COSC343

Assignment 2

Report

Sean Kim

6185410

Assignment 2 consisted of a task to implement a genetic algorithm that allowed creatures in a virtual world to learn how to survive over an extended period of time. This virtual world consists of features such as how big the world is, how long the creatures are given to survive, and the type of the world to be used. The size of the world is represented by a grid where creatures are able to move from square to square with each move representing one turn in the world. A smaller world meant that there was less creatures to run the simulation on which resulted in less variations between creatures. A larger world, would create more variations between creatures which resulted in better learning for my genetic algorithm. I have decided to run my simulation using a small world of grid size 24 as well as using a larger world of grid size 48 for comparison which is discussed later in the report. Survival for creatures in the virtual world is represented in turns with creatures that survived the specified number of turns being successful in the given simulation. A higher number of turns would help distinguish the best of creatures from the rest of the population for picking a new population hence 100 number of turns was picked for my simulation.

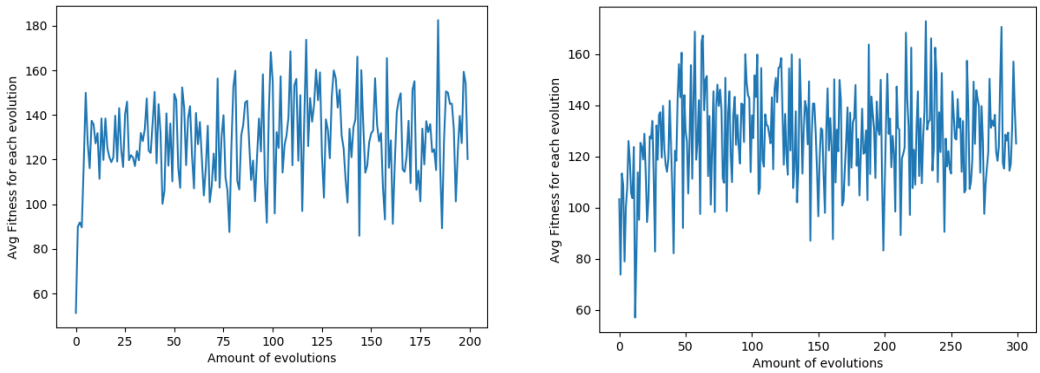
My model of the agent function consists of an action list with all values initialized to 0. It then evaluates its neighboring environment changing the probabilities of carrying out actions based on its percepts. The agent function determines that if a creature is on a green or red strawberry, the chance of the creature picking the next move to eating the strawberry increases respectively. For neighboring squares, a neighboring square consisting of a strawberry will increase the chance of the creature moving to that square. A neighboring square consisting of other creatures or monsters will stop the creature from moving onto the given square. These probabilities are added to the action list which is then combined with the chromosome of each creature to determine which action the creature is to take on the next turn.

I have created my chromosome to pick float values between 0 and 1 which is used to influence the behavior of each creature. Each value represents a percentage chance of carrying out an action that is unique to each creature. When the agent function creates a list of actions, the chromosome values at respective indexes are added to the action list which influences unique behavior between creatures.

My genetic algorithm assigns a chromosome to each creature which consists of a list of probabilities of carrying out actions. This list is modified over generations, picking the most successful parents to produce children using their successful genetics. Success was graded based on the survival and energy of each creature, with the most successful creatures being survivors with high energy levels at the end of the simulation. My selection method picks 5 random parents from the old population, which then selects the parent with the highest fitness level as the successor for a given child. This process is repeated again so that each child has 2 successors. Crossover is applied to the successors in order to create the child. A random cut point is selected in the chromosome with one half up to the cut point being the genes

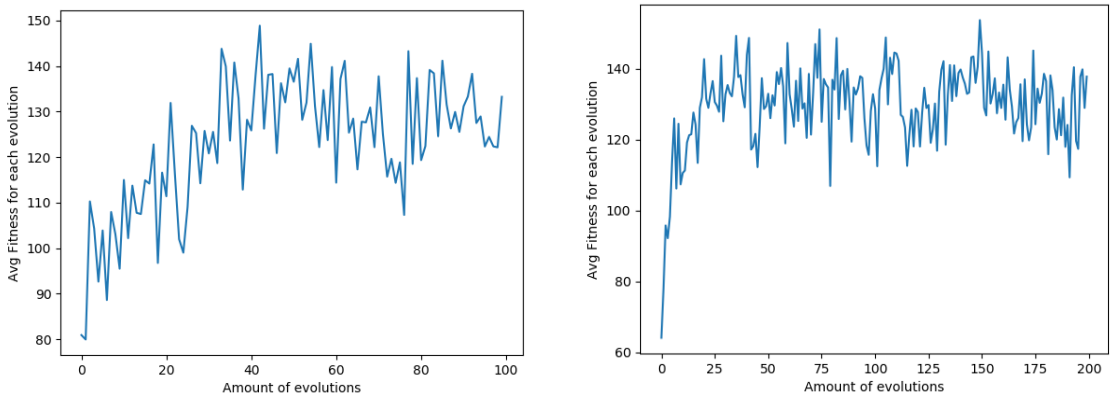
from one successor and the other half being the genes from the other successor. This means that a child can be created using successful genes that aren't the same as its successors. For randomness, each child has a chance for its chromosome to be mutated using the mutation method. The method will loop through each value of a child chromosome with a random chance to mutate it. A mutated value is replaced by a random float value between 0 and 1. This process is used on the old population until the new population is filled up with children.

Due to my genetic algorithm picking the most successful parents with high fitness scores, creatures in future generations adapted movement that was different to the first generation that was random. The most obvious behaviors were that creatures would eat red strawberries almost all the time, as well as run from monsters in neighboring squares. The less obvious movements adapted by the creatures was diagonal movements. Because only creatures can move diagonally and monsters cannot, creatures are able to move at much faster speed if they make diagonal movements. This meant that over generations, creatures were able to adapt to moving faster by making diagonal movements. Creatures were also seen in future generations moving back and forth between squares waiting for green strawberries to turn red.



The left figure shows 200 generations on grid size of 24 with the right showing 300 generations on the same grid.

The graph shows that in the first few generations, the fitness score skyrockets from around 30 to as high as 150. However, after that the graph tends to show a linear trend showing that the creatures learning behavior is capped after the first few generations. I believe that this is due to my genetic algorithm having no way of differentiating successful creatures. After the first few generations, there will be more successful creatures, hence there will be no way to distinguish the best creatures from the good creatures which I believe causes my graph to turn linear in future generations. This made me think what would happen if I was to increase the grid size, creating more variance in the creature behaviors and allowing for more learning within the creatures.



Left figure shows grid size of 48 with 100 generations and the right figure shows the same grid size with 200 generations.

When the grid size was increased from 24 to 48, the graph shows that the creatures are able to better learn, increasing for many more future generations compared to the grid size of 24. Making the grid size larger worked better for my genetic algorithm as the learning cap was increased allowing for more time to distinguish successful creatures from unsuccessful ones.

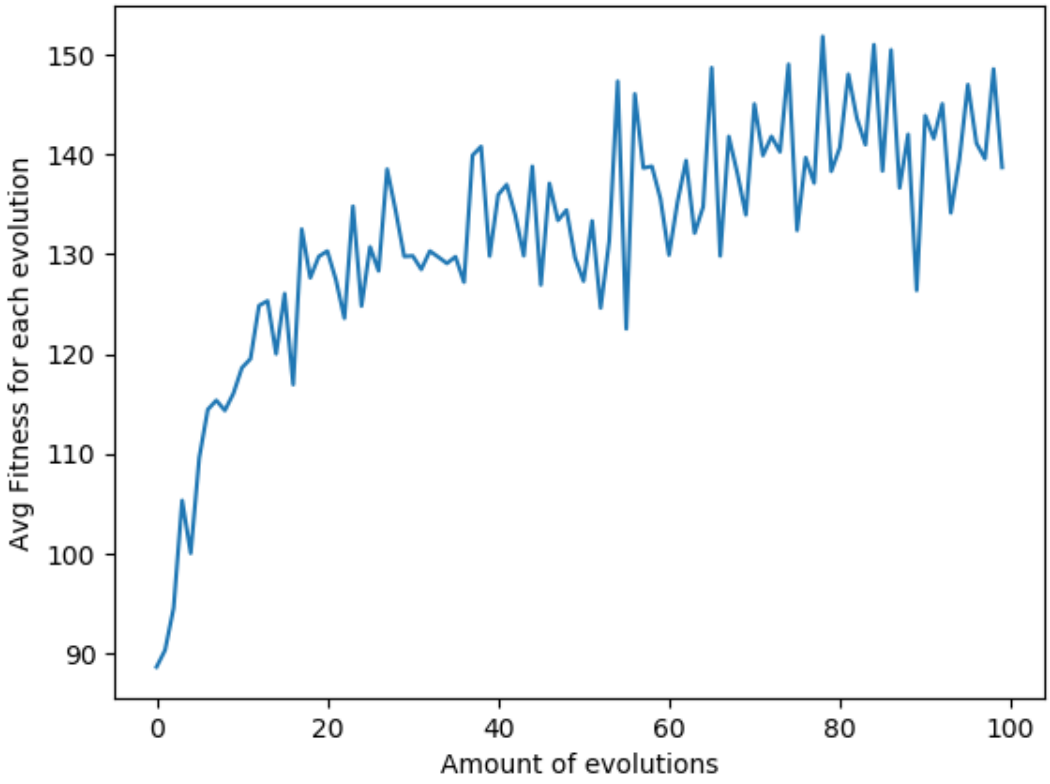


Figure shows grid size of 96 with 100 generations.