Neuroevolution

Neuralnet: 10 hidden neurons, activation functions, biases

Data representation

Weights evolved by evolutionary algorithm

Fitness function

Algorithms

Crossover

Mutation

Parent selection

Fitness selection

Parameter settings

**Data representation**

The players behaviour is determined by a neural net with no hidden layers. A neural network with a hidden layer was also tested but showed no performance gain and increased training time.

The input of the neural network is a 20 dimensional vector containing environmental data. This data contains the distance between the characters, the direction of each character and the projectile distances. the output of the neural network is a 5 dimensional binary vector containing the player’s actions.

The evolutionary algorithms evolve the weights and biases of the neural network. The weights and biases of the neural net are represented by an array containing 100 weights and 5 biases [[w1,... w100],[b1,…bn]]

**Fitness function**

the fitness is calculated by the following function:

f = 0.9 · (100 − e) + 0.1 · p − log t

where p, e are the current energy level of the player and the

enemy, respectively, and t is the total number of timesteps used

to end the fight.

**Algorithms**

Two EA’s where designed for this experiment. The difference between the two EA’s lies in the mutation mechanisms.

The mechanisms and parameters of the evolutionary algorithms where selected by trial and error and intuition based on the book Introduction to Evolutionary Computing

**Crossover**

Both EA’s used the same uniform crossover mechanism. Every weight and bias of one parent has a 50% change to be selected for the child. Because a single layered neural network was used, no dependencies in the weights was assumed. Therefore the choice for uniform crossover seemed logical.

**Parent selection**

From the population a breeding pool is filled with potential parents.

The Breeding Pool is dependent on the variable alpha, which in our case is 0.95, which means 95 % of the above average individuals (in terms of fitness) are put into the breeding pool and 1 - alpha = 5% of the below average individuals are also put into the breeding pool. After the breeding pool of individuals is constucted, the mating pairs are randomly drawn out of the pool until we have enough offspring such that we have (n\_pop - n\_elite) new offsprings.

This parent selection mechanism has a relatively low selection pression and this prevented premature convergence.

**Survivor selection**

As survivor selection mechanism elitism is used. In our case the top 10% fittest are placed into the next generation. This mechanism is chosen to make sure no optimal solution is lost.

**Mutation**

5 different mutation types where designed.

Mutation 0 - weight gets multiplied by a random number in [0,1]

Mutation 1 - Add a little bit of noise N(0, .2)

Mutation 2 - Weight gets 0

Mutation 3 - Swap Coefficient

Mutation 4 - weight gets 1

In EA1, each mutation m\_i gets p\_i = .25, all mutations are apllied sequentially.

In EA2, one of the mutations is chosen and applied with p=.25.

**Parameters**

**Expiremental setup**

Goal

The goal of the experiment was to compare the performance of the two EA’s and to see if there is a significant difference

Hypothetis

Expected is that the different mutation mechanisms cause a significant difference in performance

Setup

The evoman environment has 6 enemies to play against.

The performance of both EA’s where tested against every enemy separately. Every test was repeated 10 times, to encapsulate the non-deterministic nature of the algorithms.

Parameters to add:

Crossover probability: 100%