COSC349 – Assignment 2

Brooklyn Taylor

4308085

## Introduction

This report outlines my approach to implementing a genetic algorithm (GA) to evolve snake agents in the *Snakes on a Grid* environment. The goal is to design a model where snakes learn behaviours and show evidence of fitness improving over time, evolving to prove best behaviours are propagated in the final population and how they contribute to better outcomes.

## Genetic Algorithm Methodology

The agent uses a Genetic Algorithm to evolve better chromosomes across generations. In my implementation of a genetic algorithm, I decided to expand upon these variables and test them.

|  |  |
| --- | --- |
| * **Elitism count** | *number of top snakes that are transferred into the next population* |
| * **Tournament size** | *number of snakes selected for tournament selection* |
| * **Crossover probability** | *chance that we perform a crossover* |
| * **Mutation probability** | *chance that each gene will be mutated* |
| * **Mutation deviation** | *size of the random nudge during mutation* |
| * **Weight clip limit** | *limits of mutation range* |

### Chromosome Representation

Explain how weights/biases are flattened into the chromosome.

Justify why I chose the simple perceptron (small search space, quick convergence).

### Agent Function / Neural Model

The agent function is implemented as a simple **single-layer perceptron**. The snake perceives it’s surroundings through a **7x7 local grid** with49 percepts, each percept showing friend, enemy, food, or empty cells.

The chromosome contains the perceptron’s parameters:

* **Weights**: 49 x 3 = 147 values
* **Biases**: 3 values
* **Total chromosome length**: 150

During each turn of the game, the percepts are flattened into a 49 element vector and multiplied by the weight matrix, then shifted by the biases. The action with the highest output score is then selected for the snakes next movement. This ensured that the behaviour is entirely chromosome-driven rather than hardcoded behaviour.

### Fitness Function

In order to take in as much information as possible from the game environment, I decided to expand the fitness function to also consider the snakes actions and behaviour. By adding a reward/penalty modifier to some of the snakes previous actions, I had hoped to sway populations away from undesirable behaviours.

This worked by determining the statistics of each snakes behaviour in the previous game, such as the amount of food eaten, enemies attacked,

#### Profiles

To determine how these modifiers could affect snake behaviour I implemented some **snake** **profiles**. These are simply predefined sets of values for the fitness evaluations for purposes of having different types of snakes to test with.

*Example of an “predator” snake.*

A screenshot of a computer screen

AI-generated content may be incorrect.

*Example of a “pacifist” snake.*

A screen shot of a computer

AI-generated content may be incorrect.

Each behaviour was penalised and rewarded accordingly, this exhibited some emergent behaviour such as running towards or away from enemies depending on the snake profile selected. Though its hard to demons

### Selection Method

Tournament selection explanation & why (balance between exploration/exploitation).

The selection method

### Crossover

1-point crossover implementation.

Why a simple crossover was chosen (low complexity, clear DNA mixing).

### Mutation

Gaussian mutation details (probability, sigma).

How mutation rate affects exploration.

### Elitism

How many elites are carried forward and why.

### Training Schedule

Opponent(s) used (random vs self).

Number of generations planned.

Justification (fast training, clear results).

## Snake Performance Analysis

Here is where I describe what I’m testing and why

### Fitness Over Generations

### Plot average fitness per generation.

### Behavioural Observations

Qualitative notes from the visualiser.

Early gens: random, chaotic movement.

Later gens: snakes seek food, avoid tough enemies, fewer suicides opohros.

### Parameter Sensitivity (if time allows)

Test how different mutation rates, crossover rates, or population sizes affect learning speed, maybe compare 0.05 vs 0.1 mutation probability.

### Training Schedule Comparison

200 gens vs random agent (keeps within 500-gen cap).

Some profiles were tested with 500 against random, need to do mixed testing

Discuss possible alternatives: self-play, staged opponents (random → self → mixed).

## Results

Recap of what change*d over time, what improved or didn’t*

*Any weird outcomes?*

To evaluate the impact of different GA configurations and fitness modifiers, I ran five test scenarios with 200 generations of training followed by a 5-game tournament against the random agent.

* **Test 1 (baseline)** used a modest food reward and standard GA settings. It achieved average tournament scores around 195, showing strong performance overall.
* **Test 2 (more elitism and stronger selection pressure)** produced lower average fitness values (7–9) and tournament scores around 74, indicating premature convergence and loss of diversity.
* **Test 3 (food-first profile)** increased average fitness (15–18) and rewarded food collection heavily, but tournament scores averaged only ~108. This suggests that food-seeking alone was not a dominant winning strategy.
* **Test 4 (balanced skirmisher)** struck the best balance. With moderate food reward, stronger penalties for friendly crashes, and a reasonable bonus for enemy aggression, it consistently reached late-generation average fitness values of ~13–15 and tournament scores around ~180, the highest competitive success among all tests.
* **Test 5 (aggressive vs enemy)** encouraged hostile play but reduced stability, leading to average tournament scores of ~154, below the balanced configuration.

From these results, I selected the **balanced skirmisher configuration** as the final setup. This combination of GA parameters and fitness shaping provided the best mix of exploration and exploitation during training and produced agents that consistently outperformed the random baseline with both higher fitness values and the strongest competitive scores.

The outcome highlighted the importance of balancing the reward/penalty signals. Over-prioritising a single behaviour (e.g., food collection or aggression) led to weaker tournament play, while blending the reward functions produced more robust and effective strategies.

## Discussion

Highlight **limitations**:

* Single-layer perceptron = only local, linear decision-making.
* No memory of past states → can’t learn “strategies” like running away or planning multi-turn traps.
* Behaviour often appears noisy due to high environment randomness + simple policy.

Reflect on **what worked**:

* GA reliably improved average fitness.
* Reward shaping clearly influenced behaviours (food vs aggression trade-off).

Reflect on **what didn’t**:

* Emergent behaviours not always interpretable as “strategy”.
* Randomness + limited model capacity → snakes lack consistency in visible habits.

Possible **future improvements**:

* Add a hidden layer (non-linear capacity).
* Add memory inputs (previous action, orientation).
* More sophisticated fitness shaping.
* Training schedule with self-play.

## Conclusion

## Appendix

**Planned Concepts to Cover in the Report**

Below are the things I’m going to cover, if anyones reading this this is just a template or something of stuff I plan to cover

**Perceptron**

The perceptron will be described as the “brain” of the snake. a weighing scale that decides whether to move left, forward, or right. My plan is to show how the percept grid gets flattened into numbers, and then how those numbers get weighted through the perceptron to choose the best action.

Gonna explain that the weights and biases are not fixed etc., but are determined by the chromosome, so different snakes behave differently. Instead of them all acting like lemmings

**Chromosome**

The chromosome will be presented as the “DNA” of the snake, a flat list of numbers that encodes the perceptron’s parameters (maybe even a cool DNA model with numbers or something idk).

Going to explain how this representation lets snakes have different behaviours and how the genetic algorithm can copy, mix, and mutate these chromosomes to create new strategies.

I’ll insert some an example of how a chromosome might be mapped into weights and biases.

**Tournament Selection**

Tournament selection will be mentioned with some analogy of a sports competition or something, a few snakes are randomly picked, and the best of them goes forward as a parent. I plan to illustrate this process with a simple diagram or description, showing how it ensures stronger snakes are more likely to pass on their traits but weaker snakes still have a chance.

**One-Point Crossover**

Crossover will be described as cutting and stitching two DNA strands. I plan to show how one parent can contribute the beginning of a chromosome and the other parent the end, producing a child with mixed traits (theres some lecture slide with a good example). The aim here will be to explain how this mixing allows successful strategies to combine in new ways.

**Gaussian Mutation**

Mutation will be presented as small “typos” in the DNA, where random noise is added to some genes (maybe that DNA thing mentioned before with some highlighted changed values). I plan to explain this using the analogy of nudging weights slightly up or down, which allows exploration of new behaviours, eliminating lemming behaviour. I could highlight that without mutation, the population could stagnate and fail to find better strategies, lemmings.

**Elitism**

Elitism will be described as protecting the best individuals so they always survive into the next generation unchanged. I plan to show how a few top snakes are copied directly to the new population before breeding the rest, ensuring that good solutions are not lost due to randomness in crossover or mutation. <- google found description that im going to paraphrase in the tournament section

**Training Schedule**

Finally, I will describe the training schedule as the “calendar” of practice matches for the snake population. Explaining how I can set the population to train against random opponents, against itself, or in stages, and that the schedule determines how many generations are run. The report will include how this schedule affects the learning process and how I chose my own schedule for testing. Will still need to determine how I show this effectively.

## Behavioural Notes:

### First GA implementation

* When I first implemented the agent with some genetic algorithm, the snakes behvaiours seemed less deterministic and more random (despite winning against the random agent after a training schedule of 200 against random). This seems to be due to two main things, the single layer perceptron not allowing for memory or forming rich patterns like “if enemy is approaching from ahead-left, turn right twice..”. only mapping the current 49 numbers to an action linearly.  
    
  and the second being due to the fitness only rewards “being big on average”. This doesn’t account for pressures to consistently chase food, avoid friendlies, or pick fights. Only to survive long enough that size drifts upward. So the easiest strategy is “wander without dying too fast”
* Without a training schedule (running straight from the get go) the snakes either spin in circles or chase straight lines, mimicking the random snakes. This is just due to the random chromosomes adding no real weights or biases, since no evolution has emerged, and therefore acting in random manners.

#### Results

Game 1 – score 143

Game 2 – score 52

Game 3 – score 75

Game 4 – score 36

Game 5 – score 94

### After Adjusting EvalFitness Function

* There was a bit more cannibalism, though some defined behaviours were more emergent (chasing enemies, seeking food, etc.)
* The amount of random behaviours is still annoying, with less than half doing something useful while the others just spin in circles or attack friendly snakes. This could be either due to the mutation factor, or the need to increase penalties/rewards

#### Game Results

Game 1 – score 224

Game 2 – score 164

Game 3 – score 148

Game 4 – score 133

Game 5 – score 190