Genetic Programming Predator-Prey

Thanushan Pirapakaran
Adrian Binu
COSC 4P82
Prof: Brian Ross
Brock University
March 20th, 2024

Predator-Prey in GP

Given a set of randomly generated moving prey, find the predator that takes the most optimal path in consuming the most prey within the maximum number of moves.

Thus, to find the most optimal simulation, we must also find the best Predator-prey settings. This includes the maximum number of moves and the number of food. Furthermore, we must also find the best grid layout. Otherwise meaning that we need to decide whether the prey should be stopped when it attempts to move into the edge of the grid, or if it should end up on the opposite side of the grid.

Experiment

The parameters illustrated in Table 1, outline the standard parameters used throughout all tests; in terms of population size, crossover rate, mutation rate, etc. Furthermore, Table 2 and Table 3 illustrate the changes in parameters for the two comparisons. In particular, Table 2 compares the difference between a food count of 5 vs. 60.

Parameters	With Walls vs. Without
Pop. Size	100
Crossover Rate	70%
Mutation Rate	30%
Generations	50
# of Elites	1
Min Tree Size (Init)	0
Max Tree Size (Int)	17
Min Tree Size (Mut)	0
Max Tree Size (Mut)	17
Tournament Size	7
Grid Size	20x20
Food Count	20
Max Moves	600

Table 1: With and without walls parameters

	Settings 1	Settings 2
Food Count	5	60

Table 2: Food Count Test - All other settings are the same to the original

Moreover, Table 3 compares the difference

between a max moves count of 1000 moves, 500 moves, and 200 moves.

	Settings 1	Settings 2	Settings 3
Max Moves	1000	500	200

Table 3: # of Max Moves Test - All other settings are the same to the original

Fitness Function

For the Fitness Function, it essentially just returns the number of prey the predator has eaten for just one individual

Language

The Genetic Programming language is illustrated in Figure 12. Essentially, there are two functions and three terminals that move the predator around on the matrix. An example of the language can be seen in Figure 4.

The Simulation

For the simulation it takes an individual and runs the simulation based on the moves the predator took. We would put the predator and prey in the matrix randomly and let the predator and then simulate the moves the predator took and see how many prey it eats. Every tick the predator would either turn left/right or it would move forward and the prey would just move randomly in any direction.

Results

With Walls Vs. Without Walls Test

For this test, the aim was to decide on the best method to handle prey movement. After running each set 10 times, graphing the important results, and looking for patterns within predator and prey positions, the better choice for optimizing fitness is a matrix without walls.

There is a distinct difference in path when looking at predator and prey movements between the two options.

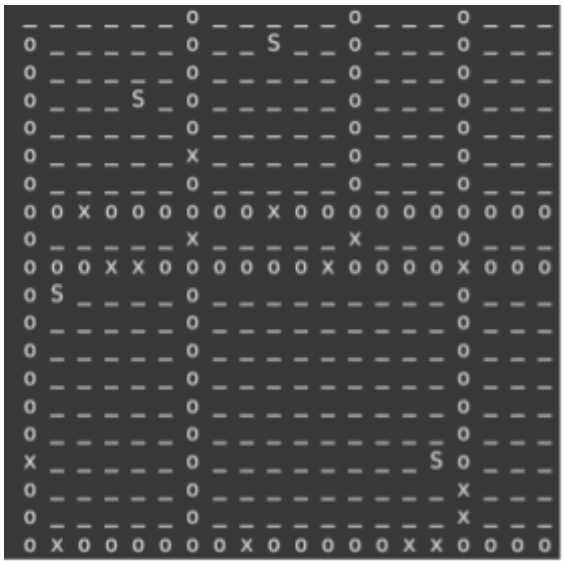

Figure 4: Output of best individual simulation with walls
Individual: prog2(if_food_ahead(prog2(move_forward, turn_left), move_forward), move_forward)

As seen in Figure 4, the prey tends to get stuck near the walls. As a result, the GP would catch on this pattern, and evolve predators that follow walls.

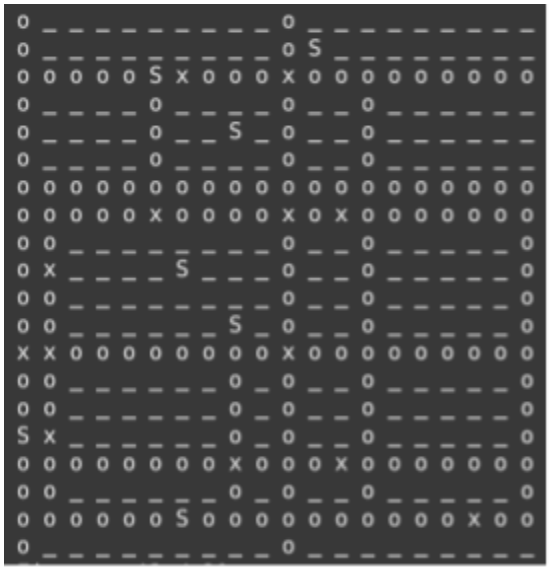

Figure 5: Output of best individual simulation without walls
Individual: if_food_ahead(prog3(move_forward, turn_right, move_forward), move_forward)

Thus, maximizing the number of prey eaten. For this reason, the average fitness with walls illustrated in Figure 1, tends to be higher than the other line.

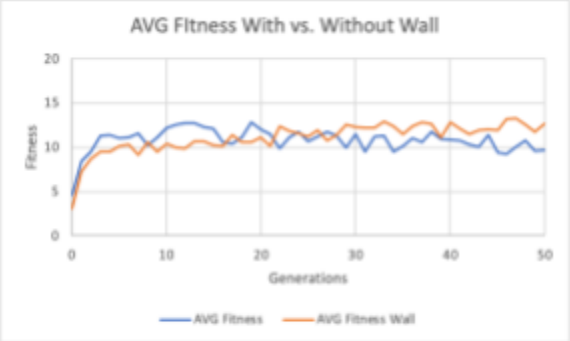

Figure 1: Average Fitness of With vs. Without walls Test

On the other hand, the prey and predator movements in a matrix without walls are much more random (as there is nowhere for the prey to get stuck), as shown in Figure 5. Essentially, the simulation with walls will usually have a better average fitness because most predators evolve to eat the clumped-up prey near the wall. Nonetheless, this leaves the prey that stays near the middle, completely untouched. For this reason, the total search approach of the simulation without walls tends to have better fitness as it covers more area. This includes grid patterns and diagonally moving paths. This can be further seen in Figure 2.

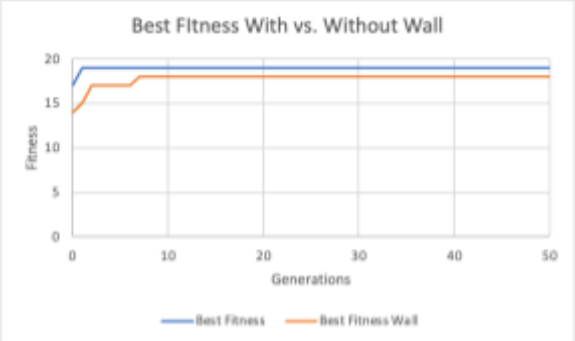

Figure 2: Best Fitness of With vs. Without walls Test

Nevertheless, a matrix with walls is not always the best choice as obstructions within the movements cause more complex solutions. As seen in Figure 3, the simulation without walls tends to have a lower tree size than the simulation with walls. This is as a result of the complexity that walls add to an individual's movement.

Figure 3: Average Tree size of With vs. Without walls Test

Overall, a simulation with walls and without walls has its pros and cons. A simulation with walls has a better average fitness, while the simulation without walls has the highest fitness and less complexity. Thus, a matrix without walls is the best method to handle prey movement.

Max Moves Test

This experiment involved changing the number of moves the predator could take so we tried limiting it to 200, 500 and 1000 moves as seen in Figure 6.

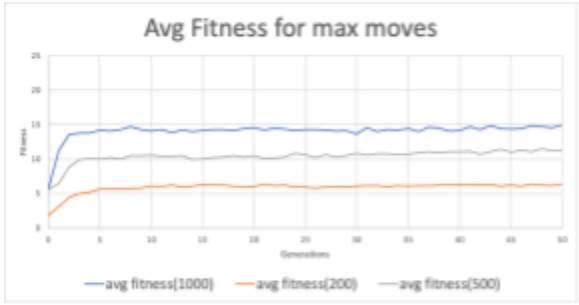

Figure 6: Average Fitness of Max Moves Test

It is clear from looking at this graph that I can conclude when we tried limiting it to lower than 1000 moves we got worse fitnesses on average.

Figure 7: Average Tree Size of Max Moves Test

When we look at Figure 7 the predator limited to 1000 moves also has a lower tree size because it could get away with doing the same pattern many more times and has a more likely chance that prey would just randomly fall into the predator hence why it is more effective and a simpler solution than the rest. So in conclusion we found limiting the predator to 1000 moves was better than limiting it to 500 or 200.

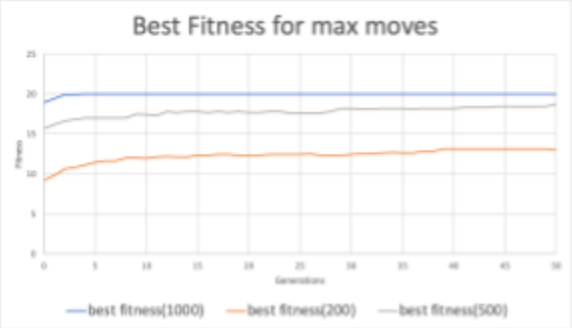

Figure 8: Best Fitness of Max Moves Test

Food Count Test

For this experiment we tested different numbers of foods that the predator could eat. We tried 5 foods and 60 prey, both are extreme numbers we used to reach some kind of conclusion for this test. If you look at Figure 9 we can see that the predator has an easier time eating 5 prey and getting better fitness. The predator has a harder time of getting better fitness with 60 prey because of how many more prey that it missed. You can also see in

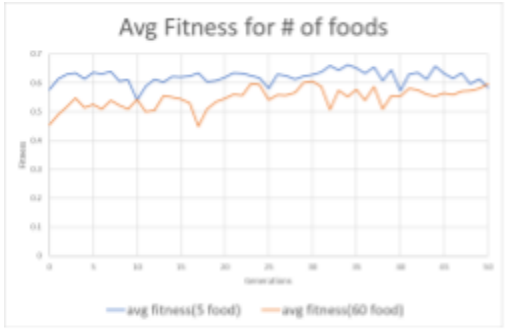

Figure 9: Average Fitness of Number of Foods Test

Figure 11 that it was also harder to find the 5 prey as it was a more complex problem so the tree size got more complex and bigger as a result.

Figure 11: Average Tree Size of Number of Foods Test

Whereas when we set the number of prey to 60 the predator had a much easier time locating the prey and had a much less complex problem at hand so therefore the tree size was not as big compared to the other one. So in conclusion we find that with our 20x20 grid 5 prey was a better parameter to get better fitness, however, if you were to try a bigger grid size then a higher prey count would give better results and therefore I can only conclude that it depends on the problem.

Another thing we wanted to mention was that the predator would follow the same pattern for a lower prey count as you can see in Figure 13 where it follows the diagonal pattern across the grid.

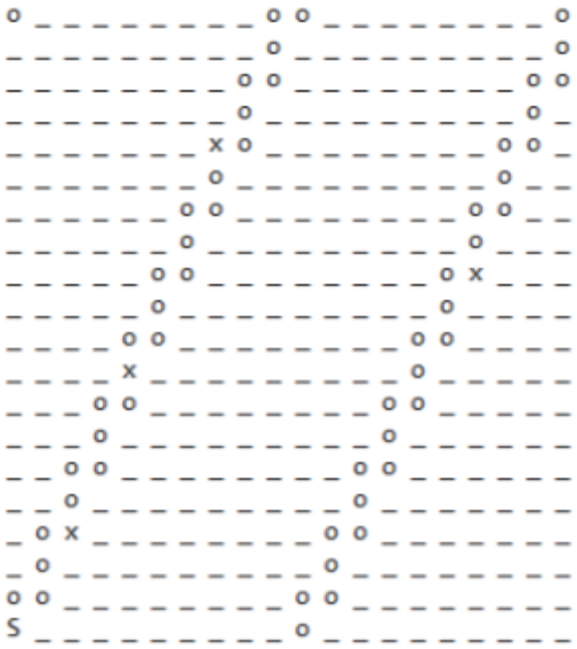

Figure 13: Output of best individual with # of foods set to 5
Individual:fittest:prog3(if_food_ahead(move_forward, turn_right), if_food_ahead(move_forward, move_forward), prog3(turn_left, move_forward, move_forward))

In Figure 14 the predator strategy changed to trying to cover as much of the grid as possible to catch as many prey as possible when the prey count was 60. So in conclusion when the environment was more dense in the population of prey the predator would try to cover as much ground as possible but for a lower dense environment with fewer prey the predator would try to locate prey by following a certain pattern.

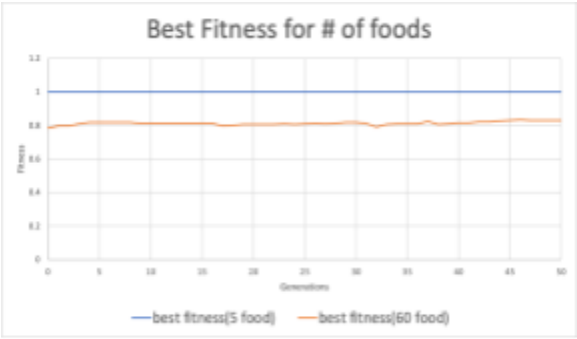

Figure 10: Best Fitness of Number of Foods Test

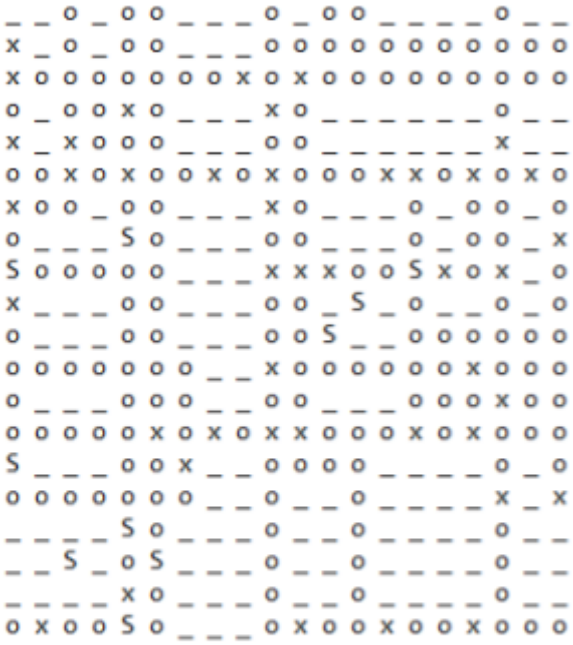

Figure 14: output of best individual with food set to 60
Fittest Individual:
if_food_ahead(prog2(move_forward, turn_left), move_forward)

Conclusion

After all comparisons, our experiments suggest that removing walls from the matrices, along with a maximum move count of 1000 moves and a prey count of 5, consistently leads to the best results. Without walls, predators tend to move more freely, leading to more random and efficient hunting patterns.

We also found that the prey density in the environment affects predator behaviour. Thus, lower prey counts encourage predators to follow specific hunting patterns, while higher prey counts cause them to explore more areas to catch prey.

Additionally, we observed how limiting the number of moves a predator can make impacts its hunting success.

Predators with fewer moves tend to be less effective, while those with more moves can employ a wider range of strategies.

Acronyms

GP Genetic Program

Appendix

Function	Arity	Example
if_food_ahead	2	if_food_ahead A B: execute A If food is in the cell in front of ant, execute A; else execute B
prog2	2	prog2 A B Execute A, then execute B.
prog3	3	prog3 Execute A, then B, then C
Terminal		
move_forward		Move one position forward in the direction its facing
turn_left		Turn 90 degrees counterclockwise
turn_right		Turn 90 degrees clockwise

Figure 12: Predator-Prey Language