Universiteit Leiden
The Netherlands

Practical Assignment Part I

Evolutionary Algorithms Course, LIACS, 2025-2026

Solving the F18 and F23 problems from the *Pseudo-Boolean Optimization* (PBO) problem set using Genetic Algorithms

F18: 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

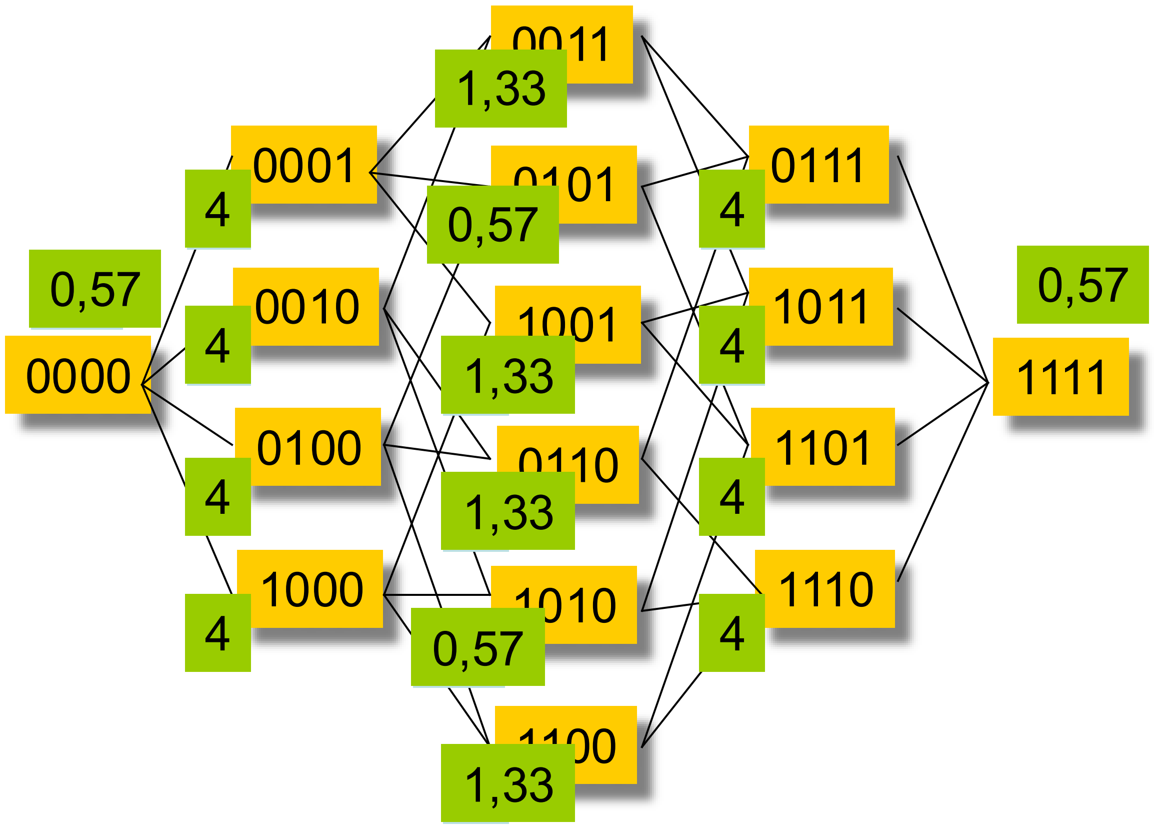
$$E(x) = \sum_{k=1}^{n-1} \Big(\sum_{i=1}^{n-k} y_i \cdot y_{i+k}\Big)^2 \to \min$$

$$y_i = 2x_i - 1$$

- Alternative formulation (merit factor):

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

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

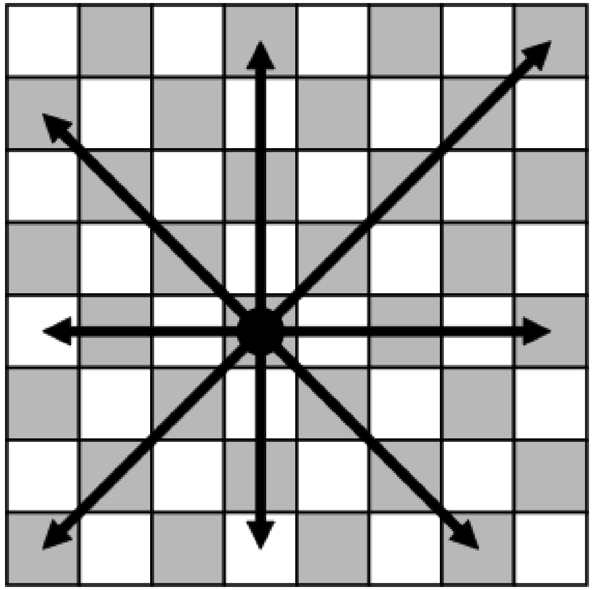
$$\lim_{n \to \infty} \arg\max F(\mathbf{x}) \approx 12.32$$

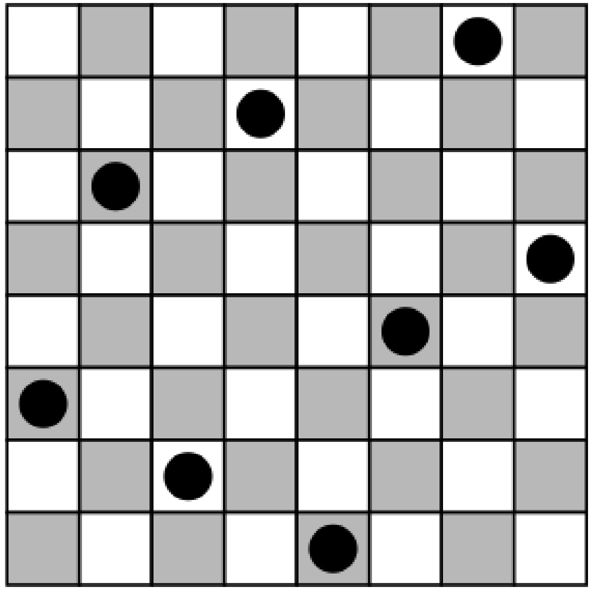
- See table for some known records
  - Values in bold are not confirmed to be best possible
  - Most optimizers get stuck around 7

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**

F23: N-Queens Problem

The N-queens problem (NQP) is defined as the task to place N queens on an $\mathrm{N} \times \mathrm{N}$ chessboard in such a way that they cannot attack each other. The figure below provides an illustration for the 8-queens problem. Notably, the NQP is actually an instance of the MIVS problem– when consideringa graph on which all possible queen-attacks are defined as edge.





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

Assignment Task

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

- You need to submit
  - Source code (Python) with the required format
    - Python template will be provided
    - We will run your code, so please make sure your program meets the requirements.
    - Please submit the version that is consistent with the result in your report.
  - A Scientific report
    - We provide the template.
    - Exercise for writing scientific articles.

- Practical Assignment:
  - PA deadline: Dec. 11, 23:59
  - Every week late: 0.5 pts grade degradation

Requirements and Details

Requirements

- Implement a Genetic Algorithm (GA) to solve the F18 and F23 problems.

- Tune the hyper-parameters of your own implementation with **100,000 function evaluations on both problems**
  - The hyperparameter setting determined by your tuning procedure has to work on both problems
  - Using different hyperparameter setting on each problem is NOT allowed
  - You can use ANY tuning methods for the hyperparameters; think about it

- Submit the code of GA, '*studentnumber1_studentnumber2_GA.py*'. For a group of three, '*studentnumber1_studentnumber2_ studentnumber3_GA.py*'.

- Submit the code for the tuning procedure, '*studentnumber1_studentnumber2_tuning.py*'. For a group of three, '*studentnumber1_studentnumber2_ studentnumber3_ tuning.py*'.
  - The tuning code should make a function call to the GA implementation.

Requirements

- Additional files of other functions are allowed. However, we will only execute the GA and the tuning codes.
- Submit a report introducing your algorithms and presenting the experimental results. We will upload an example.
- Set a fixed random seed in the implementation to obtain the same results as those in the report by running the submitted codes.

Universiteit Leiden
The Netherlands

Tuning Hyperparameters

- Goal: we want to get good performance upon **5,000 function evaluations on each problem.**
- How we evaluate your algorithm: we will execute it for **20** independent runs and take an aggregated performance value.
- Now, I give you a much larger tuning budget, **100,000** function evaluations.
- You can use any tuning method you can find or come up with.
- The tuning method consumes maximally **100,000** function evaluations and outputs a hyperparameter setting, which makes the GA perform well on both problems with a budget of 5,000 function evaluations.

Think about..

- Which variations (i.e., mutation and crossover) and selection operators will you use?
- How to encode/decode the phenotype of the N-Queens problem?
- How to tune the hyper-parameters (e.g., population size, mutation rate, etc.)?
- How do you allocate the tuning budget?
  - Remember we want to make the GA work well at a fixed budget of **5,000 function evaluations.**
  - GA is random… meaning one single run is not representative of the quality of a hyperparameter setting
  - How do I tune for two different problems at the same time?

General Info

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