Evolutionary Computing Project Proposal: Group 9

**Introduction:**

Snake is a game concept in which the player controls a “snake”, usually represented by a “line” that grows in length as the game progresses. The snake is situated within a bordered playing field and the goal is to “eat food” that appear randomly within this playing field. As the snake eats these items, it grows in length, making the game progressively more challenging. The game is over and the player loses when the snake collides with either the screen boarder or an obstacle (including parts of the snake’s own body). The aim of the game is to try and increase the length of the snake to the point where it fills up the entire screen, or simply beat the previous high score (determined by the length of the snake).

**Neural Network:**

In this assignment, we aim to create a neural network that is capable of learning to play the game Snake. The neural network will need to be trained to learn the best way to traverse the playing field in order to collect the food. The snake can travel in only 3 directions: forward; left; and right and has the ability to “sense” the location of the food. The snake will struggle to reach the food initially as the neural network’s weights are set to random values. As the network is trained (via a genetic algorithm) however, the snake will learn how to find the food better without causing collisions.

Our objective is to maximize the length of the snake. The snake’s length will only increase, when it “eats” the food. Hence, we also require information about the location of the food. This means that we have the following inputs to our NN:

• X distance to food from head

• Y distance to food from head

• What is on the left of the head

• What is on the right of the head

• What is directly in front of the head

Taking this information into account we will have 3 feed-forward neural networks (one for each of the snake’s movement options), trained by a genetic algorithm, that will have a single output. The output is a rating of each move, and the snake then choses the movement with the best rating. The best rating will be determined by the decreasing distance to the food, without hitting its own tail, or the wall.

**Genetic Algorithm:**

The Genetic algorithm will make use of mutations and crossovers. In the case of Snake, if a run was to fail on a solution (hit the wall or its tail), or the run was unable to find food / “stall” over a large number of generations (e.g. 100), then a new population will be generated. This population will consist of the best solution previously found (the stalled / game over solution) and would be measured by the average fitness function of the stalled/failed individual over four runs – this will eliminate a solution chosen due to a ‘lucky’ placement of food. The rest of the ‘new initial population’ will contain individuals derived from the previous best solution, but altered using crossovers and mutations, which could potentially present an individual that would benefit the previously stalled or failed individual.

**Fitness Function:**

The simplest fitness function to implement is setting the fitness to be the length of the snake achieved before dying. This fitness function would be the most effective in order to create an algorithm to “win” the game.

It is unlikely that the time to completion would be optimized; as the objective of the game is to get the length of the snake to be as long as possible, and not to do this in the shortest amount of time. IF time were to be optimized, then the fitness function must reward both length and the number of moves to completion (with smallest number being preferable). A Possible fitness function for this would be:

Where beta would scale the importance of the moves to completion. This function would be maximised as a low moves to completion and a large length will lead to a large fitness.

However, this could allow for 3-move-games (the minimum number of moves till death given an appropriate starting length) as now the time to completion would be very small, and the system could think that this sort of behaviour is allowed, or appear fit.

A possible solution to fix this problem (besides the scaling factor) is to square length which would reward more for every increase in length or square rooting the moves to completion (which is analogous). Another solution would be to code a minimum length before the moves to completion is considered.

It is, however, far simpler to disregard the moves to completion factor, and assess the performance of the algorithm based on the length of the snake alone, as that would provide the basic functionality required to play the game in the most effective manner.

**The Group:**

The group is comprised of 2 BSc Honour students and 3 mechatronic engineering students. The project is divided evenly amongst group members according to the strengths of each individual. We will have an agile approach to development that will be run in sprints with designated tasks, this will be conducted and delegated through version control software (GitLab). Compulsory project meetings are to be held once a week in which to update group members of the current progress as well as to discuss any problem areas. Some of the tasks will include:

* Creation of the snake game
* Implementation of the neural network
* Implementation of the Genetic Algorithm
* Optimizing the neural network
* Optimizing the genetic algorithm(crossovers and mutations)
* Collection of results

The role of each member is as follows:

1. Adriaan Wildervanck: Group Leader, Programmer (Neural Network & Game Implementation)
2. Dillon Scott: Programmer (Genetic Algorithm Implementation)
3. Gareth Meyer: programmer (Optimizer of GA fitness function)
4. Ashely Mackinnon: Programmer/Final Document Writer
5. Ridwaan Karolia: Programmer/Final Document Writer

As previously mentioned, tasks will be delegated to those that feel comfortable with that section of the work, but ultimately the final product will have been seen and reviewed by each group member.