Universiteit Leiden
The Netherlands

Pratical Assignment 2

General Info

- Deadline
  - Submission: 8th Dec, 23:59
  - Submit your report/codes on Brightspace. Please submit codes and the report separately, not compressing them in a zip file.
- Penalty : -0.5/week
- Can work with a group (max. 2 persons)
  - Register your team on Brightspace. You need to enroll into one of the teams in the new category for the second practical assignment!

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 of your report do not match the result we get using your codes, PA grade is 0.

Using Evolution Strategy to solve the BBOB problems

BBOB problems

- The Black-box Optimization Benchmarking (BBOB) suite provides 24 noise-free real-parameter single-objective benchmark problems.
- Each problem consists of a function that is to be minimized: $f: [-5,5]^n \longmapsto \mathbb{R}$
- Definition of the problems can be found in http://coco.lri.fr/downloads/download15.01/bbobdocfunctions.pdf
- Note that IOHanalyzer presents the precision $f(x) - f^*$ comparing the fitness $f(x)$ of the found solutions to the optimum f^*. However, the algorithm gets the raw fitness $f(x)$ using 'func(x)'.

Goals

- Implement an evolution strategy to solve the 24 BBOB problems.
- Provide a suggested setting for all the problems.
- Present the result of your algorithm using an approximation of **the area under the empirical cumulative distribution function curve (AUC)**. (This is one of the requirements for the assignment!)
- You can also analyze the algorithms' performance on specific algorithms using the expected running time (ERT) that was applied for the PA1. (Possible bonus if you find something interesting.)

Empirical Cumulative Distribution Function

[**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.

Area under the ECDF Curve

[**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 $T \subset \{1, 2, ..B\}$.

Experimental settings

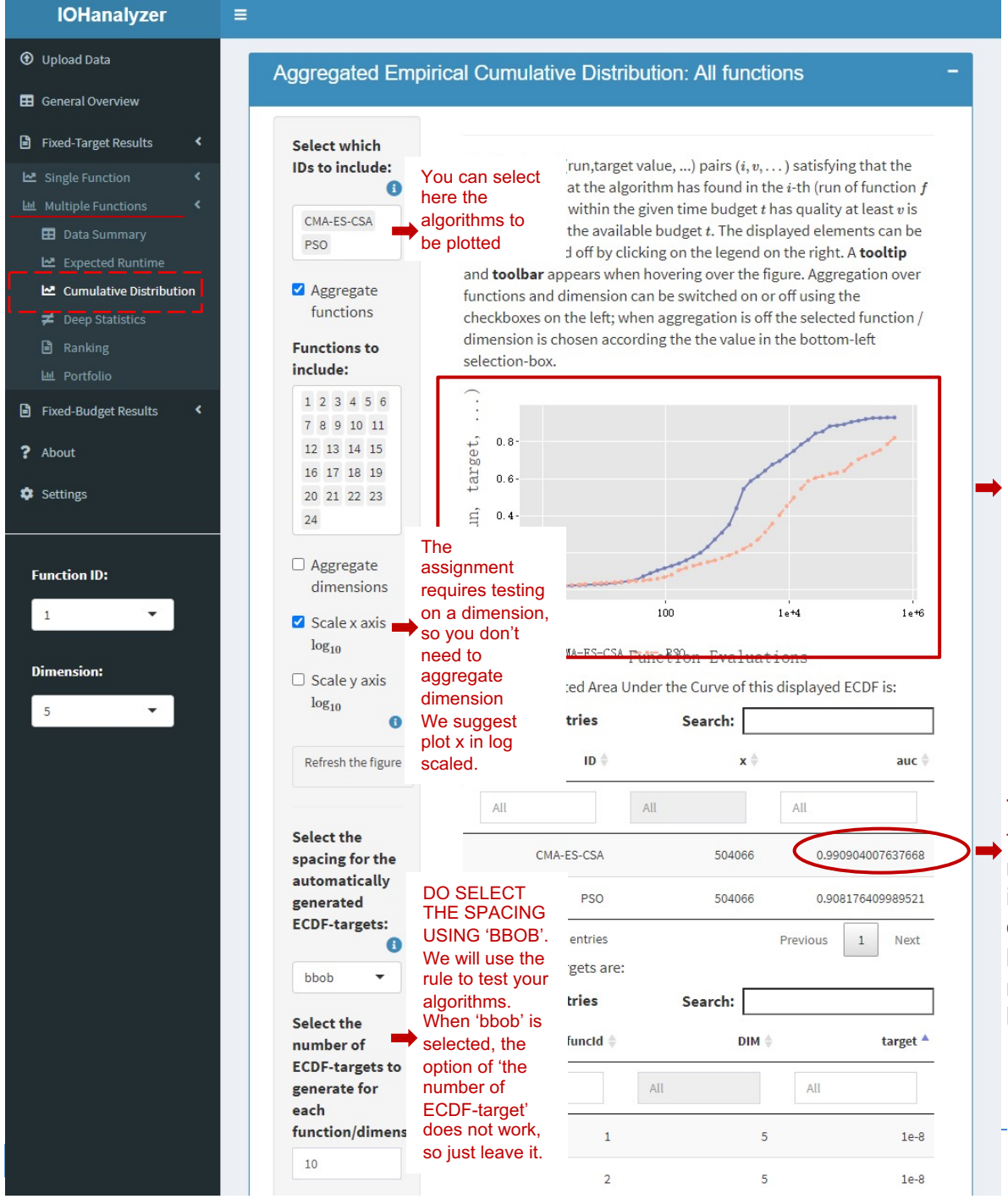
- Dimension: $n = 5$
- Budget (maximal function evaluations, not iterations): 50,000
- The algorithm shall be tested on 10 instances for each problem, and the algorithm performs once for each instance.
- Please name the 'algorithm_name' in the logger as 'studentID1_studentID2'

- Implementation: Please follow the structure of the given examples (available on Brightspace).

What to submit

- A **runnable** c++/python code, containing your implementation of the algorithm.
- A **readable** report that summaries
  - Describe the algorithm (evolution strategy) that you implement. You can present a pseudo-code and describe the procedures. DO NOT copy-paste your programming codes.
  - Parameter values that you suggest. Note that you present only **one suggestion** for all problems.
  - Results of your algorithm, including an aggregated ECDF figure and the AUC value.
  - (Possible observations/Analyses of the algorithm performance and any other interesting findings).

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



You can select here the algorithms to be plotted

The assignment requires testing on a dimension, so you don't need to aggregate dimension We suggest plot x in log scaled.

DO SELECT THE SPACING USING 'BBOB'. We will use the rule to test your algorithms. When 'bbob' is selected, the option of 'the number of ECDF-target' does not work, so just leave it.

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 being 50,000

Note that on IOHanalyzer, the set of targets and the set of budgets are predefined for calculating AUC when selecting the "bbob"spacing such that the comparison among your submissions is fair.

Tips

- This assignment requires doing minimization, not maximization!
- Please suggest only one parameter settings for all the problems.
- Please name the ‘algorithm_name’ in the logger as ‘studentID1_studentID2’
- The aggregated ECDF curve and the AUC value that are required for the assignment evaluate an algorithm’s performance across multiple functions.
- The results for a single function, for example, the expected runtime, are aggregated across the tested instances for each problem.
- Note that IOHanalyzer presents the precision $f(x) - f^*$ comparing the fitness $f(x)$ of the found solutions to the optimum f^*. However, the algorithm gets the raw fitness $f(x)$ using ‘func(x)’.
- The budget refers to the maximal function evaluations (revoking the ‘func(x)’ (or problem(x) for c++), not the iterations.