NeuroEvolution of Augmenting Topologies (NEAT)

Author: Louis Miranda-Smedley

Full description of the NEAT method and implementation

1 Introduction

NeuroEvolution of Augmenting Topologies (NEAT) was proposed by Kenneth O.Stanley and Risto Miikkulainen in their paper 'Evolving Neural Networks through Augmenting Topologies' in 2002. It claims that NEAT outperforms the best fixed-topology methods used in reinforcement learning (RL) problems. NEAT is able to gain an advantage from evolving neural network topologies along with weights and the main features of NEAT can be described by

1. Method of crossover of networks with different topologies.
2. Protecting structural innovation through speciating the population.
3. Growing networks where optimisation and complexity occur simultaneously from minimal structure.

2 Background

2.1 NeuroEvolution vs Reinforcement Learning

Neuroevolution (NE) is the evolution of artificial neural networks using genetic algorithms, i.e. algorithms which concern mutation, crossover (breeding networks) and selection. Neuroevolution searches through the behaviour space to find a high performing network. This differs from reinforcement learning algorithms which using statistical techniques which attempt to estimate the utility of particular actions in a particular state in the environment. Despite the different approaches, past studies have found Neuroevolution to be faster and more efficient than reinforcement learning methods such as Q-Learning, at solving the most complex RL problems. Because NE searches for an optimal behaviour rather than a value function, it is more effective than RL algorithms in continuous problems and those of high-dimensional state spaces. It is also possible to use NE in non-Markovian tasks where RL algorithms would fail as they are based on the Markov decision process. Markov decision chains are based on the assumption that the next state is a conditional probability based only on the current state and is independent of states prior to the current state. However, a non-Markovian system might be the weather, where there are external factors such as the time of year and carbon content of the atmosphere also play a role in the next state.

2.2 Traditional NeuroEvolution

In previous approaches of NE, the network topology is chosen before evolution begins, which is typically only includes one hidden layer of nodes which are fully connected to the input and output layer. Evolution searches the space of connection weights for this fixed-topology by allowing high-performing networks to reproduce. Therefore, crossover and mutation is used in traditional fixed-topology networks for optimising connection weights. Connection weights are not the only important attribute of a neural network which contributes to their functionality. The topology of the network, which details the amount of nodes and connections and how they are put together, also affects the performance of the network. Before NEAT was introduced, the question was: 'Can evolving topologies along with weights provide an advantage over evolving weights on a fixed-topology?'.

Before NEAT, the most convincing 'yes' argument was that the evolution of both topology and weights saves the time wasted by humans deciding which topology is best at solving the NE problem. Although many fixed-topology NE problems use a fully connected network with one layer of hidden nodes, deciding how many nodes should be in the hidden layer is a time consuming trial-and-error process. Some have even claimed that in certain problems, structure is not necessary and a fixed-topology NE approach is more efficient. The NEAT paper [1] claimed that if done correctly, evolving structure along with weights can significantly enhance the performance of NE through taking advantage of structure as a way to minimise the search space of connection weights.

2.3 TWEANNs

There have been several attempts over the years to develop a system which evolves both neural network topologies and weights. These methods have many different ideas on how Topology and Weight Evolving Artificial Neural Networks (TWEANNs) should be implemented. The goal of NEAT is to find how a NE method can use the evolution of topology to improve its efficiency.

2.3.1 Encoding

All TWEANNs must answer the question of how best to encode networks using a genetic representation to which there are two main families, direct encoding and indirect encoding. Direct encoding, used in most TWEANNs, specifies in the genome every connection and node which will appear in the phenotype whereas, indirect encoding only specifies rules for constructing a phenotyope.

2.3.2 Binary Encoding

The simplest implementation of direct encoding is Binary encoding which is based on a bit string representation of a connection matrix of a network. However, the size of the connectivity matrix is the square of the number of nodes and so this method is impractical for a large number of nodes. It is also somewhat unnatural to use a linear string of bits to represent graph structures, especially when it comes to the crossover of networks.

2.3.3 Graph Encoding

As mentioned, bit strings aren't the most natural way of representing graph networks, most TWEANNs use encodings that represent these graph structures more explicitly. An example of graph encoding would be a dual representation where the first is a representation of the graph structure and the second is a linear genome containing the incoming connection and outgoing connection of every node. This type of encoding exploits the fact that the first representation is more useful in mutations of the network topology and crossovers and the second type is more useful in mutations of the connections weights.

2.3.4 Asexual Networks

Since crossover of networks of different topologies can frequently lead to complications and lack of functionality, some methods have abandoned crossover altogether, this is called Evolutionary Programming. GeNeralized Acquisition of Recurrent Links (GNARL) is an example of Evolutionary Programming which views crossover as limitation of evolution in general. GNARL uses graph encoding and demonstrates that TWEANNs do not need crossover to work.

2.3.5 Indirect Encoding

A type of indirect encoding is Cellular Encoding (CE) where genomes are programs written in a graph transformation language. The transformations are inspired by cell division as in Biology. CE shows that cell divisions can encode the development of networks from just a single cell, akin to how much of nature's organisms have evolved from single cell organisms. Despite the elogant compactness of indirect encoding, NEAT chooses a direct encoding approach because indirect encodings do not map directly onto their phenotypes, meaning they can bias the search in unpredictable ways.

2.4 Competing Conventions

One of the main obstacles in NE is the Competing Conventions Problem, also known as the Permutations Problem. If you think of an elementary mathematical problem such as addition, we are told that the binary operation of addition of two numbers commute for example, A + B = B + A. Both the left-side of the equation and right-side of the equation compute the exact same function but their ordering is different. This causes an issue in the context of NE, since two networks which are computationally identical may have different ordering of nodes in their hidden layers. This means the genomes representing the same solution do not have the same encoding and crossover is likely to produce damaged offspring. This may be perhaps better understood visually in figure 1.

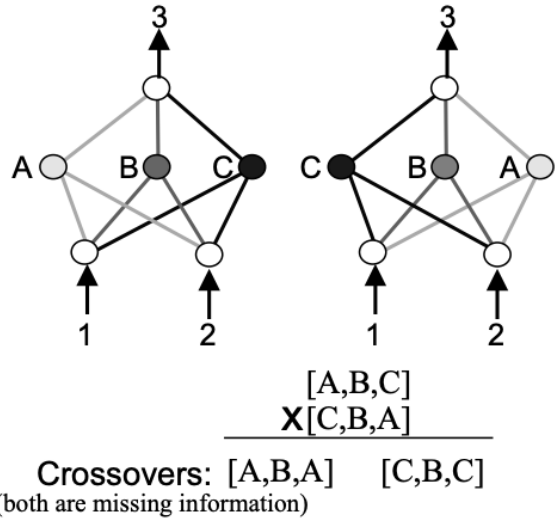

Figure 1: The competing conventions problem [1].

The three hidden nodes A,B and C, can represent the same general solution in 3!=6 different permutations. When one of the permutations is crossed over with another, vital information is most likely lost. We can see this when [A,B,C] is crossed with [C,B,A] leading to either [A,B,A] or [C,B,C] therefore, in each case the offspring loses 1/3 of the information which both parents shared. This can be generalised for n hidden nodes leading to n! functionally equivalent permutations. A far greater and more discrete form of competing conventions exists for TWEANNS, since TWEANN networks can produce similar solutions with entirely different topologies.

The main intuition of NEAT originates from this type of problem, where the representation of different structures may mean they do not match up. Sometimes, genomes have different sizes and others, genes in the exact same position on different chromosomes may express entirely different traits. This means that we cannot simply identify traits from functional units inside networks. By contrast, genes expressing the same trait may appear at different positions on different chromosomes. This issue is complex and seems very difficult to resolve. Nature also faces a similar problem of competing conventions, when it comes to the gene alignment in sexual reproduction. Genomes in nature are not of fixed length either and somewhere along the line of evolving single cells to more complicated organisms, new genes were added in a process known as Gene Amplification. If there were no constraint on where new genes could order themselves in the genome, life may not have evolved at all since the competing conventions problem would lead to a large proportion of offspring being damaged.

Nature found a way of crossing the right genes with the right genes using Homology: two genes are homologous if they are alleles of the same trait. Homology in neural networks can not be recognised from on structural analysis alone. The main insight of NEAT is that the historical origin of two genes is direct evidence of homology if the genes share the same origin. NEAT's solution is to use historical markings, allowing new structure to be added without losing track of which gene is which over the course of evolution.

2.5 Protecting Innovation through Speciation

In TWEANNs innovation takes the form of allowing new structures to be explored through mutation. It is very common that adding new structure will initially decrease the fitness of a network for example, adding a new connection can reduce the fitness before the connection weight is optimised and adding a new node introduces nonlinearity. It is very unlikely that the connection weight the new connection weight is spawned with is useful for the function at the point of it being introduced. Therefore, it is important to protect this new innovation in a way to give the new structure a chance to see if its connection weight can be optimised to lead to greater functionality of the network in the future. Without this protection, the new structure is unlikely to survive in the population as the fitness will most likely be lowered and may a few generation before the fitness of this new structure can be explored properly through connection weight mutation. Therefore, it is necessary to protect this new structure in some way.

GNARL addresses this problem of protecting innovation by deliberately adding nonfunctional structure. A node is initially added to a genome without any connections, hoping for useful connections to develop. However, nonfunctional structures may never end up connecting to the functional network, adding a greater search space to the problem.

Once again, we turn to nature for inspiration. In nature, different structures typically belong to different species which compete in different niches. For instance, birds have evolved to use their ability to fly in order to evade predators and capture prey whereas, dolphins have evolved to become efficient swimmers for the same reason. In this way, it makes no sense to evaluate the fitness of how well a dolphin can swim to the ability of a bird to fly, since these are different niches altogether. Thus, innovation is implicitly protected within a niche. If speciation were to be applied to neural networks, it would allow for different innovative structures to belong to a given species and isolated in their species in this way from the rest of the population, they would have a chance to optimise their structures before having to compete with the population at larger. Speciation, also known as niching, has not been previously used in the context of NE before NEAT. Speciation requires a compatibility function to tell whether two genomes should be in the same species or not, i.e. somewhere along the line of comparing the genomes of a dolphin and bird we should be able to see their dissimilarities emerge from a growing disparity in similar genes. It is difficult to formulate a compatibility function between networks of different topologies, this may have stumped previous attempts of introducing speciation to NE. The competing conventions problem makes the measurement of compatibility even harder in the sense that networks of very different structure may be very similar in the functions they compute.

Since NEAT had a solution to the competing conventions problem using historical markers, the population in NEAT can be speciated by taking advantage of these historical markers. NEAT uses explicit fitness sharing, which forces individuals with similar genomes to share their fitness payoff. This approach leads to protecting innovation through speciation and NEAT allows speciation to be easily integrated using historical markers. This allows for competition to arise inside species and avoids the issue of one species taking over and dominating early on.

2.6 Initial Population

Having an initial population containing random topologies is used in many TWEANN systems, ensuring diversity in structure from the very beginning. However, having random initial populations can have several drawbacks such as, there being a chance that there will be no path connecting the input layer to the output layer. Also, a more serious and subtle problem, it is desirable in TWEANNs to evolve from minimal solutions, this reduces the parameter search space. One way to force minimal topologies to be found is to penalise larger networks by incorporating network size into the fitness function. However, to modify the fitness function in this way to encourage the growth of smaller networks, presents difficulty in deciding how large the penalty should be. In some problems the topology of a good network lends itself to being quite small and in other cases it needs a lot of structure, therefore the penalising parameter must be selected on an ad hoc basis.

An alternative solution is for the neuroevolution process itself to tend towards minimality. If the initial population starts with no hidden nodes and grows structure only as it benefits the the performance, there is no need for later modification to minimize the structure. In this way, minimality is incorporated along the way rather than implemented at the end through brute force. This solution is favoured in NEAT, which uses the approach of minimising structure through starting with a minimal population and growing structure from there. In this way, simple structures turn into more complex structures only if in doing so the new network has a higher fitness.

Through starting minimally, NEAT ensures that the system searches for solutions in the lowest-dimensional weight space over each generation. This is crucial in gaining an advantage in neuroevolution of topology since minimising the search space leads to greater speed and performance gains. One of the reasons TWEANNs do not start out minimally is that if the initial population has no topological diversity, new topological innovation would not survive. The problem of protecting innovation has been solved earlier through speciation. Therefore, speciating the population enables starting minimally in NEAT, this is the crux of what makes NEAT different.

3 NeuroEvolution of Augmenting Topologies (NEAT)

In this section we will pull together some of the ideas from before and showcase the NEAT method. As is a primary step of TWEANNs we start with NEAT's answer to genetic encoding.

3.1 Genetic Encoding

NEAT chooses a genetic encoding method which is designed for ease of use when it comes to lining up two genomes during mating. Genomes are linear representations of network connectivity. Each genome includes a list of connection genes, where information on which two node genes are being connected. Node genes provide a list of of inputs, hidden nodes, and outputs that can be connected in the network. Each connection gene specified the input node and output node, weight of the connection, the enabled status and innovation number. See figure 2 for a visual representation of the NEAT encoding method.

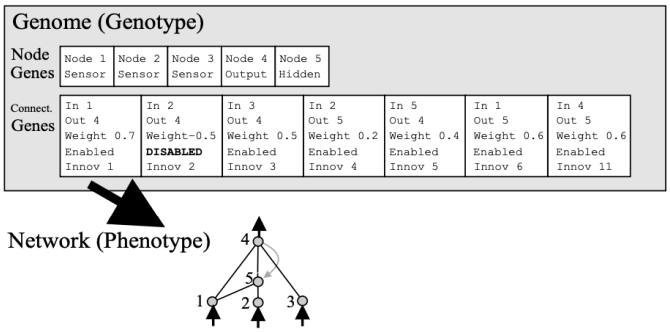

Figure 2: Genotype using dual encoding of node genes and connection genes [1].

Mutation in NEAT comes in two types: connection weight mutation and structural mutation. Connection weights mutate similar to any NE system, with each weight having a probability of being perturbed at each generation. Structural mutations can occur in two ways: adding a connection or adding a node. In the add connection mutation, a single new connection gene with a random weight is added connecting two previously unconnected nodes. In the add node mutation, an existing connection is bisected by a new node. The old connection is disabled (but still present in the genome) and two new connections are added to the genome. The new connection leading into the new node has a weight of 1, and the new connection leading out of the new node receives the same weight as the old connection. These two types structural mutation are demonstrated below in figure 3.

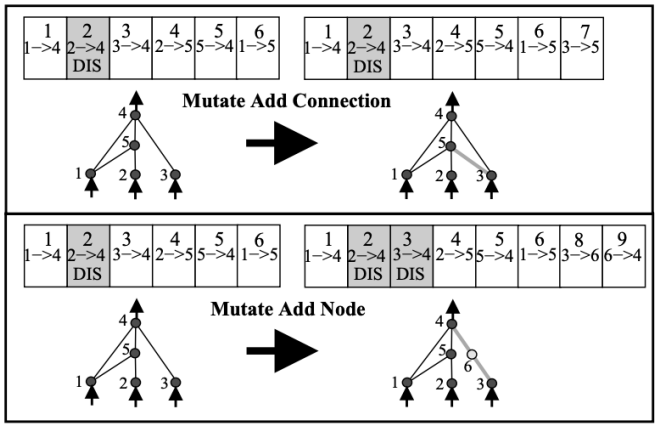

Figure 3: Two types of structural mutation in NEAT [1].

3.2 Gene Tracking through Historical Markings

Evolution has information on gene aligning between any individuals in a population with a diverse range of topologies. This information is the historical origin of the gene. Two genes with the same historical origin must represent the same structure, since they are both derived from the same ancestral gene in the past. Therefore, keeping track of the historical origin of genes in a system allows for ease in gene alignment.

This can be done through historical marking which requires very little computing power. Each time a new gene appears in a genome, a global innovation number is incremented and assigned to that gene. Therefore, the innovation numbers represent the chronology of appearance of every gene in the system. Taking the example shown in figure 3, consider the two mutations to take place one after another. First, the new connection gene is added to the genome and assigned the innovation number 7, and then two more new connection genes result from adding a node, with innovation numbers 8 and 9. With this representation, whenever these genomes mate, their offspring will inherit the same innovation numbers on each gene. This way innovation numbers are never changed and the historical origin of every gene in the system is known throughout evolution.

A problem arises from the chance that the same structural innovation may appear in the same generation through different networks mutating to form identical structures. In this case, the same innovation may be assigned different innovation numbers if we are not careful. By keeping a list of the innovations that occurred in a given generation, when the same structure arises more than once through independent mutations, each identical mutation can be assigned the same innovation number.

This prevents the number of innovation numbers from blowing up.

The use of historical markers has an important implication for NEAT when it comes to the crossover of two parent networks. The system now knows exactly which genes to match up with which, genes with the same innovation numbers are aligned. These genes are called matching genes since they appear in both parents. However, genes that do not match are further categorised as either disjoint or excess, depending on whether they occur within or outside of the range of the other's parent's innovation numbers respectively. In composing the offspring the following steps are followed: genes are randomly chosen from either parent at matching genes, whereas all excess or disjoint genes are passed on from the more fit parent. See figure 4 below for an illustration of how innovation numbers are used in gene alignment and how crossover works.

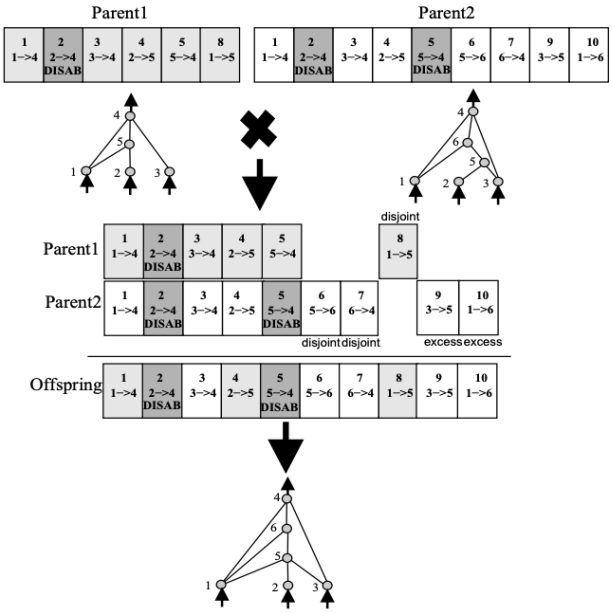

Figure 4: The crossover of Parents 1 and 2 by using innovation number for gene alignment [1].

The process of reproduction is hopeful of producing stronger offspring however, in many cases the offspring is likely to be less fit than the parents. We have discussed the importance of protecting innovation as these networks may go on to be better solutions, NEAT does this through its version of speciation.

3.3 Speciation

Speciating the population allows for individuals to compete inside their own niches instead of with the population at large. This was topological innovations are protected in a new species where they have time to optimise through competition within the niche only. The idea of species in this case is similar to Biology, where similar topologies belong to the same species. Matching topologies can often lead to difficult structural analysis but here the historical markings help out once again.

The number of genes that do not match between two individuals is a good measure of compatibility distance. The more disjoint two genomes are, the less evolutionary history they share. Therefore, the compatibility distance δ between two structures can be found from a linear combination of number of excess E and disjoint D genes, as well as the average weight difference $\overline{W}$ of matching genes, including disabled genes:

$$\delta = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 \cdot \overline{W} \tag{1}$$

The coefficients are able to be adjusted to work efficiently for a given problem, N is the number of genes in the larger genome. This equation for δ gives a measure of 'distance' away from a given species, it is therefore natural to define some border which dictates whether an individual is close enough in structure to be part of the same species or not. This is known as the compatibility threshold δ_t.

A record of the species is maintained where in each generation, genomes are sequentially placed into species. Each species in the current generation are represented by a random genomes inside that species from the previous generation. A given genome g, is placed in the first species with which g is compatible with the representative genome of that species. If g is

found not to be compatible with any species, g will become the representative genome for a new species.

During speciated evolution, NEATs solution to preventing one species from taking over is to use explicit fitness sharing, where organisms only need to compete with other organisms in the same species. This is where organisms share the fitness of their niche within a species by calculating an adjusted fitness f'_i for organism i according to its distance δ from every other organism j in the population.

$$f'_i = \frac{f_i}{\sum_{j=1}^{n} sh(\delta(i,j))} \tag{2}$$

The sharing function sh is set to 0 when the distance between two organisms $\delta(i,j)$ is above the threshold δ_j, otherwise it is set to 1. Thus, $\sum_{j=1}^{n} sh(\delta(i,j))$ becomes the number of organisms in the same species as i. Every species is assigned a potentially different number of offspring in proportion to the sum of adjusted fitnesses f'_i of its member organisms. Species then reproduce by first eliminating the lowest performing members of the population. The entire population is then replaced by the offspring of the remaining organisms in each species.

4 Comments

This document has been written with the intent of digesting the original NEAT paper [1] into hopefully more understandable language and examples. The intention is to be able to understand the NEAT method and to use it for swinging. Since the NEAT paper [1] came out in 2002, this method is relatively speaking quite new and at the time of writing this I found there to be very very limited online help and resources. The two main resources which helped my understanding were of course the original NEAT paper [1] and a 3-part Youtube series on implementing NEAT in Java. I couldn't understand the Java but his explanation along with it was very useful. These two resources will be included in the references.

I really hope other people find this useful and the following rough parameter values as well as the step-by-step implementation guide are of use.

4.1 Parameters

This is intended as a rough guide on parameters to use in your problems using the NEAT method.

Table 1: NEAT Parameter values [1]

Parameter	**Value**
c_1	1.0
c_2	1.0
c_3	0.4
δ_t	0.4
MUTATION RATE	0.5
ADD CONNECTION RATE	0.1
ADD NODE RATE	0.1
POPULATION SIZE	100
ACTIVATION FUNCTION	sigmoid

There are some nuances when it comes to some extra parameters which is included in the 'Parameter Settings' of the original NEAT paper [1]. Such examples include further paramterisation of specific mutations based on species size, additional crossover parameters, adapting mutating weights into two further classes namely, small perturbations or new random values. I encourage you to explore your own options and look online to see how people have used this method to be a bit more creative. The NEAT method is really just a tool and you can make it additionally simpler or more complex at your own leisure. There are some related articles out there which evaluate the effect on different parameter values such as δ_t. Since this value controls the degree of speciation, an online article found that some speciation works really well but too much can be as damaging as no speciation at all [2].

5 Implementing the NEAT method[3]

1. **Place Genomes into species**
   (a) Take initial population and iterate through
   (b) First member i automatically is a mascot of first species
   (c) Take next member $j = i + 1$ and calculate distance from previous member $\delta_{i,j}$ (using equation 1)
   (d) If $\delta_{i,j} < \delta_t$ then they join the same species
   (e) If $\delta_{i,j} > \delta_t$ the member becomes the mascot of a new species
   (f) This occurs recursively until all members of the population have been speciated
2. **Evaluate Genomes and assign fitness**
   (a) Iteratively go through each species and evaluate the absolute fitness of each member f_i
   (b) Adapt the fitness based on how large the species is by the equation for f_i' (using equation 2)
   (c) Order the members in the species starting from highest adjusted fitness to lowest
3. **Put best Genomes from each species into next generation**
   (a) Go through each species and take the highest performing member (first in list)
   (b) Place this member into the new population
   (c) Currently this gives a number of new entries to the new population equal to the number of species
4. **Breed rest of Genomes**
   (a) This leaves a total of $N_{population} - N_{species}$ spaces left to fill in the new population
   (b) We now want to create offspring through breeding members of the current population to fill the remaining spaces
   (c) First, choose the species:
      i. We want to favour sampling our parent networks from species with higher a overall fitness
      ii. So we give each species a biased weight based on their total fitness (sum of adapted fitnesses of members)
      iii. Then sample from this to get a species (higher probability of choosing species with greater fitness)
   (d) Second, choose two parent networks from this chosen species:
      i. Similarly, now looking within a chosen species, each member has an adapted fitness
      ii. Higher adaptive fitnesses lead to a greater weight of being chosen
      iii. Choose two networks from this biased sample (make sure they're not the same)
   (e) Third, crossover the two parent networks:
      i. Align by innovation number
      ii. Matching genes are randomly passed on
      iii. Unmatching genes passed on from more fit parent
   (f) We now have our child network but we still need to do mutations:
      i. There is a chance to randomly mutate connection weights (usually set in a range [-2,2])
      ii. There is a chance of randomly adding a node, creating two new connections and disabling the old one
      iii. The connection leading into the node has weight 1 and the connection leaving node has a random weight
      iv. There is a chance to add a new connection between two previously unconnected nodes
      v. This connection is also assigned a random weight
      vi. Every time a new connection gene is added remember to assign a new innovation number

References

[1] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary computation*, vol. 10, no. 2, pp. 99–127, 2002.

[2] T. Nodine, "Speciation in neat," *https://apps.cs.utexas.edu/tech_reports/reports/tr/TR − 1972.pdf*, 2010.

[3] Hydrozoa, "Neat implementation in java," *https://www.youtube.com/watch?v=1I1eG-WLLrY*.