NaCo 21/22 assignment report, group number

First and last name of each student in the group

Leiden Institute of Advanced Computer Science, The Netherlands

Abstract. This document contains the instructions and the format for the report required for submission of the practical assignment for the course Natural Computing.

1 Introduction

This document serves as a *description of the practical assignment* for the course Natural Computing in academic year 2021/2022. For this assignment, we would like you to write a detailed description of the Genetic Algorithm (GA), and provide an implementation in Python. For this assignment you are asked to provide:

- implementation of a GA in plain Python,
- report, written and formatted as a *scientific paper* containing:
  - description of the GA,
  - description of your implementation,
  - literature review of the various search operators used in the GA,
  - literature review of an application of a Genetic Algorithm.

To help you structure your report, we provide you with a *brief report outline* in this document. Please complete the following sections with your own results, explanations and conclusions. This includes the abstract and this introduction! For this section: introduce what the paper is about and provide a background to any relevant literature, such as the Genetic Algorithm [2].

2 Algorithm Description

Give a general overview of the working principles of a Genetic Algorithm (GA). For assignment part 1, you will implement your own version of a GA, and provide a description of your implementation in this report. Try to use clear mathematical formulations wherever possible. Make sure this description is *complete*. It should explain all core concepts, such that one should not need to refer back to the original papers to get the global picture of the working principals. For your implementation, we would like you to focus on *modular* design of the search operators (see Section 3 on what is meant by this).

In the report, we would like you to describe at least two variants found in the literature for each of the following search operators; *recombination*, *mutation*, and *selection*. We also would like you to discuss two different ways a candidate solution can be represented in the GA in the section on *representation*. You do not need to implement all these operators in your code[1], but you are

[1] You do need at least one version of each operator implemented to be able to run your algorithm.

encouraged to try them out to gain a better understanding. In your implementation, it should be possible to replace the operators with any of the variants from the literature with minimal changes to the code.

For each of these three operators, in subsequent subsections, provide an overview on the general methodology of the operator and explain how and why it can be used in the GA. For each of the two operator variants you found in the literature, describe it and how it works. In addition, provide a description about the differences between each operator variant, and discuss when it might be beneficial to use one or the other.

2.1 Recombination

(see above)

2.2 Mutation

(see above)

2.3 Selection

(see above)

2.4 Representation

Give a description on how search points are represented in a GA, i.e. describe the genotype of the canonical GA. Additionally, describe alternative forms of representation, and discuss how you would implement this in your algorithm [2].

2.5 Notation

Make sure to follow the following notation conventions:

- n: The dimensionality of the search space
- M: Number of individuals in set/array
- $\mathbf{x} = (x_1, x_2, \ldots, x_n)$: A solution candidate.
- $\mathbf{x}_i$: Solution candidate i (for $i \in \{1 \ldots M\}$) in the set/array
- $f(\mathbf{x}_i)$: Objective function value of $\mathbf{x}_i$ ($f : \mathbb{Z}^n \to \mathbb{R}$)
- $\leftarrow$: Assignment operator
- $\boldsymbol{\mathcal{U}}(\mathbf{x}^{\min}, \mathbf{x}^{\max})$: Vector sampled uniformly at random. Here it is 'U' for uniform. For other distributions, use for example $\boldsymbol{\mathcal{N}}(0, 1)$ for a single number sampled according to the *normal* distribution with mean 0 and variance 1.

If you need to use any other notation, please be consistent and clearly define your added notation. In case of doubt, feel free to contact the TAs.

[2] i.e. the representation will be binary, but how would changing this, to for example a discrete representation impacts the algorithm

3 Implementation

You should implement your algorithm in Python, using the IOHexperimenter [1] package. A skeleton implementation is provided on Brightspace as an attachment to this project description in the file `implementation.py`. You can follow the steps in the provided README to install the required dependencies and run the test functions. The skeleton implementation provides an example of running the random search algorithm on a OneMax objective function. You can use this for testing your GA as well. In case of issues, please do not hesitate to contact the TAs for assistance.

When coding your algorithm, make sure to not use any hard-coded values. Instead, all of the parameters you use should be modifyable by the user. Include a table in your report here, describing all parameters of your implementation, their default values and their range of accepted values.

As mentioned earlier, we want you to focus on a *modular* design of the search operators. What we mean by this, is that we would like you to be able to switch out a given operator variant for another, without breaking parts of the program and with minimal changes to your code. With this we do not mean replacing a *mutation* operator by a *recombination* operator, but replacing a *mutation* operator for another *mutation* operator, which performs mutation in a different fashion. In addition, your implementation should consider alternative forms of *representation* for the search candidates.

To verify whether your algorithm works correctly, call the function `test_algorithm` provided in the notebook. **Passing this test is required for submission**, so if your algorithm fails this test, please contact the TAs.

4 Application of GA in practice

Search in the literature for one paper with an application of a GA outside of the field of computer science. Minor students are encouraged to take the lead in writing this section. Summarize the paper and describe the application, and add any relevant literature. In this section, be sure to answer at least the following questions about the paper:

- How was the GA applied? Describe the field and context.
- Why did the researchers choose a GA for their application, and where there any alternatives?
- Was their approach successful? Interpret their results.
- Give your opinion on their approach. Would you have you have used a GA in their situation?
- How would you improve on their setup?

5 Part 2

The description for the second part of the assignment will be provided after the deadline for submission of part 1.

6 Conclusion

Write a short conclusion summarizing the most important findings of this assignment.

A Appendix

A.1 Submission, review and grading

This assignment consists of two separate submission stages and a peer-review.

For *the first part of the assignment*, you should submit the sections included in this template. You should also include a brief introduction and abstract, which are *not* the same as the final version, as you do not yet have all the content. Please be sure not to exceed the **soft limit of 6 pages** for this report, excluding references.

Submission should be done on Brightspace, and should include your report in PDF format and your code as a Python file in a single zipfile. Make sure that the code is readable: clear variable names, comments,. . . .

You will then receive feedback on the submission and no grade. Make sure to incorporate this feedback in the submission for part 2, as this is only aimed to help you improve your final report! The deadline for this first stage is **13/10/2021**.

References

1. Doerr, C., Wang, H., Ye, F., van Rijn, S., Bäck, T.: Iohprofiler: A benchmarking and profiling tool for iterative optimization heuristics. CoRR **abs/1810.05281** (2018), http://arxiv.org/abs/1810.05281
2. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, MI (1975), second edition, 1992