In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002).*
San Francisco, CA: Morgan Kaufmann **Winner of the Best Paper Award in Genetic Algorithms**

Efficient Reinforcement Learning through Evolving Neural Network Topologies

Kenneth O. Stanley
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712
kstanley@cs.utexas.edu

Risto Miikkulainen
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712
risto@cs.utexas.edu

Abstract

Neuroevolution is currently the strongest method on the pole-balancing benchmark reinforcement learning tasks. Although earlier studies suggested that there was an advantage in evolving the network topology as well as connection weights, the leading neuroevolution systems evolve fixed networks. Whether evolving structure can improve performance is an open question. In this article, we introduce such a system, NeuroEvolution of Augmenting Topologies (NEAT). We show that when structure is evolved (1) with a principled method of crossover, (2) by protecting structural innovation, and (3) through incremental growth from minimal structure, learning is significantly faster and stronger than with the best fixed-topology methods. NEAT also shows that it is possible to evolve populations of increasingly large genomes, achieving highly complex solutions that would otherwise be difficult to optimize.

1 INTRODUCTION

Many tasks in the real world involve learning with sparse reinforcement. Whether navigating a maze of rubble in search of survivors, controlling a bank of elevators, or making a tactical decision in a game, there is frequently no immediate feedback available to evaluate recent decisions. It is difficult to optimize such complex systems by hand; thus, learning with sparse reinforcement is a substantial goal for AI.

Neuroevolution (NE), the artificial evolution of neural networks using genetic algorithms, has shown great promise in reinforcement learning tasks. For example, on the most difficult versions of the pole balancing problem, which is the standard benchmark for reinforcement learning systems, NE methods have recently outperformed other reinforcement learning techniques (Gruau et al. 1996; Moriarty and Miikkulainen 1996).

Most NE systems that have been tested on pole balancing evolve connection weights on networks with a fixed topology (Gomez and Miikkulainen 1999; Moriarty and Miikkulainen 1996; Saravanan and Fogel 1995; Whitley et al. 1993; Wieland 1991). On the other hand, NE systems that evolve both network topologies and connection weights simultaneously have also been proposed (Angeline et al. 1993; Gruau et al. 1996; Yao 1999). A major question in NE is whether such Topology and Weight Evolving Artificial Neural Networks (TWEANNs) can enhance the performance of NE. On one hand, evolving topology along with weights might make the search more difficult. On the other, evolving topologies can save the time of having to find the right number of hidden neurons for a particular problem (Gruau et al. 1996).

In a recent study, a topology-evolving method called Cellular Encoding (CE; Gruau *et al.*, 1996) was compared to a fixed-network method called Enforced Subpopulations (ESP) on the double pole balancing task without velocity inputs (Gomez and Miikkulainen 1999). Since ESP had no a priori knowledge of the correct number of hidden nodes for solving the task, each time it failed, it was restarted with a new random number of hidden nodes. However, even then, ESP was five times faster than CE. In other words, evolving structure did not improve performance in this study.

This article aims to demonstrate the opposite conclusion: if done right, evolving structure along with connection weights can significantly enhance the performance of NE. We present a novel NE method called NeuroEvolution of Augmenting Topologies (NEAT) that is designed to take advantage of structure as a way of minimizing the dimensionality of the search space of connection weights. If structure is evolved such that topologies are minimized and grown incrementally, significant performance gains result.

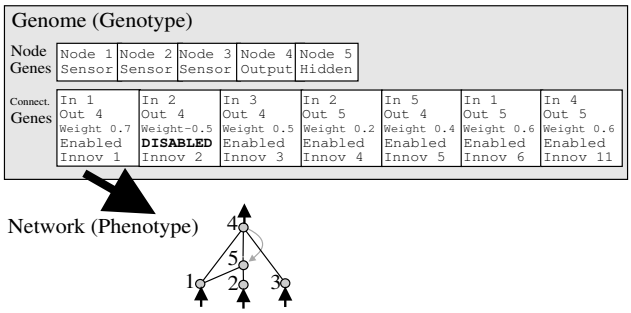

Figure 1: **A Genotype to Phenotype Mapping Example.** A genotype is depicted that produces the shown phenotype. Notice that the second gene is disabled, so the connection that it specifies (between nodes 2 and 4) is not expressed in the phenotype.

Evolving structure incrementally presents several technical challenges: (1) Is there a genetic representation that allows disparate topologies to crossover in a meaningful way? (2) How can topological innovation that needs a few generations to optimize be protected so that it does not disappear from the population prematurely? (3) How can topologies be minimized *throughout evolution* without the need for a specially contrived fitness function that measures complexity?

The NEAT method consists of solutions to each of these problems as will be described below. The method is validated on pole balancing tasks, where NEAT performs 25 times faster than Cellular Encoding and 5 times faster than ESP. The results show that structure is a powerful resource in NE when appropriately utilized.

2 NEUROEVOLUTION OF AUGMENTING TOPOLOGIES (NEAT)

NEAT is designed to address the three problems with TWEANNs raised in the Introduction. We begin by explaining the genetic encoding used in NEAT, and continue by describing the components that specifically address each issue.

2.1 GENETIC ENCODING

NEAT's genetic encoding scheme is designed to allow corresponding genes to be easily lined up when two genomes crossover during mating. Thus, genomes are linear representations of network connectivity (figure 1). Each genome includes a list of *connection genes*, each of which refers to two *node genes* being 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 (as will be explained below).

Mutation in NEAT can change both connection weights and

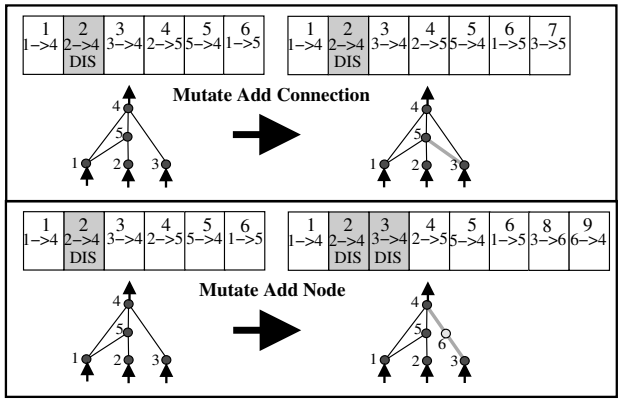

Figure 2: **The two types of structural mutation in NEAT.** Both types, adding a connection and adding a node, are illustrated with the genes above their phenotypes. The top number in each genome is the *innovation number* of that gene. The innovation numbers are historical markers that identify the original historical ancestor of each gene. New genes are assigned new increasingly higher numbers.

network structures. Connection weights mutate as in any NE system, with each connection either perturbed or not at each generation. Structural mutations occur in two ways (figure 2). Each mutation expands the size of the genome by adding gene(s). In the *add connection* mutation, a single new connection gene is added connecting two previously unconnected nodes. In the *add node* mutation an existing connection is split and the new node placed where the old connection used to be. The old connection is disabled and two new connections are added to the genome. This method of adding nodes was chosen in order to integrate new nodes immediately into the network.

Through mutation, the genomes in NEAT will gradually get larger. Genomes of varying sizes will result, sometimes with completely different connections at the same positions. How can NE cross them over in a sensible way? The next section explains how NEAT addresses this problem.

2.2 TRACKING GENES THROUGH HISTORICAL MARKINGS

It turns out that there is unexploited information in evolution that tells us exactly which genes match up with which genes between *any* individuals in a topologically diverse population. That information is the historical origin of each gene in the population. Two genes with the same historical origin must represent the same structure (although possibly with different weights), since they are both derived from the same ancestral gene from some point in the past. Thus, all a system needs to do to know which genes line up with which is to keep track of the historical origin of every gene in the system.

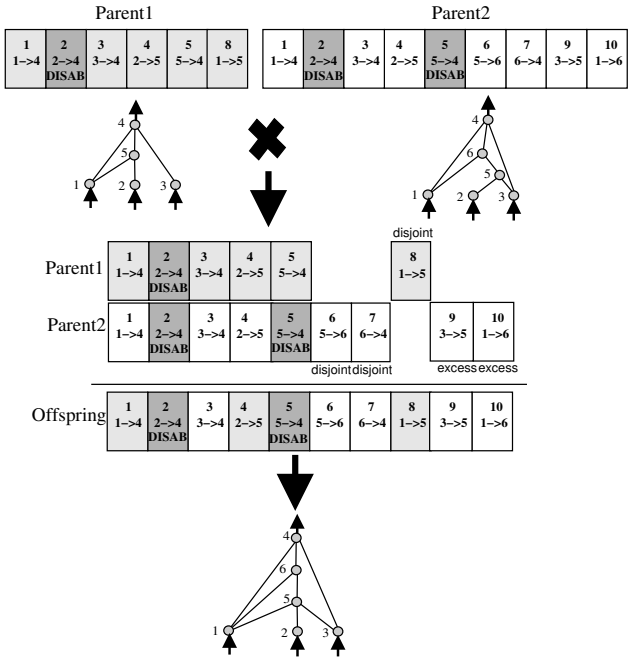

Figure 3: **Matching Up Genomes for Different Network Topologies Using Innovation Numbers.** Although Parent 1 and Parent 2 look different, their innovation numbers (shown at the top of each gene) tell us which genes match up with which. Even without any topological analysis, a new structure that combines the overlapping parts of the two parents as well as their different parts can be created. In this case the parents are equally fit and the genes are inherited from both parents. Otherwise, the offspring inherit only the disjoint and excess genes of the most fit parent.

Tracking the historical origins requires very little computation. Whenever a new gene appears (through structural mutation), a *global innovation number* is incremented and assigned to that gene. The innovation numbers thus represent a chronology of the appearance of every gene in the system. As an example, let us say the two mutations in figure 2 occurred one after another in the system. The new connection gene created in the first mutation is assigned the number 7, and the two new connection genes added during the new node mutation are assigned the numbers 8 and 9. In the future, whenever these genomes mate, the offspring will inherit the same innovation numbers on each gene; innovation numbers are never changed. Thus, the historical origin of every gene in the system is known throughout evolution.

The historical markings give NEAT a powerful new capability, effectively avoiding the problem of competing conventions (Montana and Davis 1989; Radcliffe 1993; Schaffer et al. 1992). The system now knows exactly which genes match up with which (figure 3). When crossing over, the genes in both genomes with the same innovation numbers are lined up. These genes are called *matching* genes. Genes that do not match are either *disjoint* (D) or *excess* (E), depending on whether they occur within or outside the range of the other parent's innovation numbers. They represent structure that is not present in the other genome. In composing the offspring, genes are randomly chosen from either parent at matching genes, whereas all excess or disjoint genes are always included from the more fit parent, or if they are equally fit, from both parents. This way, historical markings allow NEAT to perform crossover using linear genomes without the need for expensive topological analysis.

By adding new genes to the population and sensibly mating genomes representing different structures, the system can form a population of diverse topologies. However, it turns out that such a population on its own cannot maintain topological innovations. Because smaller structures optimize faster than larger structures, and adding nodes and connections usually initially decreases the fitness of the network, recently augmented structures have little hope of surviving more than one generation even though the innovations they represent might be crucial towards solving the task in the long run. The solution is to protect innovation by speciating the population, as explained in the next section.

2.3 PROTECTING INNOVATION THROUGH SPECIATION

Speciation is commonly applied to multimodal function optimization and the coevolution of modular systems, where its main function is to preserve diversity (Mahfoud 1995; Potter and De Jong 1995). We borrow the idea from these fields and bring it to TWEANNs, where it protects innovation. Speciation allows organisms to compete primarily within their own niches instead of with the population at large. This way, topological innovations are protected in a new niche where they have time to optimize their structure through competition within the niche.

The idea is to divide the population into species such that similar topologies are in the same species. This task appears to be a topology matching problem. However, it again turns out that historical markings offer a more efficient solution.

The number of excess and disjoint genes between a pair of genomes is a natural measure of their compatibility. The more disjoint two genomes are, the less evolutionary history they share, and thus the less compatible they are. Therefore, we can measure the compatibility distance δ of different structures in NEAT as a simple linear combination of the number of excess (E) and disjoint (D) genes, as well as the average weight differences of matching genes ($\overline{W}$):

$$\delta = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 \cdot \overline{W}. \quad (1)$$

The coefficients, c_1, c_2, and c_3, allow us to adjust the im-

portance of the three factors, and the factor N, the number of genes in the larger genome, normalizes for genome size (N can be set to 1 if both genomes are small, i.e. consist of fewer than 20 genes).

The distance measure δ allows us to speciate using a compatibility threshold δ_t. Genomes are compared to each species one at a time; if a genomes' distance to a randomly chosen member of the species is less than δ_t, it is placed into this species. Each genome is placed into the first species where this condition is satisfied, so that no genome is in more than one species. Measuring δ for a pair of genomes is linear in the number of connections even though δ precisely expresses compatibility between multi-dimensional topologies. This efficiency is possible because of the historical markings.

As the reproduction mechanism for NEAT, we use *explicit fitness sharing* (Goldberg and Richardson 1987), where organisms in the same species must share the fitness of their niche. Thus, a species cannot afford to become too big even if many of its organisms perform well. Therefore, any one species is unlikely to take over the entire population, which is crucial for speciated evolution to work. The original fitnesses are first adjusted by dividing by the number of individuals in the species. Species then grow or shrink depending on whether their average adjusted fitness is above or below the population average:

$$N'_j = \frac{\sum_{i=1}^{N_j} f_{ij}}{\overline{f}}, \qquad (2)$$

where N_j and N'_j are the old and the new number of individuals in species j, f_{ij} is the adjusted fitness of individual i in species j, and $\overline{f}$ is the mean adjusted fitness in the entire population. The best-performing $r\%$ of each species is randomly mated to generate N'_j offspring, replacing the entire population of the species. [1]

The net effect of speciating the population is that topological innovation is protected. The final goal of the system, then, is to perform the search for a solution as efficiently as possible. This goal is achieved through minimizing the dimensionality of the search space.

2.4 MINIMIZING DIMENSIONALITY THROUGH INCREMENTAL GROWTH FROM MINIMAL STRUCTURE

TWEANNs typically start with an initial population of random topologies (Angeline et al. 1993; Dasgupta and McGregor 1992; Gruau et al. 1996; Zhang and Muhlenbein 1993). This way topological diversity is introduced to the population from the outset. However, it is not clear that such diversity is necessary or useful. A population of random topologies has a great deal of unjustified structure that has not withstood a single fitness evaluation. Therefore, there is no way to know if any of such structure is *necessary*. It is costly though because the more connections a network contains, the higher the number of dimensions that need to be searched to optimize the network. Therefore, with random topologies the algorithm may waste a lot of effort by optimizing unnecessarily complex structures.

In contrast, NEAT biases the search towards minimal-dimensional spaces by starting out with a *uniform* population of networks with zero hidden nodes (i.e. all inputs connect directly to outputs). New structure is introduced incrementally as structural mutations occur, and only those structures survive that are found to be useful through fitness evaluations. In other words, the structural elaborations that occur in NEAT are always justified. Since the population starts minimally, the dimensionality of the search space is minimized, and NEAT is always searching through fewer dimensions than other TWEANNs and fixed-topology NE systems. Minimizing dimensionality gives NEAT a performance advantage compared to other approaches, as will be discussed next.

3 POLE BALANCING EXPERIMENTS

3.1 POLE BALANCING AS A BENCHMARK TASK

There are many reinforcement learning tasks where the techniques employed in NEAT can make a difference. Many of these potential applications, like robot navigation or game playing, are open problems where evaluation is difficult. In this paper, we focus on the pole balancing domain because it has been used as a reinforcement learning benchmark for over 30 years (Anderson 1989; Barto et al. 1983; Gomez and Miikkulainen 1999; Gruau et al. 1996; Michie and Chambers 1968; Moriarty and Miikkulainen 1996; Saravanan and Fogel 1995; Watkins and Dayan 1992; Whitley et al. 1993; Wieland 1991, 1990), which makes it easy to compare to other methods. It is also a good surrogate for real problems, in part because pole balancing in fact *is* a real task, and also because the difficulty can be adjusted.

Earlier comparisons were done with a single pole, but this version of the task has become too easy for modern methods. Therefore, we demonstrate the advantage of evolving structure through double pole balancing experiments. Two poles are connected to a moving cart by a hinge and the neural network must apply force to the cart to keep the poles balanced for as long as possible without going

[1] In rare cases when the fitness of the entire population does not improve for more than 20 generations, only the top two species are allowed to reproduce, refocusing the search into the most promising spaces.

beyond the boundaries of the track. The system state is defined by the cart position (x) and velocity ($\dot{x}$), the first pole's position (θ_1) and angular velocity ($\dot{\theta}_1$), and the second pole's position (θ_2) and angular velocity ($\dot{\theta}_2$). Control is possible because the poles have different lengths and respond differently to control inputs.

Double-pole balancing is sufficiently challenging even for the best current methods. Neuroevolution generally performs better in this task than standard reinforcement learning based on value functions and policy iteration (such as Q-learning and VAPS; Watkins and Dayan 1992, Meauleau et al. 1999, Gomez and Miikkulainen 2002). The question studied in this paper is therefore whether evolving structure can lead to greater NE performance.

3.2 COMPARISONS

Two versions of the double pole balancing task are used: one with velocity inputs included and another without velocity information. The first task is Markovian and allows comparing to many different systems. Taking away velocity information makes the task more difficult because the network must estimate an internal state in lieu of velocity, which requires recurrent connections.

On the double pole balancing with velocity (DPV) problem, NEAT is compared to published results from four other NE systems. The first two represent standard population-based approaches (Saravanan and Fogel 1995; Wieland 1991). Saravanan and Fogel used Evolutionary Programming, which relies entirely on mutation of connection weights, while Wieland used both mating and mutation. The second two systems, SANE (Moriarty and Miikkulainen 1996) and ESP (Gomez and Miikkulainen 1999), evolve populations of neurons and a population of network blueprints that specifies how to build networks from the neurons that are assembled into fixed-topology networks for evaluation. SANE maintains a single population of neurons. ESP improves over SANE by maintaining a separate population for each hidden neuron position in the complete network. To our knowledge, the results of ESP are the best achieved so far in this task.

On the double pole balancing without velocity problem (DPNV), NEAT is compared to the only two systems that have been demonstrated able to solve the task: Cellular Encoding (CE; Gruau *et al.*, 1996), and ESP. The success of CE was first attributed to its ability to evolve structures. However, ESP, a fixed-topology NE system, was able to complete the task five times faster simply by restarting with a random number of hidden nodes whenever it got stuck. Our experiments will attempt to show that evolution of structure can lead to better performance if done right.

3.3 PARAMETER SETTINGS

We set up our pole balancing experiments as described by Wieland (1991) and Gomez (1999). The Runge-Kutta fourth-order method was used to implement the dynamics of the system, with a step size of 0.01s. All state variables were scaled to $[-1.0, 1.0]$ before being fed to the network. Networks output a force every 0.02 seconds between $[-10, 10]N$. The poles were 0.1m and 1.0m long. The initial position of the long pole was 1° and the short pole was upright; the track was 4.8 meters long.

The DPV experiment used a population of 150 NEAT networks while the DPNV experiment used a population of 1,000. The larger population reflects the difficulty of the task. ESP evaluated 200 networks per generation for DPV and 1000 for DPNV, while CE had a population of 16,384 networks. The coefficients for measuring compatibility were $c_1 = 1.0$ and $c_2 = 1.0$ for both experiments. For DPNV, $c_3 = 3.0$ and $\delta_t = 4.0$. For DPV, $c_3 = 0.4$ and $\delta_t = 3.0$. The difference in the c_3 coefficient reflects the size of the populations; a larger population has more room for distinguishing species based on connection weights, whereas the smaller population relies more on topology.

If the maximum fitness of a species did not improve in 15 generations, the networks in that species were not allowed to reproduce. Otherwise, the top 40% (i.e. the elite) of each species reproduced by random mate selection within the elite. In addition, the champion of each species with more than five networks was copied into the next generation unchanged and each elite individual had a 0.1% chance to mate with an elite individual from another species. The offspring inherited matching genes randomly from either parent, and disjoint and excess genes from the better parent, as described in section 2.2. While other crossover schemes are possible, this method was found effective and did not cause excessive bloating of the genomes.

There was an 80% chance that the connection weights of an offspring genome were mutated, in which case each weight had a 90% chance of being uniformly perturbed and a 10% chance of being assigned a new random value. The system tolerates frequent mutations because speciation protects radically different weight configurations in their own species. In the smaller population, the probability of adding a new node was 0.03 and the probability of a new link was 0.05. In the larger population, the probability of adding a new link was 0.3, because a larger population has room for a larger number of species and more topological diversity.

We used a modified sigmoidal transfer function, $\varphi(x) = \frac{1}{1+e^{-4.9x}}$, at all nodes. The steepened sigmoid allows more fine tuning at extreme activations. It is optimized to be close to linear during its steepest ascent between activations -0.5 and 0.5.

Method	Evaluations	Generations	No. Nets
Ev. Programming	**307,200**	150	2048
Conventional NE	**80,000**	800	100
SANE	**12,600**	63	200
ESP	**3,800**	19	200
NEAT	**3,578**	24	150

Table 1: **Double Pole Balancing with Velocity Information.** Evolutionary programming results were obtained by Saravanan (1995). Conventional neuroevolution data was reported by Wieland (1991). SANE and ESP results were reported by Gomez (1999). NEAT results are averaged over 120 experiments. All other results are averages over 50 runs. The standard deviation for the NEAT evaluations is 2704 evaluations. Although standard deviations for other methods were not reported, if we assume similar variances, all differences are statistically significant ($p < 0.001$), except that between NEAT and ESP.

3.4 DOUBLE POLE BALANCING WITH VELOCITIES

The criteria for success on this task was keeping both poles balanced for 100,000 time steps (30 minutes of simulated time). A pole was considered balanced between -36 and 36 degrees from vertical.

Table 1 shows that NEAT takes the fewest evaluations to complete this task, although the difference between NEAT and ESP is not statistically significant. The fixed-topology NE systems evolved networks with 10 hidden nodes, while NEAT's solutions always used between 0 and 4 hidden nodes. Thus, it is clear that NEAT's minimization of dimensionality is working on this problem. The result is important because it shows that NEAT performs as well as ESP while finding more minimal solutions.

3.5 DOUBLE POLE BALANCING WITHOUT VELOCITIES

Gruau *et al.* introduced a special fitness function for this problem to prevent the system from solving the task simply by moving the cart back and forth quickly to keep the poles wiggling in the air. (Such a solution does not require computing the missing velocities.) Because both CE and ESP were evaluated using this special fitness function, NEAT uses it on this task as well. The fitness penalizes oscillations. It is the sum of two fitness component functions, f_1 and f_2, such that $F = 0.1f_1 + 0.9f_2$. The two functions are defined over 1000 time steps:

$$f_1 = t/1000 \tag{3}$$

$$f_2 = \begin{cases} 0 & \text{if } t < 100, \\ \frac{0.75}{\sum_{i=t-100}^{t}(|x^i|+|\dot{x}^i|+|\theta_1^i|+|\dot{\theta}_1^i|)} & \text{otherwise.} \end{cases} \tag{4}$$

where t is the number of time steps the pole remains balanced during the 1000 total time steps. The denominator in (4) represents the sum of offsets from center rest of the cart and the long pole. It is computed by summing the absolute value of the state variables representing the cart and long pole positions and velocities. Thus, by minimizing these offsets (damping oscillations), the system can maximize fitness. Because of this fitness function, swinging the poles wildly is penalized, forcing the system to internally compute the hidden state variables.

Method	Evaluations	Generalization	No. Nets
CE	**840,000**	300	16,384
ESP	**169,466**	289	1,000
NEAT	**33,184**	286	1,000

Table 2: **Double Pole Balancing without Velocity Information (DPNV).** CE is Cellular Encoding of Gruau (1996). ESP is Enforced Subpopulations of Gomez (1999). All results are averages over 20 simulations. The standard deviation for NEAT is 21,790 evaluations. Assuming similar variances for CE and ESP, all differences in number of evaluations are significant ($p < 0.001$). The generalization results are out of 625 cases in each simulation, and are not significantly different.

Under Gruau *et al.*'s criteria for a solution, the champion of each generation is tested on generalization to make sure it is robust. This test takes a lot more time than the fitness test, which is why it is applied only to the champion. In addition to balancing both poles for 100,000 time steps, the winning controller must balance both poles from 625 different initial states, each for 1000 times steps. The number of successes is called the *generalization performance of the solution*. In order to count as a solution, a network needs to generalize to at least 200 of the 625 initial states. Each start state is chosen by giving each state value (i.e. x, $\dot{x}$, θ_1, and $\dot{\theta}_1$) each of the values 0.05, 0.25, 0.5, 0.75, 0.95 scaled to the respective range of the input variable ($5^4 = 625$). At each generation, NEAT performs the generalization test on the champion of the highest-performing species that improved since the last generation.

Table 2 shows that NEAT is the fastest system on this challenging task. NEAT takes 25 times fewer evaluations than Gruau's original benchmark, showing that the way in which structure is evolved has significant impact on performance. NEAT is also 5 times faster than ESP, showing that structure can indeed perform better than evolution of fixed topologies. There was no significant difference in the ability of any of the 3 methods to generalize.

4 DISCUSSION AND FUTURE WORK

4.1 EXPLAINING PERFORMANCE

Why is NEAT so much faster than ESP on the more difficult task when there was not much difference in the easier task? The reason is that in the task without velocities,

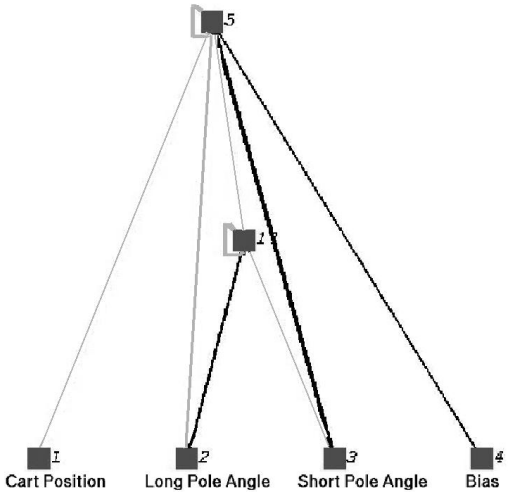

Figure 4: **A NEAT Solution to the DPNV Problem.** Node 2 is the angle of the long pole and node 3 is the angle of the short pole. This clever solution works by taking the derivative of the difference in pole angles. Using the recurrent connection to itself, the single hidden node determines whether the poles are falling away or towards each other. This solution allows controlling the system without computing the velocities of each pole separately. Without evolving structure, it would be difficult to discover such subtle and compact solutions.

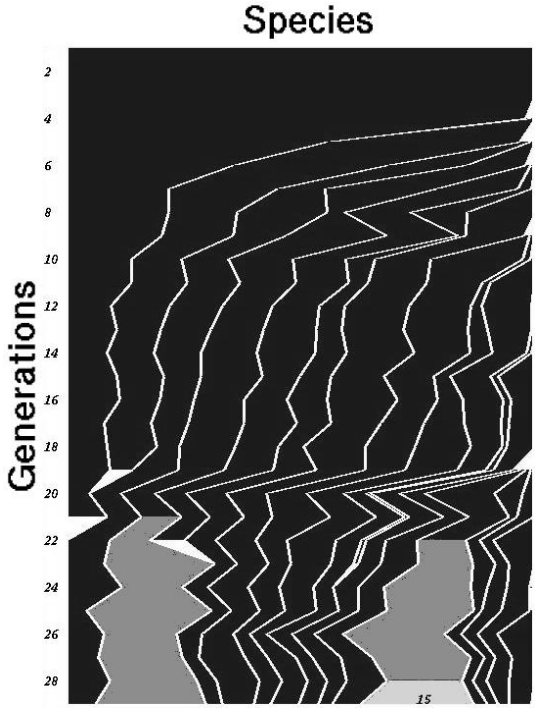

Figure 5: **Visualizing speciation.** The fixed-size population is divided into species, shown horizontally with newer species appearing at right. Time, i.e. evolution generations, are shown vertically. The color coding indicates fitness of the species (lighter colors are better). Two species began to close in on a solution soon after the 20th generation. Around the same time, some of the oldest species became extinct.

ESP needed to restart an average of 4.06 times per solution while NEAT never needed to restart. If restarts are factored out, the systems perform at similar rates. NEAT evolves many different structures simultaneously in different species, each representing a space of different dimensionality. Thus, NEAT is always trying many different ways to solve the problem at once, so it is less likely to get stuck.

Figure 4 shows a sample solution network that NEAT developed for the problem without velocities. The solution clearly illustrates the advantage of incrementally evolving structure. The network is a compact and elegant solution to this problem, in sharp contrast to the fully-connected large networks evolved by the fixed-topology methods. It shows that minimal necessary structures are indeed found, even when it would be difficult to discover them otherwise.

A parallel can be drawn between structure evolution in NEAT and incremental evolution in fixed structures (Gomez and Miikkulainen 1997; Wieland 1991). NE is likely to get stuck on a local optimum when attempting to solve a difficult task directly. However, after solving an easier version of the task first, the population is likely to be in a part of fitness space closer to a solution to a harder task, allowing it to avoid local optima. This way, a difficult task can be solved by evolving networks in incrementally more challenging tasks. Adding structure to a solution is analogous to this process. The network structure before the addition is optimized in a lower-dimensional space. When structure is added, the network is placed into a more complex space where it is already close to a solution. This process is different from incremental evolution in that adding structure is *automatic* in NEAT whereas the sequence of progressively harder tasks must be designed by the experimenter, and can be a challenging problem in itself.

4.2 VISUALIZING SPECIATION

To understand how innovation takes place in NEAT, it is important to understand the dynamics of speciation. How many species form over the course of a run? How often do new species arise? How often do species die? How large do the species get? We answer these questions by depicting speciation visually over time.

Figure 5 depicts a typical run of the double pole balancing with velocities task. In this run, the task took 29 generations to complete, which is slightly above average. In the visualization, successive generations are shown from top to bottom. Species are depicted horizontally for each generation, with the width of each species proportional to its size during the corresponding generation. Species are divided

from each other by white lines, and new species always arrive on the right hand side. Gray-scale shading is used to indicate the fitness of each species. A species is colored dark grey if it has individuals that are more than one standard deviation above the mean fitness for the run, and light grey if they are two standard deviations above. These two tiers identify the most promising species and those that are very close to a solution. Thus, it is possible to follow any species from its inception to the end of the run.

Figure 5 shows that only one species existed in the population until the 5th generation, that is, all organisms were sufficiently compatible to be grouped into a single species. In successive generations, the initial species shrank dramatically in order to make room for the new species, and eventually became extinct in the 21st generation. Extinction is shown by a white triangle between the generation it expired and the next generation. The initial species with minimal structure was unable to compete with newer, more innovative species. The second species to appear in the population met a similar fate in the 19th generation.

In the 21st generation a structural mutation in the fourth species connected the long pole angle sensor to a hidden node that had previously only been connected to the cart position sensor. This innovation allowed the networks to combine these observations, leading to a significant boost in fitness (and brightening of the species in figure 5). This innovative species subsequently expanded, but did not take over the population. Nearly simultaneously, in the 22nd generation, a younger species also made its own useful connection, this time between the short pole velocity sensor and long pole angle sensor, leading to its own subsequent expansion. In the 28th generation, this same species made a pivotal connection between the cart position and its already established method for comparing short pole velocity to long pole angle. This innovation was enough to solve the problem within one generation of additional weight mutations. In the final generation, the winning species was 11 generations old and included 38 neural networks out of the population of 150.

Most of the species that did not come close to a solution survived the run even though they fell significantly behind around the 21st generation. This observation is important, because it visually demonstrates that innovation is indeed being protected. The winning species does not take over the entire population.

4.3 FUTURE WORK

NEAT strengthens the analogy between GAs and natural evolution by not only performing the optimizing function of evolution, but also a *complexifying* function, allowing solutions to become incrementally more complex at the same time as they become more optimal. This is potentially a very powerful extension, and will be further explored in future work.

One potential application of complexification is continual coevolution. In a companion paper (Stanley and Miikkulainen 2002) we demonstrate how NEAT can add new structure to an existing solution, achieving more complex behavior while maintaining previous capabilities. Thus, an arms race of increasingly more sophisticated solutions can take place. Strategies evolved with NEAT not only reached a higher level of sophistication than those evolved with fixed-topologies, but also continued to improve for significantly more generations.

Another direction of future work is to extend NEAT to tasks with a high number of inputs and outputs. For such networks, the minimal initial structure may have to be defined differently than for networks with few inputs and outputs. For example, a fully connected two-layer network with 30 inputs and 30 outputs would require 900 connections. On the other hand, the same network with a five-unit hidden layer would require only 300 connections. Thus, the three-layer network is actually simpler, implying that the minimal starting topology for such domains should include hidden nodes.

Finally, the NEAT method can potentially be extended to solution representations other than neural networks. In any domain where solutions can be represented with different levels of complexity, the search for solutions can begin with a minimal representation that is progressively augmented as evolution proceeds. For example, the NEAT method may be applied to the evolution of hardware (Miller et al. 200a,b), cellular automata (Mitchell et al. 1996), or genetic programs (Koza 1992). NEAT provides a principled methodology for implementing a complexifying search from a minimal starting point in any such structures.

5 CONCLUSION

The main conclusion is that evolving structure and connection weights in the style of NEAT leads to significant performance gains in reinforcement learning. NEAT exploits properties of both structure and history that have not been utilized before. Historical markings, protection of innovation through speciation, and incremental growth from minimal structure result in a system that is capable of evolving solutions of minimal complexity. NEAT is a unique TWEANN method in that its genomes can grow in complexity as necessary, yet no expensive topological analysis is necessary either to crossover or speciate the population. It forms a promising foundation on which to build reinforcement learning systems for complex real world tasks.

Acknowledgments

This research was supported in part by the NSF under grant IIS-0083776 and by the Texas Higher Education Coordinating Board under grant ARP-003658-476-2001. Thanks to Faustino Gomez for providing pole balancing code.

References

Anderson, C. W. (1989). Learning to control an inverted pendulum using neural networks. *IEEE Control Systems Magazine*, 9:31–37.

Angeline, P. J., Saunders, G. M., and Pollack, J. B. (1993). An evolutionary algorithm that constructs recurrent neural networks. *IEEE Transactions on Neural Networks*, 5:54–65.

Barto, A. G., Sutton, R. S., and Anderson, C. W. (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13:834–846.

Dasgupta, D., and McGregor, D. (1992). Designing application-specific neural networks using the structured genetic algorithm. In *Proceedings of the International Conference on Combinations of Genetic Algorithms and Neural Networks*, 87–96.

Goldberg, D. E., and Richardson, J. (1987). Genetic algorithms with sharing for multimodal function optimization. In Grefenstette, J. J., editor, *Proceedings of the Second International Conference on Genetic Algorithms*, 148–154. San Francisco, CA: Morgan Kaufmann.

Gomez, F., and Miikkulainen, R. (1997). Incremental evolution of complex general behavior. *Adaptive Behavior*, 5:317–342.

Gomez, F., and Miikkulainen, R. (1999). Solving non-Markovian control tasks with neuroevolution. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*. Denver, CO: Morgan Kaufmann.

Gomez, F., and Miikkulainen, R. (2001). Learning robust nonlinear control with neuroevolution. Technical Report AI01-292, Department of Computer Sciences, The University of Texas at Austin.

Gruau, F., Whitley, D., and Pyeatt, L. (1996). A comparison between cellular encoding and direct encoding for genetic neural networks. In Koza, J. R., Goldberg, D. E., Fogel, D. B., and Riolo, R. L., editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, 81–89. Cambridge, MA: MIT Press.

Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: MIT Press.

Mahfoud, S. W. (1995). *Niching Methods for Genetic Algorithms*. PhD thesis, University of Illinois at Urbana-Champaign, Urbana, IL.

Meuleau, N., Peshkin, L., Kim, K.-E., and Kaelbling, L. P. (1999). Learning finite-state controllers for partially observable environments. In *Proceedings of the Fifteenth International Conference on Uncertainty in Artificial Intelligence*.

Michie, D., and Chambers, R. A. (1968). BOXES: An experiment in adaptive control. In Dale, E., and Michie, D., editors, *Machine Intelligence*. Edinburgh, UK: Oliver and Boyd.

Miller, J. F., Job, D., and Vassilev, V. K. (200a). Principles in the evolutionary design of digital circuits – Part I. *Journal of Genetic Programming and Evolvable Machines*, 1(1):8–35.

Miller, J. F., Job, D., and Vassilev, V. K. (200b). Principles in the evolutionary design of digital circuits – Part II. *Journal of Genetic Programming and Evolvable Machines*, 3(2):259–288.

Mitchell, M., Crutchfield, J. P., and Das, R. (1996). Evolving cellular automata with genetic algorithms: A review of recent work. In *Proceedings of the First International Conference on Evolutionary Computation and Its Applications (EvCA'96)*. Russian Academy of Sciences.

Montana, D. J., and Davis, L. (1989). Training feedforward neural networks using genetic algorithms. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, 762–767. San Francisco, CA: Morgan Kaufmann.

Moriarty, D. E., and Miikkulainen, R. (1996). Efficient reinforcement learning through symbiotic evolution. *Machine Learning*, 22:11–32.

Potter, M. A., and De Jong, K. A. (1995). Evolving neural networks with collaborative species. In *Proceedings of the 1995 Summer Computer Simulation Conference*.

Radcliffe, N. J. (1993). Genetic set recombination and its application to neural network topology optimization. *Neural computing and applications*, 1(1):67–90.

Saravanan, N., and Fogel, D. B. (1995). Evolving neural control systems. *IEEE Expert*, 23–27.

Schaffer, J. D., Whitley, D., and Eshelman, L. J. (1992). Combinations of genetic algorithms and neural networks: A survey of the state of the art. In Whitley, D., and Schaffer, J., editors, *Proceedings of the International Workshop on Combinations of Genetic Algorithms and Neural Networks (COGANN-92)*, 1–37. IEEE Computer Society Press.

Stanley, K. O., and Miikkulainen, R. (2002). Continual coevolution through complexification. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002)*. San Francisco, CA: Morgan Kaufmann.

Watkins, C. J. C. H., and Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3):279–292.

Whitley, D., Dominic, S., Das, R., and Anderson, C. W. (1993). Genetic reinforcement learning for neurocontrol problems. *Machine Learning*, 13:259–284.

Wieland, A. (1991). Evolving neural network controllers for unstable systems. In *Proceedings of the International Joint Conference on Neural Networks* (Seattle, WA), 667–673. Piscataway, NJ: IEEE.

Wieland, A. P. (1990). Evolving controls for unstable systems. In Touretzky, D. S., Elman, J. L., Sejnowski, T. J., and Hinton, G. E., editors, *Connectionist Models: Proceedings of the 1990 Summer School*, 91–102. San Francisco, CA: Morgan Kaufmann.

Yao, X. (1999). Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447.

Zhang, B.-T., and Muhlenbein, H. (1993). Evolving optimal neural networks using genetic algorithms with Occam's razor. *Complex Systems*, 7:199–220.