University of Konstanz Winter Semester, 2025
Cyber-physical Systems Group Assignment 4
Group member name(s): Benedict Finsterwalder, Florian Shabani
Group member UID(s): 1564941, 1587539

INF-16580: Evolutionary Robotics

Introduction

In this task we updated the simulation from task 2 to instead evolve its own controller. The goal is to have the robot explore as much area of the world. To achieve it, we implement a hill climbing mechanism on the weights of the proximity sensors. We start with random weights and on each iteration we randomly adjust some of the weights and only accept the change if it leads to a higher score on the simulation. This process iterated for a number of generations.

Solution Method

We implemented the algorithm in our repo Github. First we generate an agent with a random genome of length 6. This is going to represent the weights for the linear equation of each of the proximity rays. The robot will then turn to the right based on the result of the formula

Let g_i be the genome values

$$v_l = g_0 * s_l + g_1$$

$$v_r = g_2 * s_m + g_3 + g_4 * s_r + g_5$$

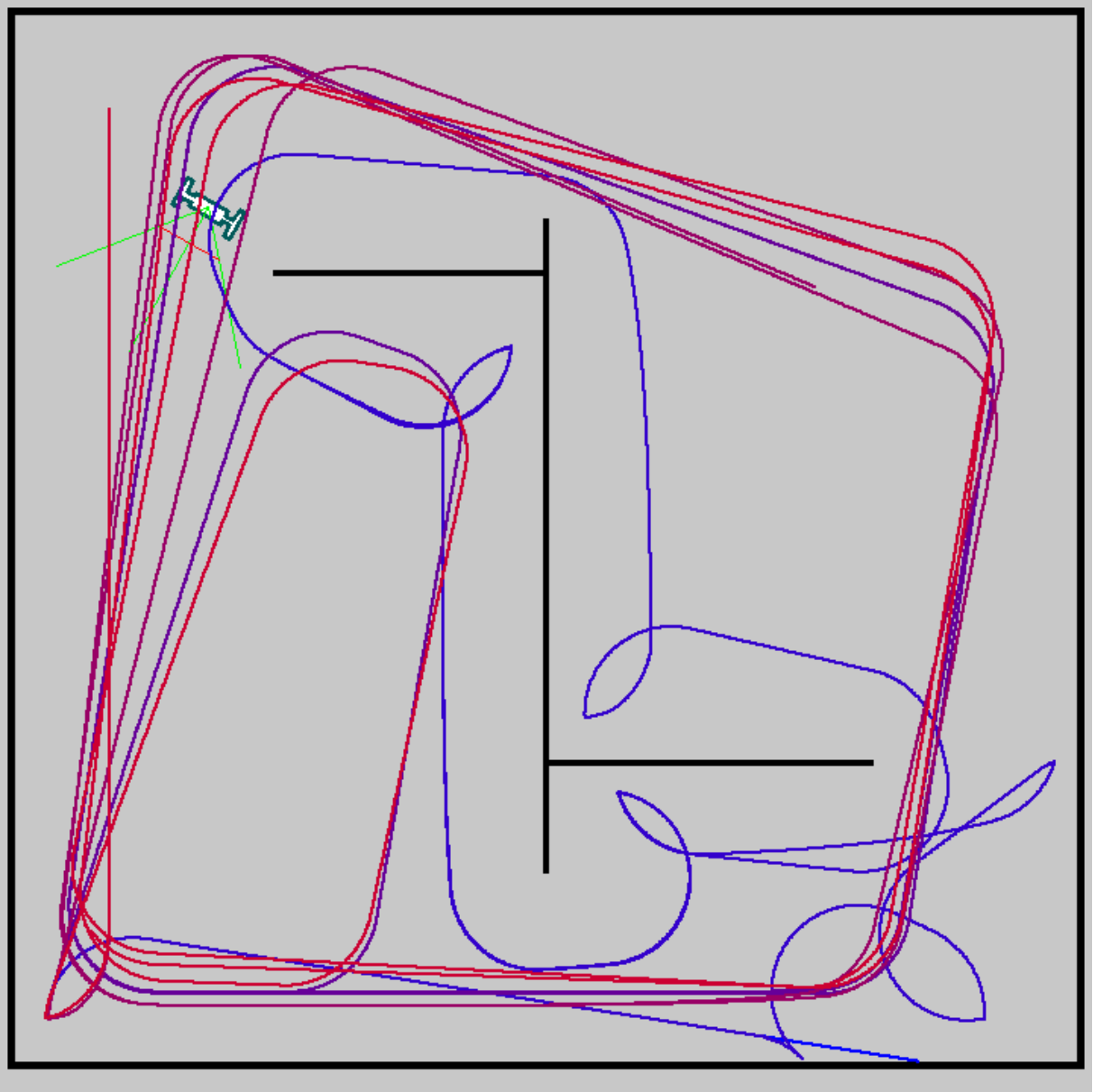
$$turnRight = v_r - v_l$$

Then on each iteration, for each value of the genome we update it with a certain probability. The updated genome is then evaluated. The experiment is run and the robot has some time to explore its world.
Based on the simulation, the fitness function is then calculated by checking the trajectory of the robot and counting the number of unique grid cells of the world it has been on.
The updated genome is accepted only if it achieved a higher score than the previous phenotype. Otherwise we revert to the old genome.

Results

The results generally varied a lot. It looked it was very hard for the robot to finetune the weights and most of the time the robot ended up spinning in circles.
To produce better results we varied a lot of parameters, such as: the probability of a weight to be updated, the amount it updates for, and the size of the grid cells which played quite an important role. Greater grid cells forced the robot to deviate from the spinning strategy and forced it to look for further areas. However, having grid cells too big made the robot more "rigid" and follow only straight paths and only avoiding walls, which was not exactly what we were looking for.

In the image we show the paths our "best" agent took and how it evolved. We filtered only the ones that had improvement to reduce clutter. The paths that are more blue appeared later in the simulation.
The simulation ran for 100 generations



We think that the design of our world made it harder for the robot to learn, especially at the beginning. The barriers are floating in the middle, not touching any walls and they punish the robot quite quickly if it hits one of the "spikes". The robot mostly ended up avoiding walls as much as possible and not sticking to them for guidance to exploration.

Conclusion

The hill climbing algorithm worked somewhat okay for this problem. The experimental results showed a lot of variability, but still noticeable learning of the robot toward the desired behavior. Different worlds might have given different results.