Artificial Intelligence for Games

# A\* Pathfinding Algorithm

The A\* pathfinding algorithm is an algorithm that tries to find the shortest path between two points in a fast way. There are simpler pathfinding algorithms, such as the breadth first search, which can find shortest paths, but takes a long time if the area it must search is large.

A\* works by splitting an area up into a grid and assigning a cost to each node in the grid. The cost is the sum of the distance between cell and the starting and finishing points. The nodes with the lowest costs are used to traverse the map. This way, the algorithm avoids must search through the entire map to find the shortest path, which is why breadth-first-search is slower on larger maps.

Each node also must contain a reference of some kind to their parent node, so that the path can be constructed when the algorithm is finished.

The basic algorithm is this:



# Artificial Neural Networks

Neural networks can be best thought of as simulating a simple brain. A neural network is essentially a group of digital neurons connected in a way that they produce a desired result from an input. Neural networks have at minimum, 3 layers. An input layer, an output layer, and a hidden layer. The input layer is where the input is given to the neurons, and the output layer is where the result ends up. The more complex the neural network, the more layers are between the input and output.

Each neuron in a neural network uses an activation function and a theta to give the output. A neuron takes an input signal (in this case a number) and changes it to produce an output. The way the neuron changes the signal is by using an activation function. There are a few activation functions, step, sign, sigmoid and linear.

Having different amounts of neurons connected in different patterns and ways effects the output of the network, so a neural network needs to be laid out in a way that will accomplish it’s task.

# Genetic Algorithms

Genetic algorithms simulate evolution. A genetic algorithm will start with a random population of chromosomes (usually a string of bits) and have a goal it wants to reach. For each chromosome, their fitness (how close they are to their goal) is calculated somehow. The chromosomes with the higher fitness values will have a higher chance of being picked as a parent for a new chromosome. When two chromosomes are picked, they are then used to produce another chromosome, which will be half of each parent. This is repeated until the new generation is the same size as the starting generation. When a new generation is complete, the old generation is discarded, and the process is repeated until one chromosome with the desired fitness is produced.

Here’s the basic algorithm:



# Comparison

Out of these three methods, A\* is by far the best for pathfinding. It’s faster than the other two methods and is also a lot simpler. A\* was designed specifically to find shortest paths in as little time as possible, and it does this job well. The other algorithms can do pathfinding, but not as quickly, and they may not find the shortest paths, just a path.

However, genetic algorithms and neural networks have one major advantage over A\*, they have more general use. For example, in a genetic algorithm, all that you must change is how the fitness of the chromosomes are calculated, and how long each chromosome is, and then it can be used for something else that could be completely unrelated to pathfinding.

By learning how to create one neural network/genetic algorithm, you can apply them to many different problems, whereas A\* only solves one.

Between neural networks and genetic algorithms, I believe that genetic algorithms are more suited for pathfinding. This is because they don’t have to be trained and can have more definable behaviour. In a game, it helps to not spend lots of time pathfinding as if you have many entities all finding different paths to different places, it can take up a lot of processing power. It is also easier to understand the inner workings of a genetic algorithm, which means that bugs would be easier to track down and fix.

Neural networks could however be used to do more than one task for an entity other than pathfinding. They could be used to help enemies in a game act slightly more “human” and make decisions on their own.