1 Getting started

Last week we worked on using an evolutionary algorithm to optimize the PID controller from the previous lab report. This week, we will be evolving a neurally controlled car that can complete a round around a track. In this case, we will again be evolving parameters, but now the parameters are the weights of the neural network that controls the car. Again, your lab report on evolutionary robotics will include results from all three weeks, so keep detailed notes!

1.1 Software prerequisites

Make sure you install `neat_python` and all the libraries listed in the `requirements.txt` file (although your IDE may do that automatically). Download the required files from Brightspace, we will be working in this code base.

1.2 Documentation prerequisites

Look up the `neat_python` documentation and browse it through to get some familiarity with its classes, methods and functions. Specifically look at the specifications for the configuration file.

2 Your analysis

2.1 Initial run

Use the default settings to run the code using `python Game.py`. Note how many generations it takes for the best member to reach the fitness threshold. Implement a visualization method where you plot maximum and average fitness against the number of generations. You can find an example implementation in the documentation here. Note that the code produces animated GIFs to show your population on the track for each generation.

2.2 Mutation parameters

Look at the configuration file, specifically at the mutation rate parameters. By default, there is no mutation of network topology. Try manipulating the `weight_mutate_power` and `weight_mutate_rate` parameters, and look at their effects. Visualize their effects, and describe what they do exactly.

One of the advantages of NEAT is that it can also evolve network topology. Look in the documentation for the `node_add_prob` and `node_delete_prob` and `conn_add_prob` and `conn_delete_prob` parameters. Describe what they do, and investigate the effects of manipulating these values on speed of evolution.

2.3 Population size

The default population size is 10, which may be somewhat small. Try your manipulations above with larger population sizes.

The default elitism value in `neat_python` is 2, which may be too small for larger population sizes. Change the elitism value accordingly. Of course, carefully describe the values that you use!

2.4 Other variables

Think of two other variables (that do not necessarily have to be hyperparameters) that can affect the speed of evolution. Describe these two variables, manipulate them, and carefully study their effects. Use visualizations!

3 Submission

This is the last session for evolutionary robotics. You will submit your lab report after this week. It will consist of your work from all three weeks. Submit your code to a separate assignment that is able to accept .py files. Instructions will be given on Brightspace.

Good luck with this week's assignment, and remember: we are here to help you if you get stuck.