CE310 – Evolutionary Computation and Genetic Programming (Spring 2020/21)
Reinhold Scherer

Part 1 - Implementation of the basic functionality for Evolutionary Algorithms

>> Task definition
The objective of part 1 is to familiarize yourself with **genetic algorithms** (GAs). Design and implement the core functions needed for **a steady-state binary GA** to solve combinatorial optimisation problems. Software architecture, design and development decisions are entirely up to you. The only requirement is the use of Python. Genetic operators that **must** be implemented are: **1-point crossover**, **flip bit mutation**, and **tournament selection**.

Note that you do not need to install additional libraries. Software libraries included in the standard Python installation are sufficient. You can use *Matplotlib* to visualise results or *NumPy* if you plan to implement more efficient array operations.

>> How to test your implementation?
Implement the following problems to test your implementation:

1. **Maximise the ones**: Test your implementation by creating individuals that are represented as 100 bit long arrays/strings/etc. and evolve the solution with the aim to maximise the number of ones in the representation.

   To get a better picture about the GA performance, please experiment with the GA **hyper-parameters** such as population size, generations, crossover rate, mutation rate and tournament selection size. Start with standard values that we discussed in the lecture. You can keep values that you find most appropriate as basis for solving problems 2 and 3.

2. **Knapsack Problem:** Design and implement the representation and the fitness function and solve the Knapsack problem. Please select meaningful configurations for the problem.

3. **Function optimisation**: Explore the impact of different representations and fitness functions on the performance of your GA. Be sure to check whether you have a maximisation or minimisation problem.

   Pick at least one single-objective and one constrained optimisation functions from https://en.wikipedia.org/wiki/Test_functions_for_optimization and implement them as your fitness functions. Then run simulations to find the minimum (or maximum, depending on the problem) of the function.

>> Free choice of additional tasks
You can get even better acquainted with GA by solving additional problems, do more in dept-analysis. For example**:**

- **Function optimisation**: You can explore multi-objective fitness functions and/or implement a real-valued GAs
- **Time to move:** Design and implement the representation and the fitness function that solves the combined N Bin Problem and Knapsack problem.
- **Come up with your own optimisation problem:** Find or come up with your own combinatorial optimisation problem one and solve it.
- Do a more **in-depth exploration** of the **GA hyper-parameters**
- **Impress me with your creativity and exploratory skills**

>> What must be submitted to FASER?
Submit your Jupyter/iPython notebook including the code, results, and a brief description of solution representation, **fitness** function and **observations** for each problem the you solve to FASER. Finish the document with a short reflection on GA. Use a few whole sentences and not just key words for statements. Do not forget to include your name. **The Jupyter/iPython notebook is what will be assessed. We will run the code** (including changing the **fitness** function in the "Maximise the ones" problem) **and read the observations**

Marking criteria (50% of coursework):

- Maximise the ones: 40%
- Hyper parameters: 15%
- Knapsack Problem: 5%
- Function optimisation: 15%
- Free choice: 25%