Application of NEAT algorithm in PC Games

Ankur Rai (10114), Sourya Basu (13710)
Guide: Prof. Harish Karnick
Group No. 9

Abstract

- Neuroevolution (NE) is the artificial evolution of neural networks using genetic algorithms.
- In this project, we explore the possibilities of the Neuroevolution of augmenting topologies (NEAT) algorithm by using it in a PC game.
- The game is about two robots in a 2D plane learning to shoot each other and escape attacks at the same time.
- Experiments were performed for over 50 generations of Neural Evolution, the results of which shows good learning by the robots in the given environment.

Introduction

The main advantage of the NeuroEvolution of Augmenting Topologies (NEAT) is that it minimizes the dimensionality of the search space of con necting weights.

The structure in this algorithm is evolved such that the topologies are minimized and grown incrementally which results in significant gains in learning speed.

It is claimed in [1] that the increased efficiency is due to

- employing a principled method of crossover of different topologies
- protecting structural innovation using speciation.
- incrementally growing from minimal structure.

The performance obtained and the variation in fitness and structure parameters thus formed after evolving is being studied.

Previous Work

NEAT algorithm is not the first algorithm that evolve both neural network topologies and weight. TWEANNs is one such algorithm involving.

They faced several problems such as Competing Conventions Problem which was taken care of by NEAT using the historical information about the genes.

Other problems such as survival of the innovation when optimization takes place over a long number of generation is also taken care of by using the concept of speciation.

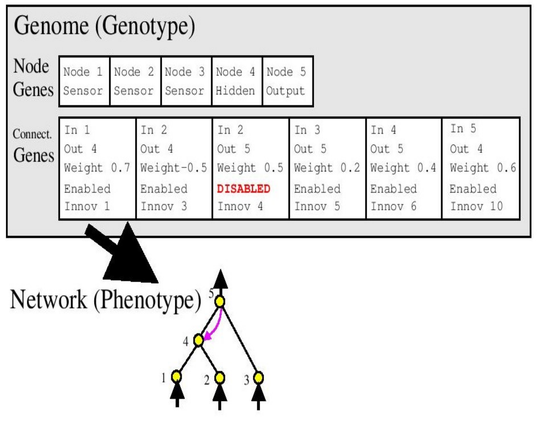
Methodology

overcome the drawbacks in TWEANNs as mentioned earlier, the NEAT algorithm uses several innovative modifications. These are described in detail in the following subsections as mentioned in [2].

Genetic Encoding:

Genomes are linear representations of network connectivity. Each genome includes a list of connection genes, each of which refers to two node genes being connected.

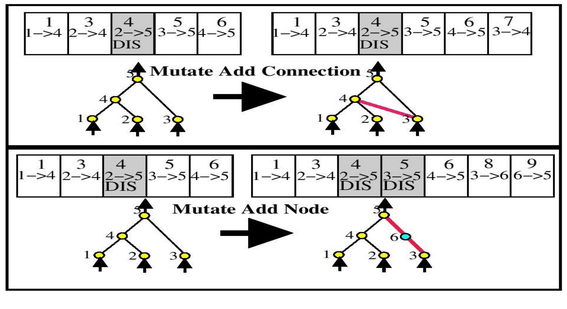
Node genes provide a list of inputs, hidden node, and outputs that can be connected. Each connection gene specifies the in-node, the out-node, the weight of the connection, whether or not the connection gene is expressed (an enable bit), and an innovation number, which allows finding corresponding genes.



The above figure demonstrates two kind of structural mutation that takes place, which are add connection and add node.

- In an add connection mutation, a single new connection gene is added connecting two previously unconnected nodes.
- Whereas in an add node mutation, an existing connection is split and the new node placed where the old connection used to be.

Tracking Genes through Historical Markings:



Origin of each gene is tracked using an innovation number to each gene formed, which represents the chronology of the appearance of the gene in the system.

Solves the problem of competing conventions.

Protecting Innovation through Speciation:

Speciating the population is a solution to one of the problems of the TWEANNs mentioned above.

It allows the organism to compete within their own niches rather than the whole population at large.

In this way, population innovation gets more time to optimize in their own niches and thus innovations are protected using speciation.

Minimizing Dimensionality through Incremental Growth from Minimal Structure:

TWEANNs typically start with an initial population of random topologies in order to introduce diversity from the outset whereas, NEAT biases the search towards minimal dimensional spaces.

This is done by starting out with a uniform population of networks with zero hidden nodes and structural modifications are done incrementally of which only those survive which are found useful through fitness evaluation.

Since the population starts minimally, the dimensionality of the search space is minimized, and NEAT is always search- ing through fewer dimensions than other TWEANNs and fixed-topology NE systems.

Datasets and Algorithm Used

Datasets were self generated.

A 2D region was formed where both the robots were allowed to move around and shoot in any possible way and get feedback from the Fitness function thereafter.

This feedback was used for the evolution of genome for over 50 generation.

The inputs were the X and Y coordinate data of both the robots, and two outputs were taken which were fed to the fitness function.

Code Used

The code was implemented in python for which the library used is neat-python.

Neat-Python examples: https://github.com/CodeReclaimers/neat-python

Results

The results obtained are in the form of videos, each robot performing some basic tasks(the main task was divided into several subtasks which helped in achieving the main goal).

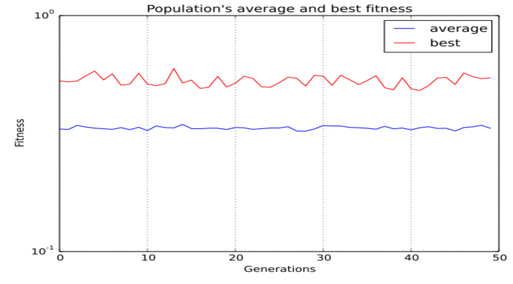
Statistical data involves variation in Fitness value and species size over generations.

Fitness evaluation

The average and the best fitness was evaluated over several generations as shown in the figure.

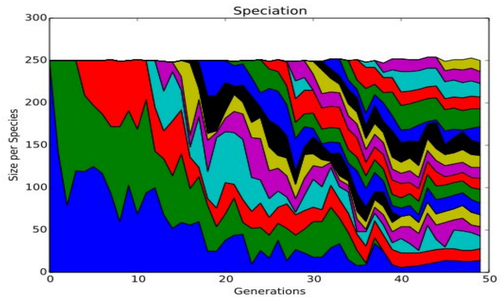
Where the average fitness is found to be almost at, the best fitness is found to have short peaks.

The likely reason being the fact that the minimum of the two fitnesses is taken as the overall fitness.



Speciation

it started with one single species and whenever any mutation resulted in any structure which had a distance greater than some threshold from all the other current species, a new species is formed. More about such structure and threshold is described in [2].



References

1. Stanley, Kenneth O., and Risto Miikkulainen. "Eficient reinforcement learning through evolving neural network topologies." Network (Phenotype) 1.2 (1996): 3.
2. Stanley, Kenneth O., and Risto Miikkulainen. "Evolving neural networks through augmenting topologies." Evolutionary computation 10.2 (2002): 99-127.
3. Stanley, Kenneth O., Bobby D. Bryant, and Risto Miikkulainen. "Evolving neural network agents in the NERO video game." Proceedings of the IEEE (2005): 182-189.