Brief Article

Cai Crumley

October 8, 2013

1 Introduction

This project conducted research in the field of neuroevolution (NE), in which evolutionary algorithms (EAs) are applied to generate artificial neural networks (ANNs). In particular, we attempted to improve upon the NE method called Enforced Sub-Populations (ESP) by using behaviour and genotype metrics. We used a simulation of a simple predato-prey scenario as the testbed to compare ESP and its modifications with varying parameters.

An ANN is a processing structure that simulates a neural network. An ANN consists of a set of nodes or neurons with weighted, directed connections between them. ANNs can be classified as either feedforward or recurrant depending on the connections between the nodes. An ANN is classified as feedforward if there exists a way of numbering the nodes such that there are no connections from a node to another node with a lower number. If no such numbering exists then the ANN is classified as recurrant [6]. Thus a feedforward ANN can be modelled by a directed acyclic graph, whereas a recurrant ANN cannot. The architecture of feedforward networks typically consists of layers of nodes in which each node is connected to all nodes in adjacent layers, so that the outputs of a layer of nodes become the inputs to the next layer. The first layer of such an ANN corresponds to external inputs and the last layer corresponds to the outputs of the ANN. The output of a node is a function of the sum of its inputs minus the bias, or threshhold, of the node. Formally, we can express this using the following equation [6]:

$$y_i = f_i \left(\sum_{j=1}^{n} w_{ij} x_j - \theta_i \right) \tag{1}$$

where y_i is the output of node i, n is the number of incoming connections, x_j is the jth input to node i, w_{ij} is the weight of the connection between node i and the node with the output correponding to x_j, and θ_i is the bias or threshhold of node i. f_i is called the transfer function and is typically a nonlinear function such as a Gaussian or sigmoid function [6].

Evolutionary algorithms are population-based stochastic search algorithms that mimic evolution in nature in order to generate a solution to a problem [6][1]. A population of genotypes that represent candidate solutions to the problem is gradually improved by

repeated application of genetic operators, including crossover, mutation and selection. The evolution process is guided by a fitness function which defines the goal of the algorithm and is used to rate the fitness of the genotypes. In each generation genotypes are evaluated using the fitness function, and then genotypes that have a poor fitness are replaced with offspring created by applying the crossover and/or mutation operators to fitter genotypes. This process is repeated until some termination condition is met, such as a maximum number of generations being reached or a desired fitness level being achieved. EAs are well suited to solving high-dimensional optimisation problems that have complex fitness landscapes containing many local optima [6].

EAs can been used in conjunction with ANNs in a few different ways. They can be used to optimise the connection weights between nodes (which is known as training the ANN), and they can also be used to optimise the overall architecture or topology of ANNs [6]. EAs can also be used together with other algorithms as part of the traning process [6]. In this project we used a predefined architecture and evolved only the connection weights.

In this project we used a predator-prey simulation similar to the one used by Yong and Miikkulainen in [7] in which 3 predators attempt to capture a single prey on a toroidal grid-based world. This project attempts to expand on the work done by Gomez and Miikkulainen in [2] and Yong and Miikkulainen in [7]. We modify the evolution process by applying crossover between populations of genotypes of different predators based on their similarity, as measured by behaviour and genotype metrics. The hypothesis of this research is that, by using inter-agent crossover appropriately, we can reduce the number of generations taken until a predetermined level of fitness is achieved.

2 Background

2.1 Neuroevolution

EAs are well suited to searching over large, complex search spaces that have many local optima, and are less likely to get stuck in local optima than gradient-based methods [6]. Since they do not rely on gradient information, they can be used in situations where the search space is non-continuous or non-differentiable, or where gradient information is difficult or expensive to obtain. Such situations would be problematic for gradient-based search methods. Thus EAs allow for greater flexibility in the definition of a fitness function versus an error function for a gradient-based method [6][1].

When applying EAs to find an optimal set of connection weights and/or architecture of an ANN, the decisions of how to encode the ANN in a genotype and what genetic operators to use are very important. In [1], Floreano et al. provide three clasifications of genetic representations: direct, indirect and implicit. Direct and indirect representations are discussed in more detail below. Implicit representations are inspired by gene regulatory networks in biology, and won't be examined in this report.

Direct representations have a one-to-one mapping between the genotype and the parameters of the ANN. For example, if the network architecture is fixed, then the

connection weights could simply be encoded as a binary string with each weight corresponding to a fixed number of bits, or the weights could be encoded as a vector of real numbers. In [3], Igel achieved very good results using direct encoding with Covariance Matrix Adaption Evolution Strategy on pole balancing problems using realtively small recurrant ANNs with between 4 and 16 hidden nodes. Floreano et al. suggest that direct representations may not work as well for large networks as they do for small networks, since the size of the genotype must scale with the size of the network which may impede the evolution process [1]. In [6], Yao makes the same observation with regards to direct encoding of network architecture using a connection matrix: large networks will require a large matrix, which greatly increases the size of the search space. However, he points out that by constraining the network architecture, for example to a layered feedforward architecture, the length of the genotype can be greatly reduced. A similar idea could be applied to training ANNs, but it would require knowledge about the search space which is unlikely to be available, especially for large, complex search spaces associated with large ANNs.

Symbiotic, Adaptive Neuroevolution (SANE), introduced by Moriarty and Mikkulainen in [4] is an example of a direct encoding scheme in which each individual genotype in the population encodes the weights of a node in the hidden layerof an ANN. SANE and its extensions are discussed in detail below. Neuroevolution of Augmenting Topologies (NEAT) is another direct encoding scheme introduced by Stanley and Mikkulainen in [5] in which the architecture of ANNs and their connection weights are simultaneously evolved. NEAT attempts to find the minimal necessary topology of an ANN by incrementally growing individual networks. It uses uses historical markers that indicate when a gene (representing either a node or a connection) is added to a genotype in order to allow for meaningful crossover between networks with different topologies. NEAT also protects innovation in topology by using speciation. This allows the algorithm to explore topological changes even if they initially result in reduced fitness (which is often the case). This also promotes diversity, which helps to avoid premature convergence on local optima, which is a common problem with NE [1].

The competing conventions problem, also known as the permutation problem, is a major problem in NE [6][1][5]. The problem arises from the fact that the same ANN has many different possible representations. For example, if two ANNs differ only in the order of nodes in the hidden layer then they are effectively the same ANN, but they would not have the same genetic representation. This means that two genotypes which appear dissimilar may represent the same solution, and applying crossover to such genotypes would very likely result in offspring with poor fitness. This problem is somewhat addressed in NEAT using the historical markings. By tracking the origins of genes, NEAT is able to identify which genes correspond to the same structure, which allows for meaningful crossover even if the networks are topologically dissimilar.

References

[1] Dario Floreano, Peter Dürr, and Claudio Mattiussi. Neuroevolution: from architectures to learning. *Evolutionary Intelligence*, 1(1):47–62, January 2008.

[2] Faustino Gomez and Risto Miikkulainen. Incremental evolution of complex general behavior. *Adaptive Behavior*, 5(3-4):317–342, 1997.

[3] Christian Igel. Neuroevolution for reinforcement learning using evolution strategies. In *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, volume 4, pages 2588–2595. IEEE, 2003.

[4] David E Moriarty and Risto Mikkulainen. Efficient reinforcement learning through symbiotic evolution. *Machine learning*, 22(1-3):11–32, 1996.

[5] Kenneth O Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, January 2002.

[6] Xin Yao. Evolving artificial neural networks. In *Proceedings of the IEEE*, volume 87, pages 1423–1447. IEEE, 1999.

[7] Chern Han Yong and Risto Miikkulainen. Cooperative coevolution of multi-agent systems. *University of Texas at Austin, Austin, TX*, 2001.