?
- What encoding and decoding methods will you use?
- Which variators (e.g., one-σ mutation, correlated mutation, discrete recombination, etc) and selection operators will you use?
- What's your suggestion for the parameter settings (e.g., population size, step size, etc.)?

IOHprofiler
A modular benchmarking platform

IOHprofiler Architecture Overview



Workflow of IOHexperimenter



- For example, the problem to be solved for the assignment.

- **The default logger** stores the algorithm's performance (i.e., fitness) in csv files, and the algorithm's parameters can be included based on customized settings. The csv files can be uploaded to IOHanalyzer for analyzing and visualizing algorithms' performance.
- **Customized** loggers can be created for tasks such as algorithm configuration.

Usage of IOHexperimenter

- Installation: pip package is available at https://pypi.org/project/ioh
- A tutorial is available at https://github.com/IOHprofiler/IOHexperimenter/blob/master/example/tutorial.ipynb
- A full documentation is available at https://iohprofiler.github.io/IOHexperimenter/python

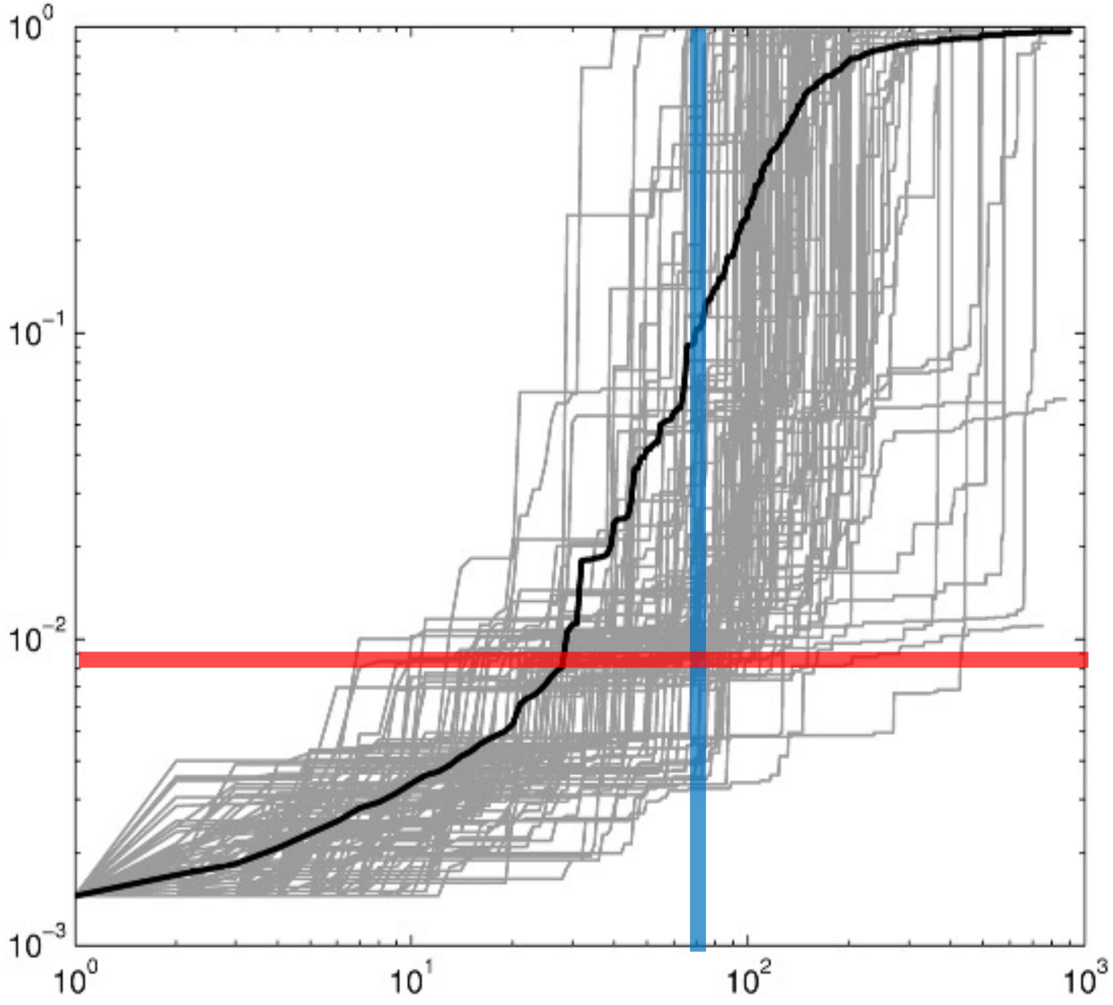
Performance Assessment with IOHanalyzer

- What perspective to consider?

Fixed-budget -
objective samples given
runtime budget

Fixed-target - runtime samples
given target value

success rate: some cruns might
not hit this line

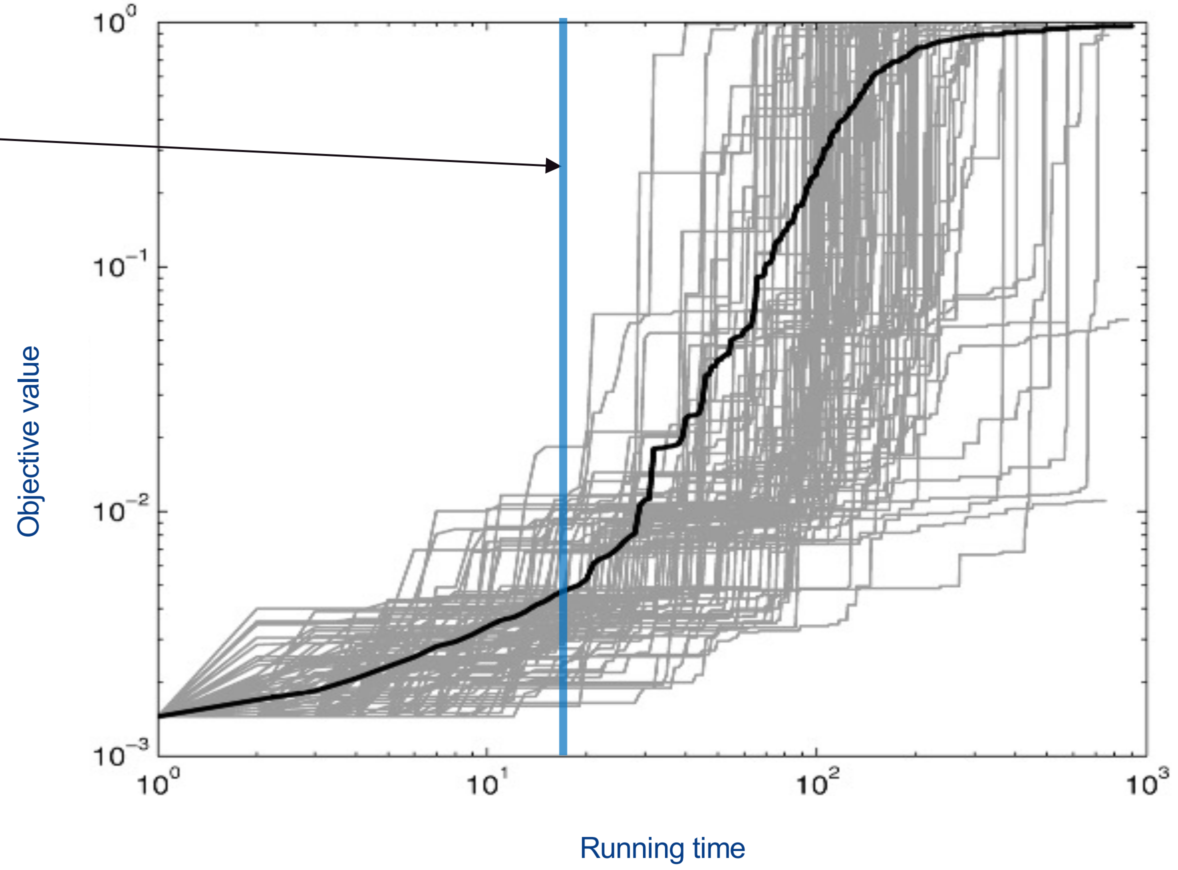


Fixed-budget analysis

Universiteit Leiden
The Netherlands

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

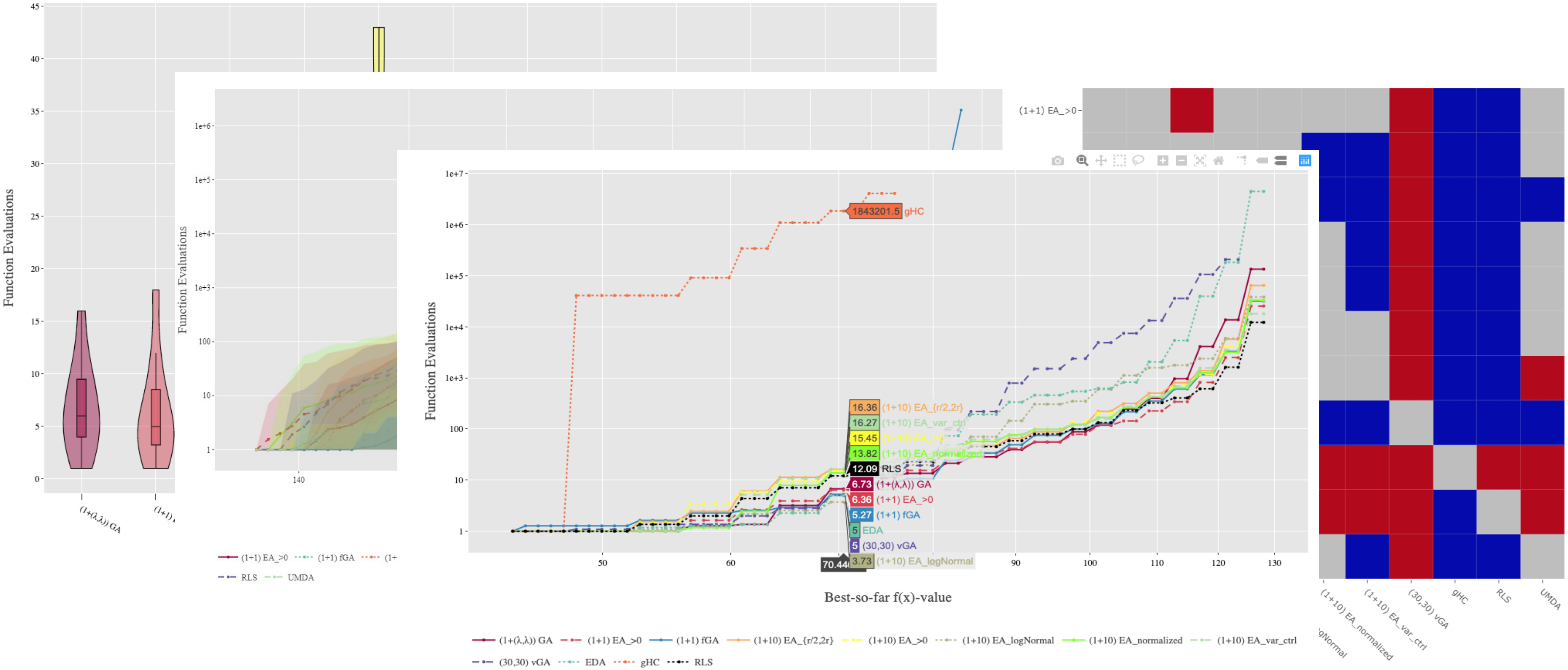
$$V(b) \in \mathbb{R}, \boxed{b} \leq B$$



- The number of runs – *r*

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

Fixed-target Analysis



Algorithm Performance Measures

[**ERT**: Expected Running Time] Given a target value ϕ for a problem P, the ERT of algorithm A for hitting ϕ is

$$\mathrm{ERT}(A, P, \phi) = \frac{\sum_{i=1}^{r} \min\{t_i(A, P, \phi), B\}}{\sum_{i=1}^{r} \mathbb{1}\{t_i(A, P, \phi) < \infty\}},$$

where r is the number of independent runs of A, B is the cutoff time (i.e., the maximal number of solution candidates that algorithm A may evaluate in one run), $t_i(A, P, \phi) \in \mathcal{N} \cup \{\infty\}$ is the running time (for finite values, the running time is the number of function evaluations that the i-th run of A on problem P uses to hit the target ϕ and $t_i(A, P, \phi) = \infty$ is used if none of the solutions is better than ϕ), and $\mathbb{1}(\mathcal{E})$ is the indicator function returning 1 if event $\mathcal{E}$ is true and 0, otherwise. If the algorithm hits the target ϕ in all r runs, the ERT is equal to the average hitting time (AHT).

Algorithm Performance Measures

[**ECDF**: empirical cumulative distribution function of the running time] Given a set of targets $\Phi = \{\phi_i \in \mathbb{R} \mid i \in \{1, 2, ..m\}\}$ for a real-valued problem P and a set of budgets $T = \{t_j \in \mathbb{N} \mid j \in \{1, 2, ..B\}\}$ for an algorithm A, the ECDF value of A at budget t_j is the fraction of (run, target)-pairs (r, ϕ_i) that satisfy that the run r of the algorithm A finds a solution has fitness at least as good as ϕ_i within the budget t_j.

Algorithm Performance Measures

[**AUC**: area under the ECDF curve] Given a set of targets $\Phi = \{\phi_i \in \mathbb{R} \mid i \in \{1, 2, ..m\}\}$ and a set of budgets $T = \{t_j \in \{1, 2, ..B\} \mid j \in \{1, 2, ..z\}\}$, the AUC $\in [0, 1]$ (normalized over B) of algorithm A on problem P is the area under the ECDF curve of the running time over multiple targets. For minimization, it reads

$$\mathrm{AUC}(A, P, \Phi, T) = \frac{\sum_{h=1}^{r} \sum_{i=1}^{m} \sum_{j=1}^{z} \mathbb{1}\{\phi_h(A, P, t_j) \leq \phi_i\}}{rmz},$$

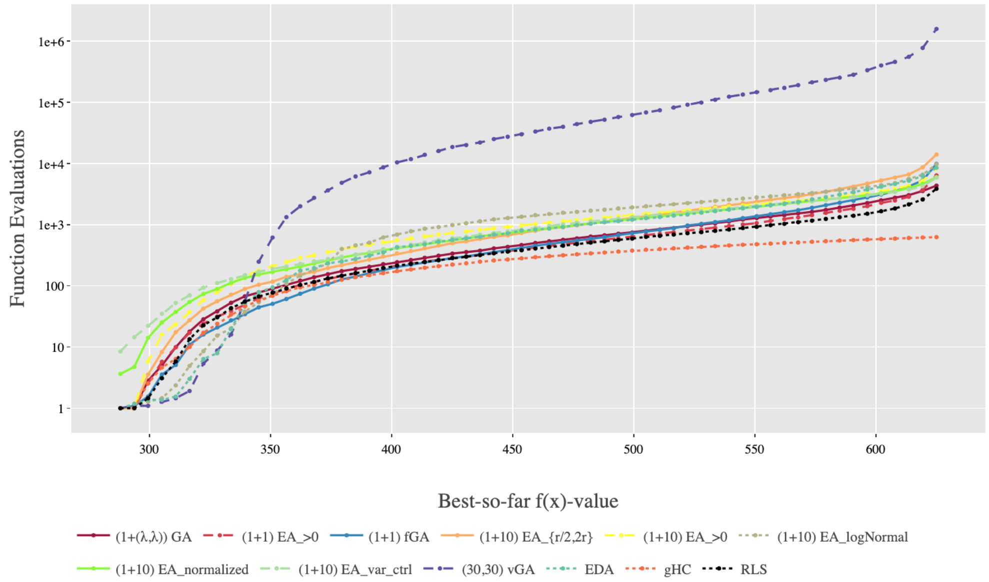
where r is the number of independent runs of A and $\phi_h(A, P, t)$ denotes the value of the best solution that A evaluated within its first t evaluations of the run h. Note that, for this assignment, we consider an approximation of AUC for a set of budgets $T \subset \{1, 2, ..B\}$.

Usage of IOHanalyzer

- GUI: https://iohanalyzer.liacs.nl
- GitHub project: https://github.com/IOHprofiler/IOHanalyzer . You can also install a local version following the tutorial on the git page.
- Paper: https://dl.acm.org/doi/full/10.1145/3510426

What to report

- Description of the genetic algorithm and the evolution strategy
  - Introduce the operators and the parameter setting that you suggest. Though you might have tested different configurations, please make it clear which setting you choose for the submission
- Report the average best-found fitness f(x) of **20 independent runs**
- Report the AUC values
- The empirical analysis of your algorithms' performance using ERT and ECDF curves
- Any other details/results you wish to include



General Info

- How to evaluate your PA?
  - Following the guidelines (10%)
    - You will get full score if you follow all guidelines
  - Experimental Results (45%)
    - Based on the rank of your algorithm result among algorithms of all groups.
  - Presentation (45%)
    - Based on the presentation of the design of algorithms, experimental settings, and discussion about the results.
- Other:
  - Plagiarism check: if report copies more than 30%, PA grade is 0.
  - If the results in your report do not match the results we obtain from using your codes, PA grade is 0.
  - If the results of your algorithms rank top 2, you will get a 0.5 bonus for the final grade.

Evaluation

- The submissions will be ranked based on the *averaged best-found fitness* $f(x)$ and AUC *values* obtained using the genetic algorithm and the evolution strategy (there will be four rank lists).
- For each list (i.e., f_ga, f_es, auc_ga, and auc_es), your score will be calculated as

$$10 - \frac{s - s_{last}}{s_{best} - s_{last}} \times 3,$$

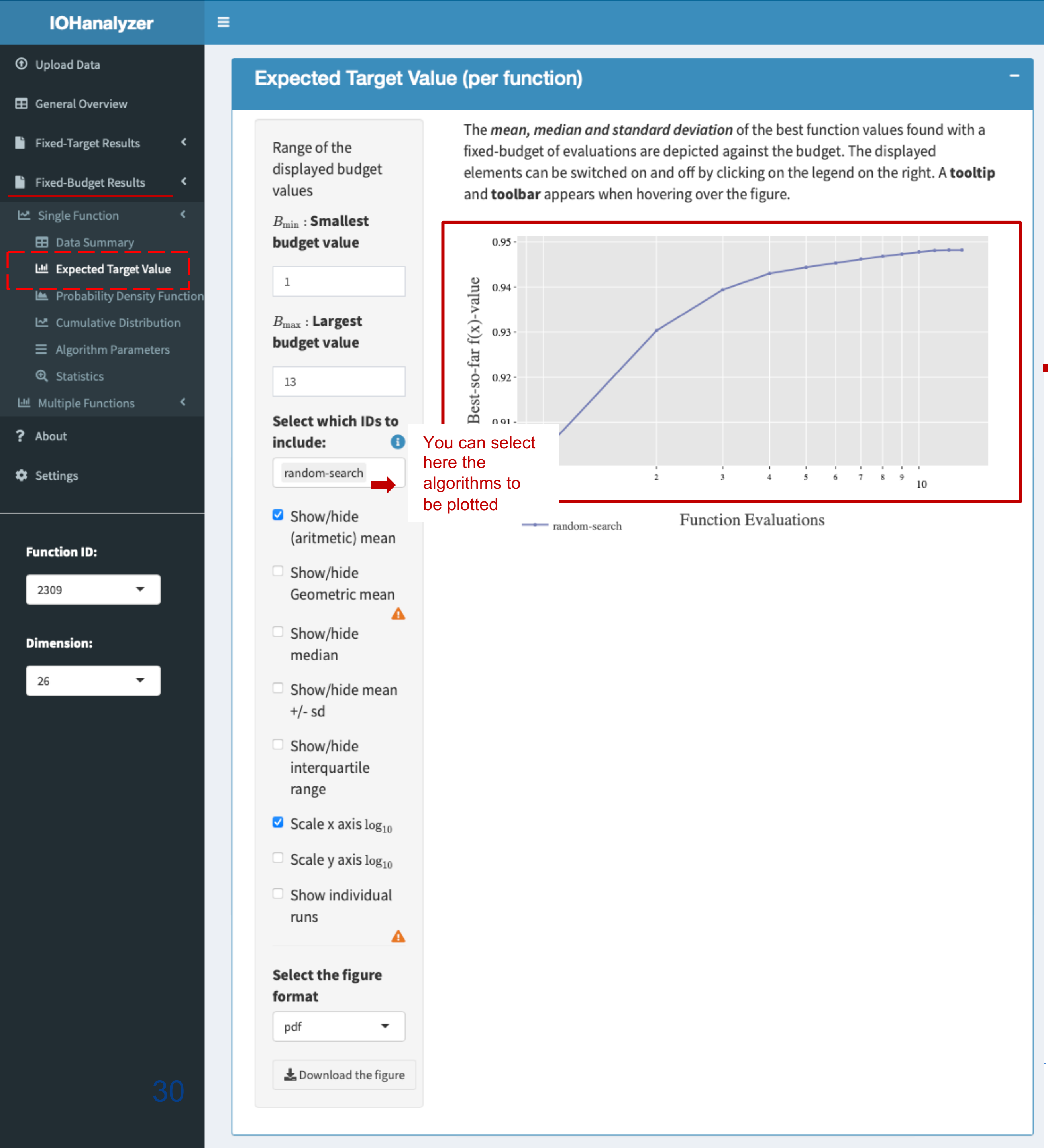
where s is the performance value obtained by your algorithm, s_{best} is the best value obtained by all the submissions, and s_{last} is the lowest ranked value of all the submissions. Your submission will be graded basde on the averaged score of the four criteria for the part of the experimental results (45%).

- The top two teams of the experimental results will obtain a 0.5 bonus adding to the final grade of the course.
- All submissions whose codes produce the identical results in the report can obtain at least 0.7*4.5 for the part of the experimental results. When the results are not identical or the submitted codes are not executable, this part will be scored as 0.6*4.5 after convincing arguments from the team, or the practical assignment will be scored as 0.

Tutorial

- Install ioh package
  - In terminal: “python –m pip install ioh”
  - You can follow the tutorial (https://github.com/IOHprofiler/IOHexperimenter/blob/master/example/tutorial.ipynb) if you are interested with the additional functionalities of the tool.

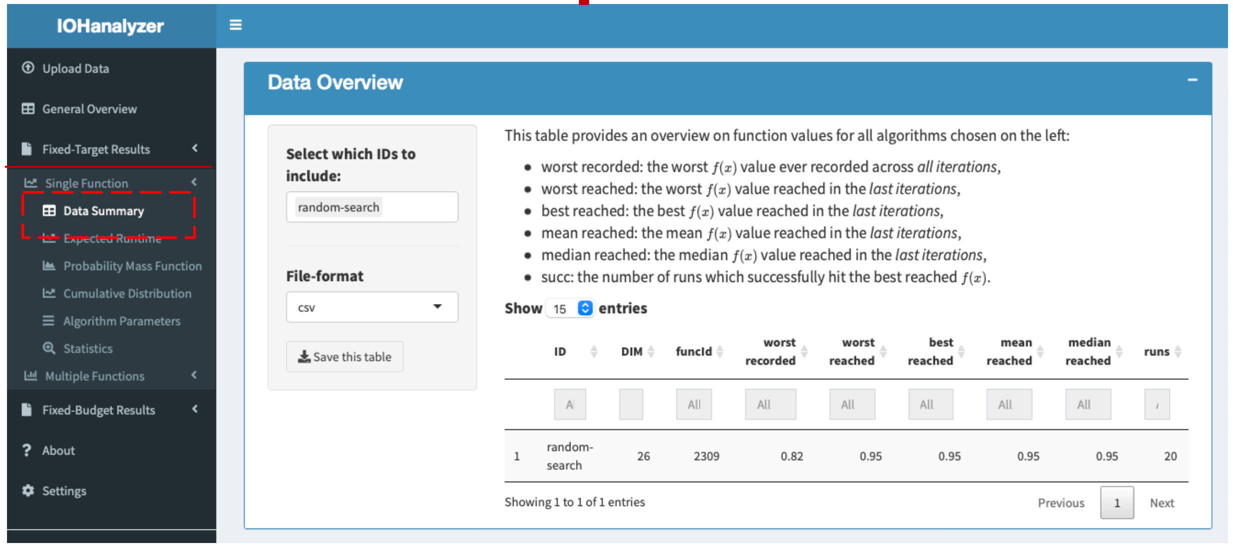
How to access the best-found fitness using IOHanalyzer.



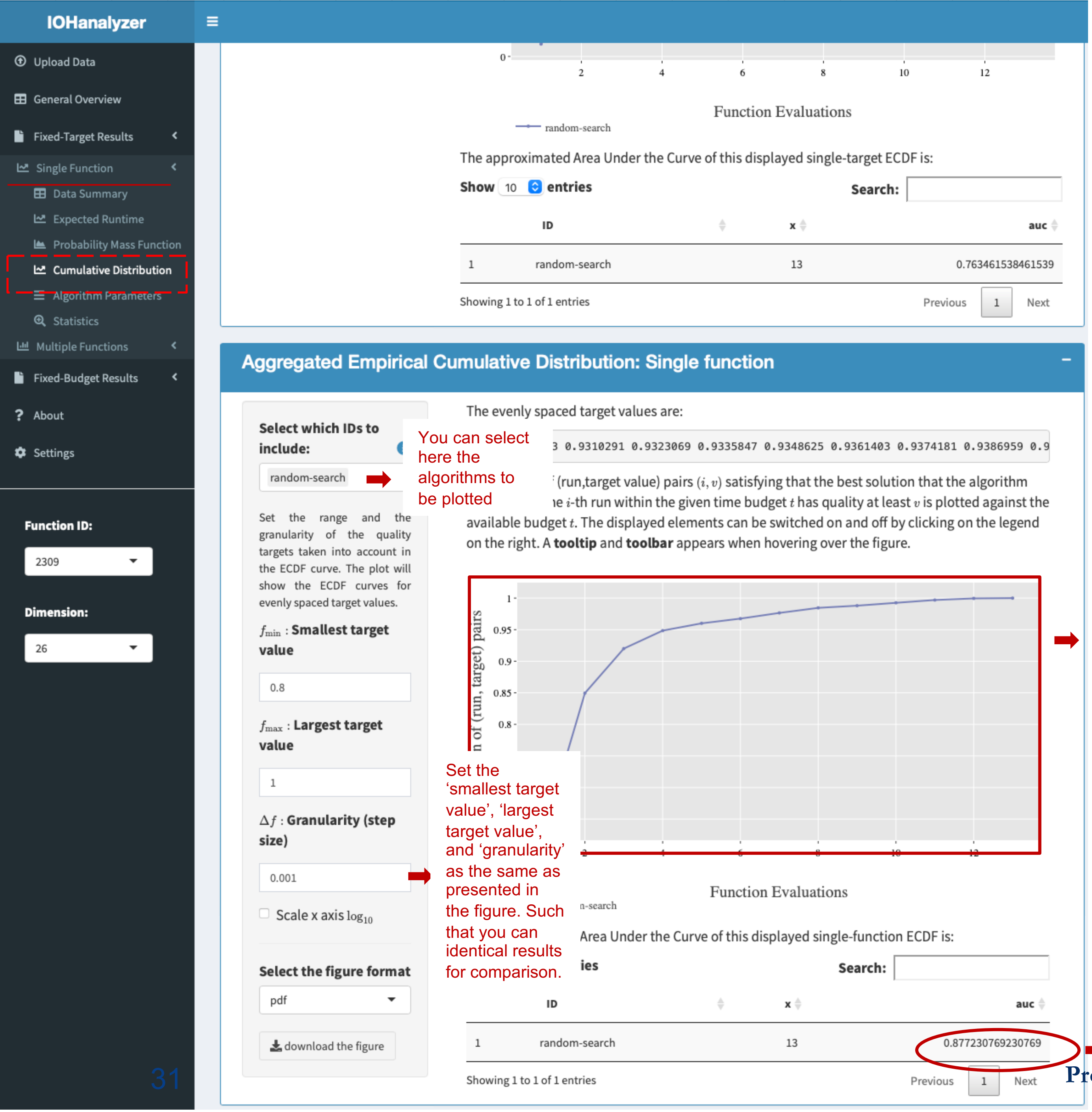
You can select here the algorithms to be plotted

The figure presents the convergence process of your algorithm while the function evaluation increases.

Useful information, e.g., best-found fitness, the average of the best-found fitness, etc., can be obtained in the plotted table.



How to access the ECDF curves and the AUC values using IOHanalyzer.



The figure shall be downloaded and presented in your submission. Please explain your observations on the figure.

This is the approximation of the AUC value you shall report. And for your implementation, the value of 'x' shall be 50=,000 because the assignment requires the given budget of 5,000

Tips

- Test the algorithm in **20** independent runs
- The budget is **5,000** for each run. The ioh package records this automatically, so no way to cheat!
- Remember to execute '.reset()' after each independent run of your algorithm. This will reset the records (i.e., best_found fitness, uesd evaluation times, etc.) of the ioh package.
- Set a fixed random seed in the implementation so that we can obtain the same results as those in the report by running the submitted codes.
- Set the algorithm_name as 'studentnumber1_studentnumber2'