Assignment 2: Pseudo-Boolean Optimization

Core Body of Knowledge classification: Abstraction, Design, Teamwork concepts & issues, Data, Programming, Systems development, Analysis

Due date: 11:59pm, 6th October 2025, **Weight:** 10% of course

1 Assignment structure and practical requirements

The assignment is done in **groups of 5-6 students**. Each student has to take major responsibility for parts of the assignment and collaborate with the team members. The group has to make sure that the workload is evenly distributed. Report progress and task distribution to your tutor during the workshops and obtain feedback.

Do not use generative AI for coding or text answers. Do not copy code or text from anywhere. We are automatically running plagiarism checks and generative AI detectors. Instead, make your submission original and unique by writing everything yourself, choosing function and variable names well and documenting your code with your own comments.

You have to use the programming language Python and upload all source code, data, and documentation in one zip file by the due date.

2 Team roles and contributions

Include in a file `doc/team_contributions.txt` the name and student id of each group member and write a short paragraph for each group member about the role and contributions to the assignment. **Submission of the file `doc/team_contributions.txt` detailing the contribution of each student is mandatory to receive marks for this assignment.**

3 Assignment

In this assignment, you will work with IOHprofiler https://iohprofiler.github.io/ which is a tool to run and analyse iterative search algorithms. A general tutorial is available at https://github.com/IOHprofiler/IOHexperimenter/blob/master/example/tutorial.ipynb You will implement different algorithms for Pseudo-Boolean Optimization (PBO) and examine their performance with the use of IOHprofiler. IOHprofiler provides different benchmark problems as part of PBO problem set and a detailed description of them can be found at https://iohprofiler.github.io/IOHproblem/PBO

Exercise 1 *IOH Basic functionality (10 marks)*

We will make use of objective functions from the IOHexperimenter package, which we first need to install as follows: *pip install ioh*. The number (1-15) of each function in PBO serves as an identifier to access the corresponding problem. To get familiar, with the environment, run the Random Search algorithm provided in *problem_example.py* on the problems F1 (om), F2 (lo), F18 (labs). Upload the resulting data to IOHanalyzer https://iohanalyzer.liacs.nl/ and plot the fixed budget results for each function separately. Provide the 3 plots you obtained as part of your submission.

Exercise 2 *RLS and (1+1) EA (5+15+5+5 marks)*

- Implement the algorithms RLS and (1+1) EA introduced in the lecture in week 3.
- Run Random Search, RLS, and (1+1) EA ten times on each of the PBO benchmark problems F1, F2, F3, F18, F23, F24, F25 with problem size $n = 100$ for 100000 iterations. Provide for each of the functions a fixed budget plot showing the mean and standard deviation for the three algorithms. Provide for each function a summary of the results and interesting observations.
- Prove that Random Search needs with probability $1 - e^{-\Omega(n)}$ at least a budget of $2^{n/2}$ fitness evaluations to reach an optimal search point for the function F1.
- Prove that the number of fitness evaluations of RLS and (1+1) EA to reach an optimal search point for the function F3 is $O(n \log n)$ with probability $\Omega(1)$.

Exercise 3 *Your Genetic Algorithm (20 marks)*

Design your own GA using uniform crossover, mutation and a parent population of at least 10 individuals. Run your GA with the same function and iteration budget setting as in Exercise 2. Your algorithm should perform as best as possible. Add the results for comparison to the plots from Exercise 2 and provide them and an analysis of your results as part of your submission.

Exercise 4 *MMAS and MMAS* (10+10 marks)*

This exercise is based on the article "Analysis of different MMAS ACO algorithms on unimodal functions and plateaus" available at https://link.springer.com/article/10.1007/s11721-008-0023-3.

- Implement the algorithms MMAS and MMAS* shown in Figure 3 of the article using the construction procedure outlined in Figure 1 and 2.
- Run the algorithms for each value of $\rho \in \{1, 1/\sqrt{n}, 1/n\}$ with the same function and iteration budget setting as in Exercise 2. Add the results for comparison to the plots from Exercise 2 and provide them together with an analysis of your results as part of your submission.

Exercise 5 *Your ACO Algorithm (20 marks)*

Design your own ACO algorithm using at least 10 ants based on the lecture material (you may incorporate local search to improve constructed solutions). Run your ACO algorithms with the same function and iteration budget setting as in Exercise 2. Your algorithm should perform as best as possible. Provide plots for the considered functions that compare your ACO approach to MMAS and MMAS* together with an analysis of your results as part of your submission.

Exercise 6 *Runtime Analysis of ACO (Postgraduate students only) (30 marks)*

This exercise is for COMP SCI 7316 students only. PG Students have to submit for this exercise individually through MyUni (in addition to the group submission for Exercise 1-5.)

You have to provide in your own words an explanation of the results obtained in Section 3.1 and 3.2 of the article "Analysis of different MMAS ACO algorithms on unimodal functions and plateaus" available at https://link.springer.com/article/10.1007/s11721-008-0023-3. As part of your submission you should:

- Discuss how the methods for the analysis relate to the method of fitness-based partitions discussed in the lecture.
- Compare the upper bounds for MMAS and MMAS* to the ones for the (1+1) EA.
- Discuss how the parameter ρ impacts the upper bounds on the runtime of MMAS and MMAS* on the functions considered in Section 3.1 and 3.2.

Word limit for your submission of this exercise is 500 words.

4 Marking

Marking will be done according to the following guidelines:

- correct overall implementation – 20%
- quality of the results – 20%
- analysis of results in form of plots and explanations - 60%

5 Submission

Make sure you have addressed every single instruction in the assignment and that your code runs correctly. Construct a file `doc/readme.txt` which outlines how to run your code. Place all implementations placed into a directory `final/code/` and all figures and analyses into a directory `final/doc/`. Create subfolders for the different exercises.

Submit a zip file including all files on MyUni (one per group).