COSC343 Assignment 2 Report

Jayden Prakash 4718680

Evolve a species in a game

Agent Function:

Initially my chromosome was initialised with the percepts and actions, so it was an array with shape 5,75. This mapped well to percepts, as when the percepts are flattened it is a 1x75 array, and there are 5 different indexes for actions so for each index of action I got the dot product of the first array of chromosomes and the percepts. However this initial implementation did not seem to work well so I decided to rework it.

Upon reflection of the percepts map that was provided in the assignment pdf I was able to better understand how the actions and percepts properly worked. The current agent function iterates through every map, and for each value of the map it checks the square. For example in the creature map, it will check each square to see if there is a creature there, if there is and the creature is an enemy it will head towards the square at a random chance which is assigned by the randomly initialised chromosome. This is the same for all maps; the creature will attempt to head toward food, away from walls and enemies and eat food if it's on the square. Through the genetic algorithm, certain values of the chromosome will end up becoming larger than others in order to promote behaviour that will ensure the population's victory.

At the start of this model, there was a problem with eating. Since there is only one case where the creature will eat, but 4 cases for all the other actions the creature then began to never eat as the other values in actions were guaranteed to be higher. My fix for this was; if a creature is on a square with food, over every iteration the creature will be incentivised to eat, however this now caused a problem where the creature would always eat if there was food which counted as hard coding. I offsetted this by dividing the value of the chromosome added everytime by 3, this turned out to be perfect as it allowed randomness but also made sure that the creature would eat. When tested, I made the fitness discourage eating and the creature ended up never eating, and when normally run the creature eventually learns on its own to eat when it can.

Chromosome:

My chromosome has 1 dimension and is a list of 17 random floats. This is because 17 numbers perfectly map to the amount of different actions when iterating through the percept maps. Floats were chosen as they result in more variety. While integers look cleaner and might be slightly easier to manage, there were a lot of duplicates and not as much randomness as you would expect.

Initially, when my chromosome was still a 5x75 2D array, the crossover was difficult to fully implement due to the 2-Dimensional structure but now with the 1-Dimensional list it is a lot easier. By association the mutation also becomes a lot easier, and allows me to easily understand the relationships between percepts and actions and how the genetic algorithm actually evolves.

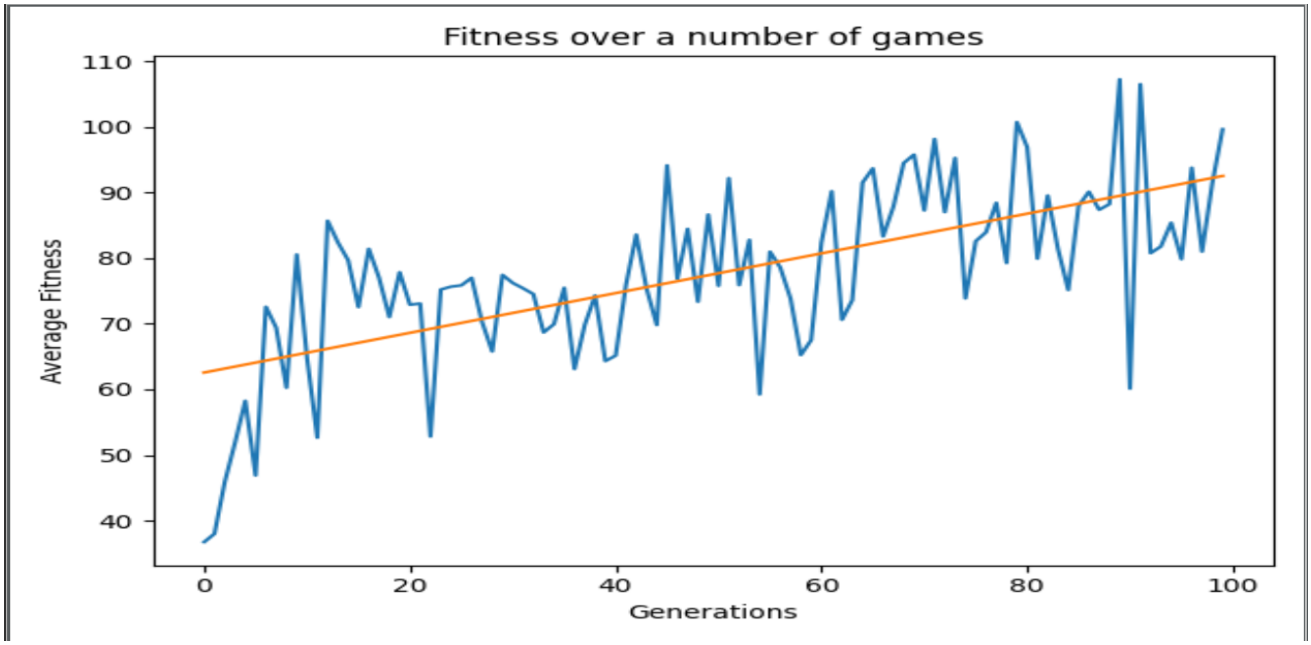
```
[0.97220888 0.39751761 0.10304777 0.30312418 0.47818263 0.76510202
 0.39355811 0.31281251 0.73207244 0.73439755 0.25970275 0.23040294
 0.00238601 0.01663783 0.37128155 0.7112741  0.76693427]
```

Visual representation of the chromosome of a random creature.

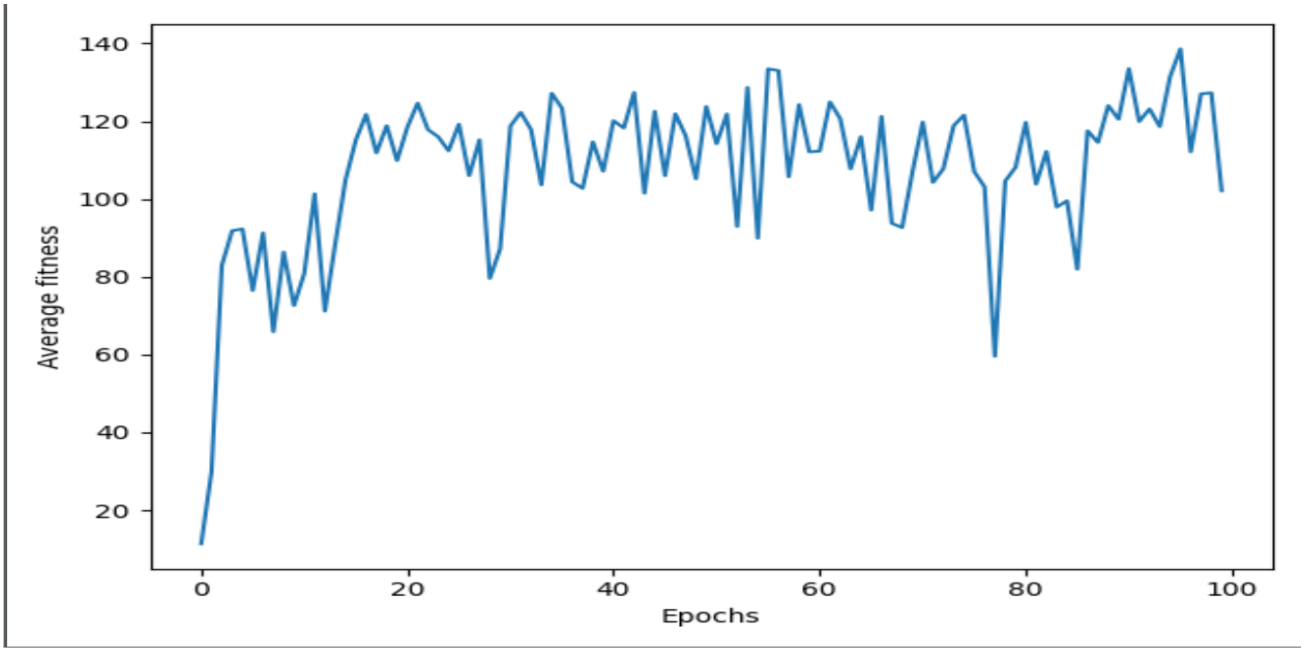
Fitness Evolution:

My genetic algorithm, as expected, does not evolve in exactly the same way on each run through, and because of mutation and randomly initialised chromosomes it can cause various different graphs. Nevertheless, all graphs produced show a clear upward trend that indicates that the genetic algorithm is actually evolving.

Here is one of the graphs produced:



As you can see in this graph which follows the creatures over 100 generations, there is a clear upward trend. Looking at the trend line, we can see that on this iteration the genetic algorithm learned gradually and over the generation became more and more fit at a consistent rate. However it is not always the case.



In this graph you can see a much sharper incline. The population evolves rapidly in the beginning, and then reaches a plateau, likely unable to evolve too much further due to the limited food available on the map, as the current fitness function emphasises eating over all other things. However it is good to see that it reaches its peak quite fast, at least against the random player which means it does not take too many generations for the population to evolve past the behaviours of a random player.

Other than emphasising eating strawberries and other creatures, the fitness function also takes in the different squares a creature visits in order to promote exploration, and also rewards creatures that achieve a big size. The fitness function is offset if the creature dies. In that case the fitness of a creature is divided by 1.3. This number was used as it wont offset the fitness too much as having creatures that die early are important in terms of diversity but it is enough to discourage behaviour that is detrimental to the population. The offset is used because initially with a heavy emphasis on eating enemies, many creatures that weren't big enough would run into creatures much larger than them to attack and would end up dying themselves. The intended goal of this offset, is to encourage the population to only attack if they are big enough to win the fight.

Genetic Algorithm:

The first part of the genetic algorithm was selecting which creatures to use as parents for the new generation. Initially, the population is sorted in order of fitness, it is then split in two and only the upper half of the population is retained. When picking creatures, elitism is used for the first few. The first 3 creatures in a generation are a direct clone of the fittest creature of the last generation.

The remaining population is then selected using tournament selection. The upper half of the population that was split earlier is first taken as a parameter. Then from this population, 5 individuals are randomly chosen and out of those 5 individuals, the two most fit individuals are returned by the function. When this was first implemented, the tournament selection function was actually choosing the exact same creature for both parents which caused a problem as many of the new children were simply clones, to fix this a boolean variable is used that specifies if the two parents are duplicates or not; if they are then it will pick 5 new creatures and then choose the two of them that are fittest until the two parents returned are different creatures.

After the two individuals are chosen via tournament selection, they are then used as parameters for the crossover method. I personally used single-point crossover, by randomly selecting an index of the chromosome as a crossover point and then iterating through it, using the first parents chromosome till the crossover point was reached, and then using the second parents chromosome in order to produce the child's new chromosome which is returned by the function

The new child chromosome is then mutated. The mutation method is fed the child as a parameter, and then has a 1% chance of mutating any one creatures chromosome by randomly selecting a gene and assigning it a new float value. Initially my mutation percentage was a lot higher, but I found that it caused too much randomness, the population would learn and improve but eventually, at random points it would suddenly dip and cause the population to forget some of the good behaviours that it learned. I found that using a 1% mutation rate was a good balance as it allowed the population to have a bit more diversity without causing too much randomness.

Retrospect:

Overall, I am quite pleased with the results of the algorithm. In the beginning I found that it was evolving only slightly but then eventually regressing and repeating that process over and over. Now, against a random player, the population seems to evolve rapidly and learn the right behaviours in order to succeed without taking too much time.

While the population can not consistently win against the hunter player, it is still able to evolve and get better and accumulate a few wins. This shows a positive trend and perhaps with a large amount of training it may eventually be able to consistently beat the hunter itself.

With the first generation, creatures just randomly move around seemingly with no purpose and occasionally eat, this behaviour gets them eaten themselves and causes early versions of the population to lose. Through evolution, the creatures seem to “sweep” over the board in a diagonal motion and most of the time eat any food that they encounter in their path; be it another creature or some fruit. This takes away the food from the other population and does not allow them to grow, which means our population will almost always come out on top of the other population.