Project 3 - Flatland

Aleksander Skraastad

April 13, 2016

1 EA Choices

Most of the parameters for the EA were only slight adjustments from the previous exercise. The main focus of the experimentation phase revolving around adjusting the crossover, mutation and tournament selection parameters. Each of the previous adult and parent selection classes were tested once using the below attributes for mutation, crossover etc. The result landed on using TournamentSelection with GenerationalMixing. The choice of pool sizes were maximized while considering a minimum of 15 generations in reasonable time. This due to the 5 scenario runs being much slower than the single scenario ones.

Param	**Value**
Mutation Rate	0.7
Component mutation rate	0.15
Crossover Rate	0.9
Tournament Selection K	3
Tournament Selection E	0.2
Child pool size	30
Adult pool size	20

Table 1: General EA parameters

To assess the fitness of each run through the FlatLand environment, the following fitness function was used:

$$f = \frac{1+(F_{eaten}/F_{total})}{1+P_{eaten}*c}$$

Which essentially means to assess the fraction of available food items eaten, and penalize it by the amount of poison eaten, multiplied by a poison penalty scaling factor.

The genotype used in this implementation was an integer genotype, which in turn is translated into a real weight vector phenotype that is used when configuring the neural network.

2 Neural Network Implementation

For this project, a very simple network structure sufficed to provide very good results. It was a trial and error on a small scale that provided the quick good results, and upping the neuron count did not provide any significant performance increase. Especially not given the extra computation time that entails.

A network structure of 3 hidden neurons with 6 input neurons connected, one for each sensor, and 3 output neurons, one for each possible direction was chosen. The activation function for the hidden layer was the rectified linear unit (RELU), and the softmax regression function was used on the output layer. No bias neurons or explicit threshold parameters were needed to create a network yielding decent results.

This should be sufficient to solve this problem, since we have separate inputs for each of the poison sensors, along with associated weights, including separate outputs for each direction with associated weights. This yields 18 weights that can be adjusted, more than enough to encode the information necessary for the agent to move in the appropriate direction.

3 EA Performance analysis

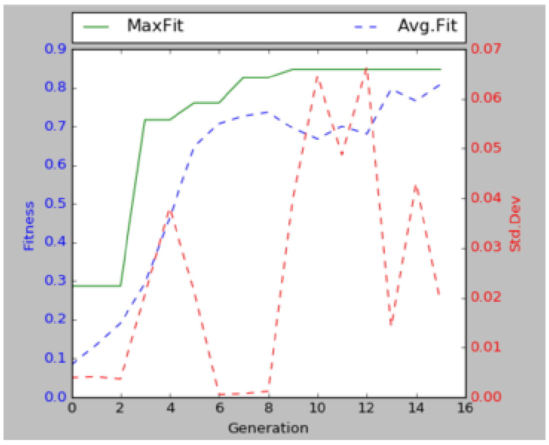

Figure 1: Static - 1

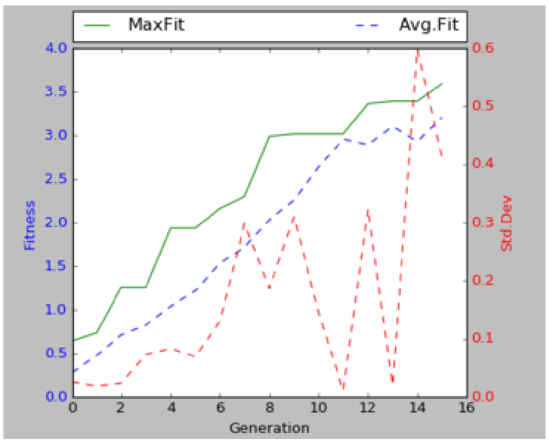

Figure 2: Static - 5

As we can see from the plot, the static 1 scenario run performed quite well, and it managed to consume 38 food without consuming any poison. On the new random generated board, the agent only managed to eat 31 food and 1 poison. In the dynamic environment the agent performed a wrong turn, which it shouldn't. It was not in a dead end that was unavoidable, so it made a bad judgement call. In the static run, the agent behaved very reasonably, given that it cannot see further than 1 cell.

When it comes to the static 5 run, we see from the plot that the agent had a more gradual development, and resulted in a quite high overall fitness. The agent performed very reasonable during most of the scenarios. It did however encounter a quite difficult level in one of the scenarios, yielding a quite low food value of 20, but it did not eat any poison.

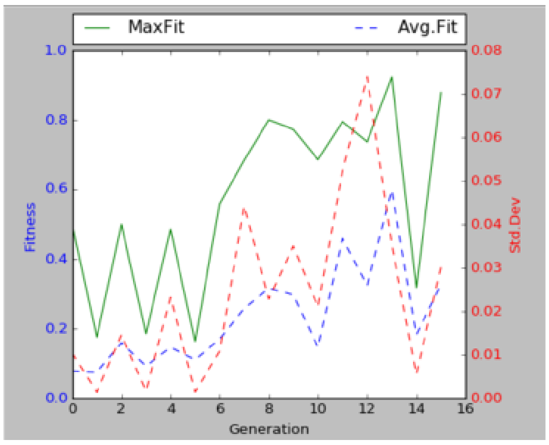

Figure 3: Dynamic - 1

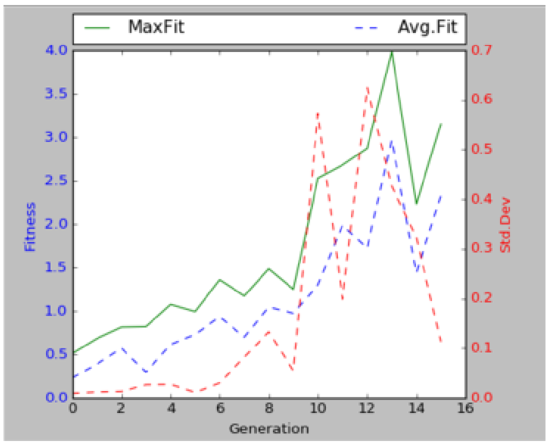

Figure 4: Dynamic - 5

Now we can see something interesting on the plot. The dynamic 1 scenario run oscillates fitness-wise when assessing max-fit, while gradually increasing when assessing the average fitness. There is however a steep drop towards the end of the training, which can be attested to a very difficult level being generated. Possibly a level with poison barriers around much of the food, or pathways of no return. Generating a new random world and running the agent on it produce decent results, with 28 out of 32 food eaten, and no poison ingested. The agent is witnessed as behaving intelligibly throughout the course.

When we look at the plot for the dynamic 5 scenario run, we see the fitness level ascend more smoothly. A very interesting thing here is that generation 13 spawned a 5 scenario set that the agent managed to gain an almost perfect score on, which is really very much based on luck during this run. The fitness drops rather drastically on the next set of levels, before climbing back up again towards the end. As we can see on these dynamic scenarios, we are training the network to play on a generalized level. When we look at the static levels, the fitness never drops. That is because we are constantly getting better, fitting, and possibly over-fitting, on that particular set of levels. When assessing the performance of the agent from the dynamic 5 scenario training run, it performed quite well on 4 out of the 5 levels it played. On the second to last one, however, it managed to clear a complete path from end to end on the level. With the weights being such that it enforced forward direction if no food was present in either direction, the agent simply charges forward, not picking up any more food. This is analogous to when the agent is biased towards turning to one particular side when facing poison either in front or to the opposite side. In a particularly challenging level, an agent may start performing circles, as it continuously attempts to avoid poison by turning to one side.

3.1 General differences and viewpoints

As we can see, the main difference between the static and dynamic runs are the rate of change functions for the plot-lines. The static ones does never go into the negative, while the dynamic runs do. This is, as described earlier, due to the fact that a future randomly generated level might be very difficult, lowering the score of the agents attempting to run its course. Given a set of difficult levels, we can see sharp declines in maximum fitness, as well as average fitness. This is best witnessed (for multiple scenarios) in figure 4, where after achieving an almost perfect fitness score, the drops back to below 2.5, before climbing back upwards. An additional note to read from the plots, are that the relative variations in fitness from generation to generation in the dynamic runs are significantly reduced when running on a set of 5 subsequent scenarios. This effectively reduces the impact of being unusually unlucky on some levels.