EVOLUTIONARY COMPUTATION FINAL

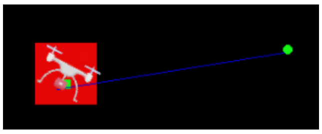
CENG 3512: EVOLUTIONARY COMPUTATION

Görkem Savran
gorkemsavran@posta.mu.edu.tr

Saturday 12^{th} June, 2021

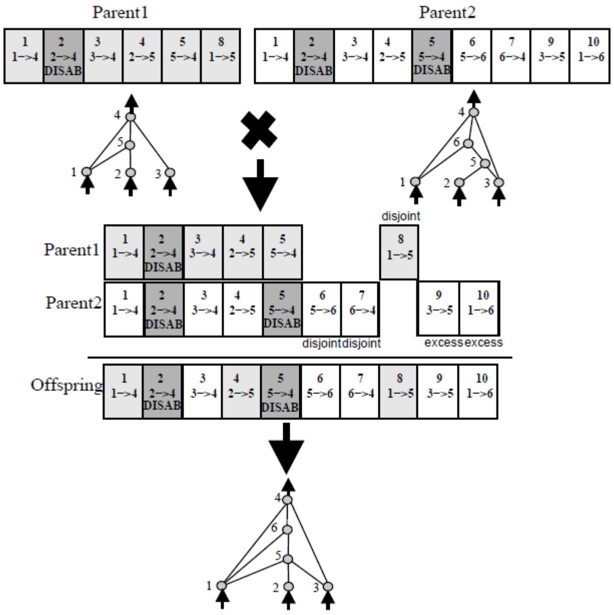
1 Problem Statement

Our problem is to speed up the process and make it cheaper by automating and unmanning difficult and troublesome jobs such as cargo transportation. Writing software that can handle such a problem can be difficult because of elements that people can't think of. Therefore, it would be much better to simulate the problem and enable self-learning agents to develop within this simulation. Here I created a very simple 2D simulation of cargo transportation. In this simulation, the sole purpose of our drone is to deliver the cargo transport box you see with the green square just below our drone to the green point. This simulation can be developed later, and agents that can be used in the real world can be developed by simulating the 3D world environment.



2 Methodology

I used NEAT algorithm in this project. NEAT (NeuroEvolution of Augmenting Topologies) is an evolutionary algorithm that creates artificial neural networks. Then, by looking at the fitness measurement and according to the values in the config file we defined, it performs the evolutionary calculation and creates a new population. You can see an example individual down below.



My fitness function can be defined as follows:
1) If drone reached the target it earns 10 points.
2) If drone is dead it earns -1 point.
3) If drone is closer to target than it's previous position it earns 0.1 point.
4) If drone is further from the target than it's previous position it earns -0.1 point.
5) During the time it can't reach the target, it gains -0.1 points for each frame pass.

3 Results

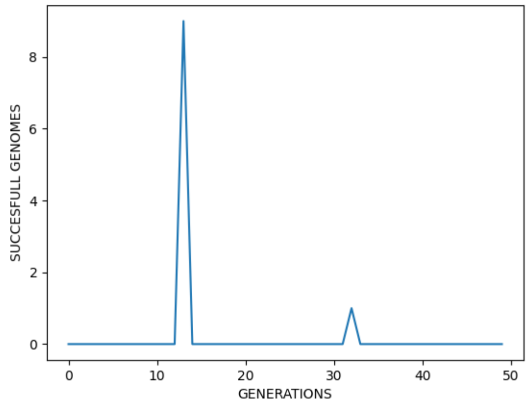
3.1 Wrong Inputs

When I started this experiment, I was giving 4 inputs to the neural network. These inputs are respectively: x position of the center of the cargo box, y position of the center of the cargo box, x position of the center of the target and y position of the center of the target.

When I gave the inputs in this way, the drones could not perform their duties and could not understand what to do. I couldn't understand why the problem was caused because actually all the inputs it needed were here.

This video is the video of my experiment with these four inputs: `https://drive.google.com/file/d/1-4rKRIYbVyfgwZNcsZS831OuIgqW2HlD/view?usp=sharing`

You can see the results down below:



As you can see there is no improvement at all. I changed many things over and over during the experiment. At first I thought it had something to do with the fitness function, but I couldn't make any progress. I even simplified the simulation. Normally there was gravity. I took out Gravity.

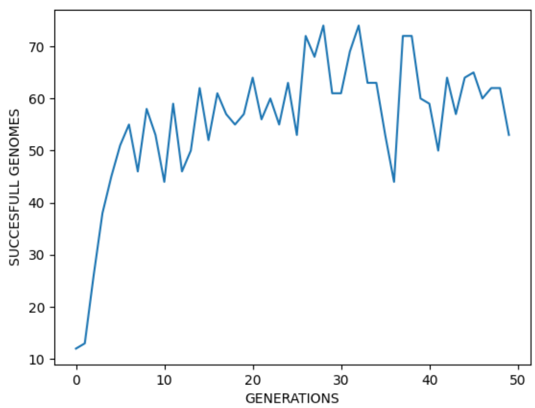
3.2 Good Inputs Without Gravity

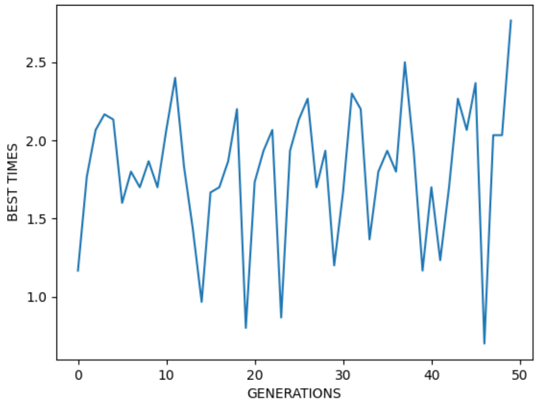
Then I thought of trying it with different inputs. It seems that our neural network could not calculate distance by itself. Therefore, I gave the following 2 inputs in order: x position of the center of our cargo box minus x position of the center of our target and y position of the center of our cargo box minus y position of the center of our target.

It was an incredible development when you gave the inputs like this and the drones started moving towards the green dot.

This is the video of this experiment: `https://drive.google.com/file/d/1FmVb27XVc4WVBsPJMbvfdlnVNYdRuzuc/ view?usp=sharing`

You can see the results down below:



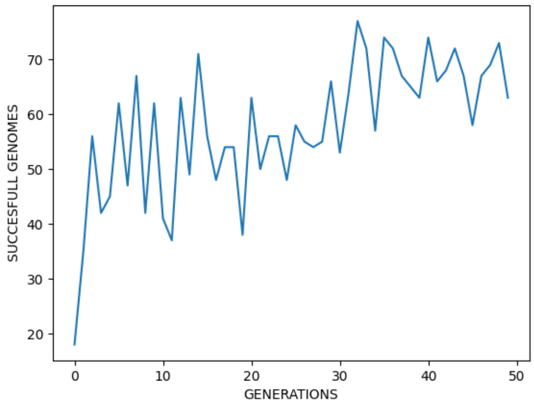


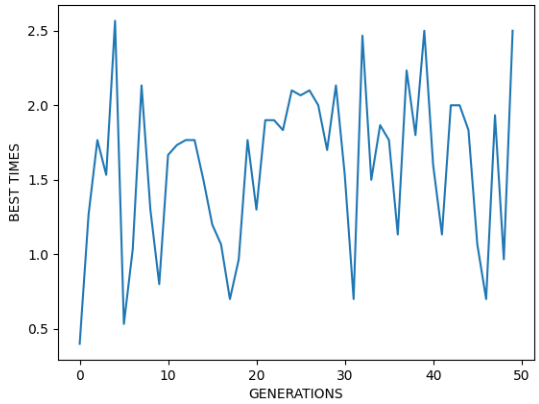
3.3 Good Inputs With Gravity

When I succeeded in this way, I decided to make the game more difficult and added gravity. Likewise, it was successful.

This is the video of this experiment: `https://drive.google.com/file/d/1YLh0yqDj0hWqn_OomRzcmFHFi5ft1LaG/view?usp=sharing`

You can see the results down below:





3.4 Game With Winner Genome

I saved the best performing genome to a file using pickle. Here I changed the game and the target's position became following my mouse's position. So it was a moving target. The winning genome tracked the target very well while in motion.

This is the video of this winner genome: `https://drive.google.com/file/d/1pAbSFwhnSAA2PJ2c-zqGIJi0nJ8k view?usp=sharing`

3.5 Tensorflow

I would also like to mention that when I didn't get good results using NEAT at first, I created the models myself using Tensorflow and subjected the genomes to evolutionary processes myself. But here the performance was so low that I realized that it would not lead to any result and I turned back to the NEAT algorithm. The code I wrote with Tensorflow is in the repository as main_tensorflow.py.

4 Conclusion

As a result, we have seen from these experiments how evolutionary computing can be used in many different places and how effective it can be. With better, more realistic simulations and well-calculated fitness functions, we can solve much more difficult problems. In addition, the use of evolutionary computing together with artificial neural networks may lead to groundbreaking new discoveries.