Lab 13: Evolutionary Algorithms

This assignment is due to Gradescope by the **end of lab today**. You may work with a partner on this lab – if you do, submit only one solution as a "group" on Gradescope.

Introduction

The goal of this lab is to implement a genetic algorithm that evolves 2D soft robots to move as quickly as possible in one direction. If you recall the soft robotics video from class, this task is very similar, but in 2D instead of 3D.

The robots you will evolve consist of 8x8 grids of squares. Each square is either "bone," "ligament," "muscle A," "muscle B," or "None." The genotype representation is (and **must** remain) a 64-element list of global constants `BONE`, `LIGAMENT`, `MUSCLE_A`, `MUSCLE_B`, and `NONE`. The indices of this list correspond to the squares in the robot. For example, if a genotype started with `[BONE, MUSCLE_A, ...]`, it would mean that square 0 (lower left corner) was made of bone, square 1 (bottom row, second from left) was muscle A, etc.

The physics simulation used for this lab is from https://github.com/mikemarek/soft-body-physics with modifications for Python3 compatibility & memory optimization.

Provided Files

- `Lab13.pdf`: This file
- `Evolution.py`: The Python file that you will modify
- `SoftRobot.py`: The Python file implementing the robot simulation
- A `RobotSimulation` directory with other simulation and physics files

Instructions

The robot simulation, genotype representation, and fitness function have already been implemented. Your task is to complete the marked functions in `Evolution.py` to implement genotype creation, mutation, crossover, and selection.

To run the program, execute `python Evolution.py [num_generations]`, specifying the number of generations over which the robots should evolve. You can see the available optional commandline arguments by running `python Evolution.py --help`

Deliverables

Submit your final version of `Evolution.py` to Gradescope.

Grading

Your grade will be based on the following:

- 10 pts: `random_genotype()` function
- 10 pts: `mutation()` function
- 10 pts: `crossover()` function
- 20 pts: `selection_and_offspring()` function

Optional Extra Credit

If you are curious, try modifying the parameters at the start of the `SoftRobot` class in `SoftRobot.py`. These control the environment and materials simulated and can have a substantial effect on the robots. If you find a combination that works significantly better than the provided defaults, show Prof. Apthorpe for a few points of extra credit in the lab portion of the course.