AI Genetic Algorithm Report

The World

For this assignment, I implemented a genetic algorithm which optimises the fitness of a species of creatures in a simulated two-dimensional world. The simulation that my creatures evolve in is an important aspect in determining and comparing progress between generations.
World one was my world of choice as it mapped percepts to actions directly, making my agent function simpler to develop. World two would have been used for further development once I had proof of improvement through evolutions in world one.
The size of the grid was decided as a way to minimise time expenditure and to ease comparisons between variations of creatures. A grid size of 29 would create a population of 50 creatures, and a grid size of 41 would lead to a population of 100 creatures. Using these two sizes allowed me to decipher if the grid size was a determining factor in the lifespan of my creatures (which it is not). The grid size also determines the number of monsters and berries generated. The fact that this is in proportion to grid size also means that grid size doesn't have an effect of the fitness of creatures.
The number of turns per simulation is set to 100. This is the initial value and makes it easy to see improvement in early stages of development. Once populations begin to have a significant number of survivors per simulations, the turns could be increased to add more competition between the fitness of creatures.
For a demonstration of my genetic algorithm that will result in the most common expected behaviour (based on my development), use world one with the number of turns set to 100 and the grid size to 29.

Creatures

Chromosome

The chromosome that will govern my creatures actions is based on a simplistic and specific model. It is comprised of an array of length 11; directly correlating to the actions returned from the agent function. Each gene is an integer between 0 and 100 which acts as the likelihood of thw creature to complete the associated action. Genes with the index's 0 - 8 govern how reliably the creature can move, index 9 is the creature's ability to eat food and index 10 is their probability to take a random action.

Agent Function

My agent function works based on a priority list of actions and the creature's chromosome. My simulation requires creatures to consume extra energy in order for them not to die of starvation, thus eating food is the highest priority activity in the agent function.
This action of eating food is make up of multiple actions. The first of these actions occurs if a creature is on red food (edible); they will always eat that food with the probability of their food eating gene (9). If the creature is on green (unripe) food, which will temporarily hinder the creature but still give them some energy, the creature will eat the food (with the probability of their 9th gene) if it is not in danger (there are no monsters around). If a creature is not in a position to eat food, or

they are unable to eat due to a low probability of their 9th gene, they will move on to the movement phase of the agent function.
Movement is necessary as it will move creatures closer to food or move them further away from monsters, thus lengthening their potential lifespan. Movement in the agent function is broken down into three sections; movement when monsters are around, movement towards food and uninhibited movement. Monsters pose the biggest threat to creatures and thus the first condition governing creature movement is when there is a monster around it. The ideal movement can be calculated based on the location of a monster (its percept index) by using a circular avoidance equation. If the creature is on a percept index below 4, adding 5 and modding that number by 9 will result in the creature moving the furthest possible distance from the monster. If the creature is on a percept index above 4, performing the same equation, except adding 4 instead of 5, will give the best direction to move in. The creature will then move in this direction with the probability of the ideal direction's gene (eg. moving into the 6th square is decided by the 6th gene). If the creature is not able to avoid the monster it will default to moving towards (with the appropriate probability) food if there is some near, or moving in a random direction. If there are multiple monsters around the creature, it will opt to move in a random direction.
If the creature reaches the end of the movement options, they will either explore or a random action will be decided. The explore option is chosen based on creatures 10th gene's probability, a random action will be chosen if the probability is too low which could result in the creature moving in a random direction, or potentially performing a useless action (eating when there is no food).
The actions array is returned with a float corresponding to the desired action. When the agent function of a creature has decided on an action, it will set that action index to the maximum value of a float thus ensuring the desired action.

Genetic Algorithm

Fitness

In order to determine which creatures fared better in the simulation than others, creatures are assigned a quantitative fitness value. During simulations, creatures that were able to survive till the end were more favoured and thus had a higher fitness (automatically set to 500). If the creature did die, it's fitness was determined to be its energy. The number of turns a creature lived for could also have been considered in evaluating fitness but this lead to creatures that had starved obtaining a higher fitness that they deserved (due to their inability to eat food).
A creatures fitness is utilised during parent selection in my genetic algorithm. By quantifying the successfulness of a creature during a given simulation, future populations can be generated from the most successful parents.

Selection

The selection algorithm determines which creatures will make up the new population (the evolved population). Through the processes of elitism and dual parent combination, a new population of children is generated.
Elitism is the process of including a certain percentage of top individuals in the new population. This process aids in early evolutionary improvement and in future proofing the next generation. In order to maintain a reasonable amount of diversity in populations, a maximum of 10% of the population can be selected for elitism. The individuals that will be used in elitism practices are

determined by their overall fitness, the top 10% of the population become the first 10% of the next generation. After elitism has been implemented, the rest of the next generation is chosen through a truncation selection algorithm.
Truncation selection selects the creatures with the highest fitness to be parents of the next generation. In my implementation each creature can be used twice, once as parent one and once as parent two before they are removed from the list of potential parents. The fittest parent is first selected, followed by the second fittest parent. A cross over method is then used to generate the child's chromosome and that child is added to the new population. Once the process of generating the next child is complete, the first parent is removed from the list of potential parents, and the process of selection begins again. This is performed until the remaining 90% of the next generation is determined.

Crossover

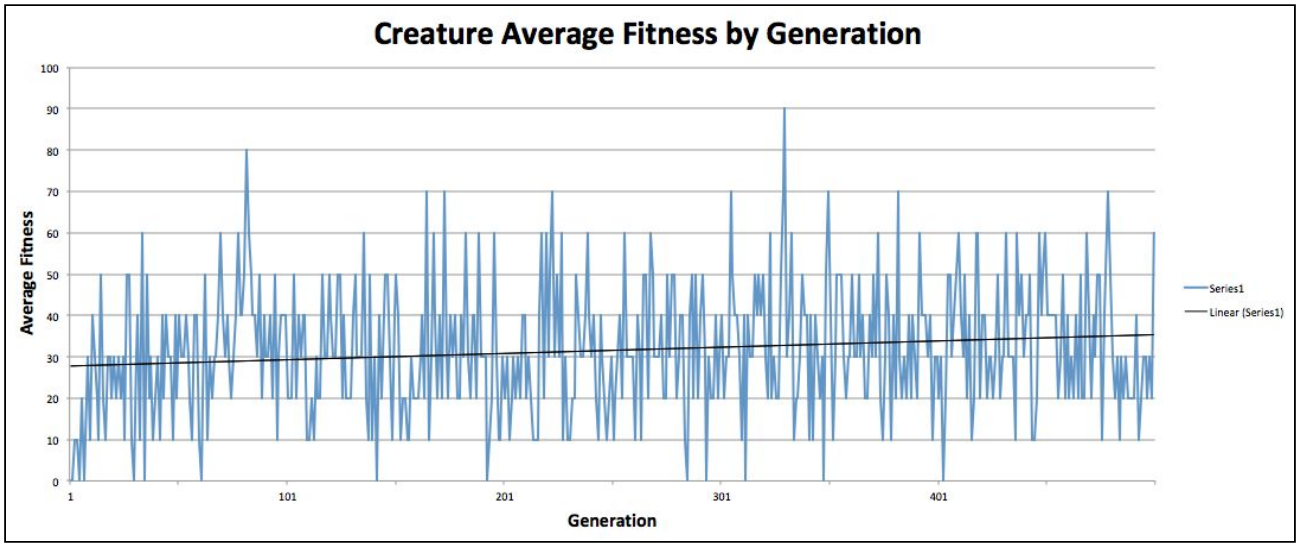
The crossover function takes two parent creatures in as parameters are returns a child who is the combination of the parents chromosomes. A random chromosome is chosen to be the crossover point and the child's chromosome is then generated based on the part of parent one's chromosome and part of parent two's. The first part of the chromosome comes from parent one and the second half from parent two.
The final stage in the crossover function, and child generation, is potential mutation. There is a 0.06% chance that a child will experience mutation in their chromosome.

Mutation

As previously mentioned a child has a 0.06% chance of mutating when their chromosome is being generated. Once a child has been selected for mutation a random gene is selected to be mutated. This mutation consists of inverting the gene probability (eg. 36 will mutate to 64, 88 to 12). Mutation is a necessary process in the maintenance of diversity within population generation, especially the further into the evolution process populations get.

Results

Average Fitness through evolution



The above graph shows how the average fitness of my populations increases over evolutions. This increase is however, very small. The fact that my genetic algorithm doesn't significantly improve across such a long timeline leads me to believe that some part or parts of my algorithm are not working as well as they could be.

Evolution Results Improvement

Clearly there are aspects of my genetic algorithm or creature functions that could be improved. The logic behind my genetic algorithm seems as if it should result in more improvement than it does, thus I believe it is my creature chromosome and agent function that needs improving.
Firstly, my agent function might be too prescriptive and logically coded. I intended to create a priority of actions but it appears as if I have limited the actions considered by the creatures in doing this. I believe that my creatures would have benefited significantly more from learning which action they should take through the use of perceptrons. This would have resulted in good behaviours (eating food and avoiding monsters) being rewarded more through the genetic algorithm and bad decisions punished (through early death). A change in the agent function would also require a change in my chromosome model.
If I was to let my creatures learn which actions to take through the use of a perceptron, my chromosome should lend itself to providing weights and biases for said perceptron. This means my chromosome would need to evolve into a two-dimensional matrix. The values within this matrix should be reminiscent of a weight vector expected by a perceptron along with a bias value. Currently my chromosome has too narrow of a scope to be used in such a way that creatures could learn what they should do in scenarios.

The creature function model is not the only aspect of my simulations that could also use some improvement. In particular, my crossover function could be improved. Currently the crossover point is selected randomly which maintains diversity well, but potentially too well. By developing a crossover function that favoured a selection of one parents genes over the other, it would be possible to pass on more "fit" genes from one parent. If done correctly, a crossover method like this could lead to faster and more consistent improvement in average fitness across evolutions.

Conclusion

The task for this assignment was to develop creature and genetic algorithms that optimise the fitness of a species of creatures in a simulated two-dimensional world. My implementation of creature behaviours and the genetic algorithm does improve the fitness of the creatures and results in an upwards trend in average fitness across generations (however small). With further exploration and development I believe that more improvement could be seen in my creatures faster and more consistently.