Efficient Evolution of Neural Network Topologies

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

Abstract

Neuroevolution, i.e. evolving artificial neural networks with genetic algorithms, has been highly effective in reinforcement learning tasks, particularly those with hidden state information. An important question in neuroevolution is how to gain an advantage from evolving neural network topologies along with weights. We present a method, NeuroEvolution of Augmenting Topologies (NEAT) that outperforms the best fixed-topology methods on a challenging benchmark reinforcement learning task. We claim that the increased efficiency is due to (1) employing a principled method of crossover of different topologies, (2) protecting structural innovation using speciation, and (3) incrementally growing from minimal structure. We test this claim through a series of ablation studies that demonstrate that each component is necessary to the system as a whole and to each other. What results is significantly faster learning. NEAT is also an important contribution to GAs because it shows how it is possible for evolution to both optimize *and complexify* solutions simultaneously, making it possible to evolve increasingly complex solutions over time, thereby strengthening the analogy with biological evolution.

I. INTRODUCTION

Neuroevolution (NE), the artificial evolution of neural networks using genetic algorithms, has shown great promise in reinforcement learning tasks. NE outperforms standard reinforcement learning methods in many benchmark tasks [6, 10, 11]. Neural networks are a good class of decision making systems to evolve because they are capable of representing solutions to many different kinds of problems, and the mapping from genotype to phenotype is generally efficient. NE is particularly well suited to reinforcement learning tasks because NE does not require supervision.

A major question in NE is how to gain an advantage from evolving topology in addition to connection weights. On one hand, evolving topology might overcomplicate the search. On the other, it can also save time by finding the right number of hidden neurons for a particular problem automatically [7].

A previous study showed that fixed-topology NE can outperform a topology-evolving system on the benchmark double pole balancing task [6]. This finding is important because pole balancing has been a benchmark task in NE and reinforcement learning for over 30 years [1, 6, 7, 9], and double pole balancing is challenging to even the best of modern methods. Doing well at this important benchmark suggests that a method will do well in other tasks as well. Whether Topology and Weight Evolving Artificial Neural Networks (TWEANNs) can enhance the performance of NE remains an open question.

In this article, we aim to show that evolving topology can indeed increase performance. We present a new TWEANN, NeuroEvolution of Augmenting Topologies (NEAT), that significantly outperforms the fixed-topology NE method that currently takes the fewest evaluations on the double pole balancing task. We identify three major challenges for TWEANNs and present solutions to each of them: (1) Is there a genetic representation that allows disparate topologies to crossover in a meaningful way? Our solution is to use historical markings to line up genes with the same origin. (2) How can topological innovation that needs a few generations to optimize be protected so that it does not disappear from the population prematurely? Our solution is to separate each innovation into a different species. (3) How can topologies be minimized *throughout evolution* without the need for a specially contrived fitness function that measures complexity? Our solution is to start from a minimal structure and grow only when necessary. This paper establishes that each of our solutions is necessary by showing that NE performance significantly declines with the ablation of any of the major solution components. Working together in NEAT these components constitute a promising new approach to difficult reinforcement learning tasks.

We begin by describing the NEAT method, including results showing that NEAT is significantly faster than other NE methods on the hardest pole balancing benchmark. We then present ablation studies designed to explain NEAT's performance in terms of its components.

II. NEUROEVOLUTION OF AUGMENTING TOPOLOGIES (NEAT)

A. *Genetic Encoding*

NEAT is designed specifically to address the three challenges raised in the introduction. Each genome includes a list of *connection genes*, each of which refers to two *node genes* being connected (figure 1). Each connection gene specifies the in-node, the out-node, the weight of the connection, whether

0-7803-7282-4/02/$10.00 ©2002 IEEE

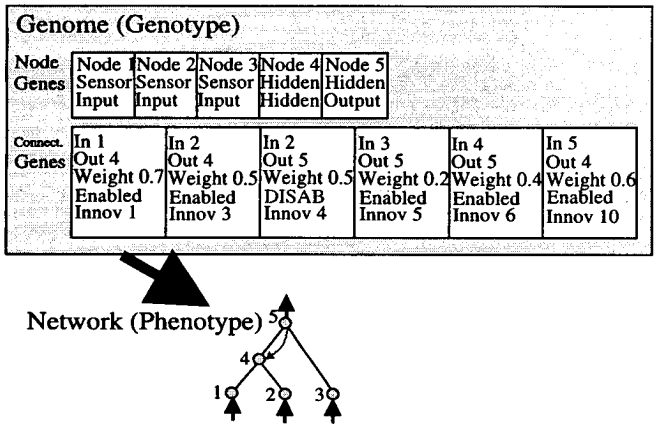

Fig. 1. **A genotype to phenotype mapping example.** The third gene is disabled, so the connection that it specifies (between nodes 2 and 5) is not expressed in the phenotype.

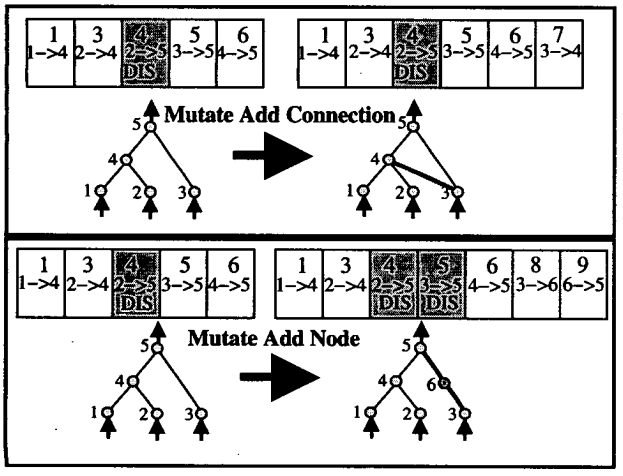

Fig. 2. **The two types of structural mutation in NEAT.** Both types, adding a connection and adding a node, are illustrated with the genes above their phenotypes. The top number in each genome is the *innovation number* of that gene. These numbers identify the original historical ancestor of each gene, making it possible to find matching genes during crossover. New genes are assigned new increasingly higher numbers.

or not the connection gene is expressed (an enable bit), and an *innovation number*, which allows finding corresponding genes during crossover (as will be explained below). Although the experiments in this paper evolve networks with a single output, NEAT can evolve networks with any number of inputs or outputs.

Mutation in NEAT can change both connection weights and network structures. Connection weights mutate as in any NE system, with each connection either perturbed or not. Structural mutations, which expand the genome, occur in two ways (figure 2). 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, genomes of varying sizes are created, sometimes with completely different connections specified at the same positions. The next section explains how NEAT can cross over such diverse genomes.

B. *Tracking Genes through Historical Markings*

In order to perform crossover, the system must be able to tell which genes match up between *any* individuals in the population. The key observation is that two genes that have the same historical origin represent the same structure (although possibly with different weights), since they were 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.

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 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 crossover, 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 solving the problem of competing conventions for disparate topologies (the "Holy Grail" in neuroevolution [14]). The system now knows exactly which genes match up with which (figure 3). Genes that do not match are either *disjoint* or *excess*, depending on whether they occur within or outside the range of the other parent's innovation numbers. When crossing over, the genes in both genomes with the same innovation numbers are lined up. Genes that do not match are inherited from the more fit parent, or if they are equally fit, from both parents randomly. This way, historical markings allow NEAT to perform crossover without the need for expensive topological analysis.

The method of crossover presented here is notable for its simplicity. Any two structures can be combined in a principled manner *without* the need for any topological analysis, even though the problem appears to be a topology combination problem. By recasting the problem as a problem of historical matching, it becomes tractable and significantly simpler to solve.

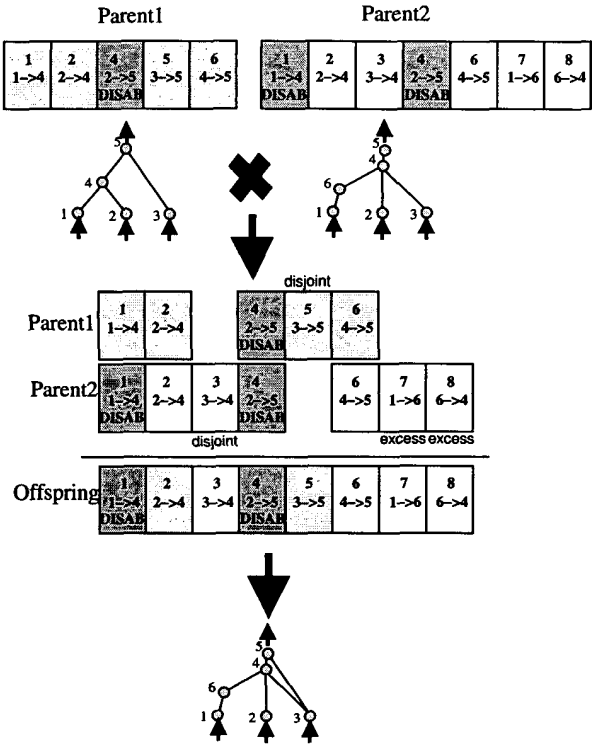

Fig. 3. **Matching up genomes for different network topologies using innovation numbers.** Although Parent 1 and Parent 2 look different, their innovation numbers (shown at the top of each gene) tell us which genes match up with which without the need for topological analysis.

C. *Protecting Innovation through Speciation*

Adding new structure to a network usually initially reduces fitness. However, NEAT speciates the population, so that individuals compete primarily within their own niches instead of with the population at large. This way, topological innovations are protected and have time to optimize their structure before they have to compete with other niches in the population. Speciation is commonly used in multimodal function optimization and in coevolution of modular systems, where its main function is to preserve diversity [8, 12]. We bring the idea to TWEANNs, where its main task is to protect innovation.

Historical markings make it possible for the system to divide the population into species based on topological similarity. 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 adjust the importance of the three factors, and the factor N, the number of genes in the larger genome, normalizes for genome size.

The distance measure δ allows us to speciate using a compatibility threshold δ_t. Genomes are tested one at a time; if a genome's 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.

As the reproduction mechanism for NEAT, we use *explicit fitness sharing* [4], where organisms in the same species must share the fitness of their niche, making it difficult for any one species to take over the population. 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:

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

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]

D. *Minimizing Dimensionality*

TWEANN algorithms typically start with an initial population of random topologies [2, 7, 21, 22]. Such topological diversity must be introduced from the start because new structure frequently does not survive in these methods, which do not protect innovation. However, it is not clear that such diversity is necessary or useful. A population of random topologies has a great deal of 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 begins with a uniform population of networks with no hidden nodes. Because NEAT protects innovation using speciation, it can start this way, minimally, and grow new structure only as necessary. New structure is introduced incrementally as structural mutations occur, and only those structures survive that are found to be useful through fitness evaluations. This way, NEAT searches through a minimal number of weight dimensions, significantly reducing the number of generations necessary to find a solution.

[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.

III. PERFORMANCE OF NEAT

A. *Pole Balancing as a Benchmark Task*

We demonstrate the efficacy of NEAT on the problem of balancing two poles simultaneously without giving velocity inputs to the network. This problem is a known benchmark in the reinforcement learning literature, which makes it possible to compare NEAT to other methods. Pole balancing has been used in RL and NE research for over 30 years [1, 3, 6, 7, 9, 11, 15, 17–20]. 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. We present the hardest such problem, balancing two pole simultaneously without velocities, in order to show that NEAT performs well on a difficult task.

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 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 (0.1m and 1.0m in our experiments) and respond differently to control inputs.

Taking away velocity information makes the task non-Markovian. It is difficult because the network must estimate an internal state in lieu of velocity, which requires recurrent connections.

Gruau et al. [7] 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.) The fitness function penalizes oscillations. Because the only NE methods that have solved this task were evaluated using this special fitness function, NEAT uses it on this task as well.

Under Gruau *et al.*'s criteria for a solution, the champion of each generation is tested on generalization to make sure it is robust. 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 1,000 times steps. In order to count as a solution, a network needs to generalize to at least 200 of the 625 initial states.

B. *Performance Comparison*

A number of NE methods have solved the easier double pole balancing *with* velocity information task.[2] NE methods take fewer evaluations than standard reinforcement learning methods such as Q-Learning on all levels of difficulty of pole balancing [6, 11]. Because double pole balancing *without* velocity information is a significantly more difficult task, to our knowledge only two systems have been demonstrated able to solve the problem so far. NEAT is compared to these two systems: Cellular Encoding [CE; 4], and Enforced Subpopulations [ESP; 3]. The success of CE was first attributed to its ability to evolve structures. However, ESP, a fixed-topology NE system, completed the task five times faster by restarting with a random number of hidden nodes whenever stuck. Our experiment aimed at showing that evolution of structure can lead to better performance if done right.

The experiment used a population of 1,000 NEAT networks. ESP evaluated 1,000 networks per generation, while CE needed a population of 16,384 networks to solve the problem. The top 60% of each species reproduced. The coefficients for measuring compatibility were $c_1 = 1.0$, $c_2 = 1.0$, $c_3 = 3.0$, and $C_t = 4.0$. The probability of adding a new node was 0.03, and the probability of adding a new link was 0.3. These parameters were chosen for their intuitive appeal: It makes sense to add links significantly more often than nodes, and we considered an average weight difference of 3.0 to be about as significant as one disjoint or excess gene. Performance is robust to moderate variations in these values.

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

Table 1. **Performance Comparison.** CE is Cellular Encoding of Gruau et al. [7]. ESP is Enforced Subpopulations of Gomez and Miikkulainen [6]. 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 ($p < 0.001$).

Table 1 shows that NEAT indeed 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.

IV. ABLATION STUDIES: DECONSTRUCTING NEAT

We have argued that NEAT's performance is due to historical markings, speciation, and incremental growth from minimal structure. In order to verify the contribution of each component, we performed a series of ablation studies. We disabled each component of NEAT separately and observed the effect on performance. We did not ablate historical markings directly because every component of the system relies on historical markings. Without historical markings, the system cannot function. All other components were systematically verified.

A. *Experimental Setup*

Ablations can have a significant detrimental effect on performance, potentially to the point where the system cannot solve the task at all. Therefore, we use an *easier* version of the pole

[2] In other work, NEAT took the fewest generations among 5 competing NE methods to solve the easier double pole balancing *with* velocity information benchmark task [16].

balancing problem for the ablation studies: double pole balancing *with* velocities. Because this task is significantly easier to solve, we used a smaller population of 150. A smaller population is sufficient because less diversity is necessary to search the solution space.

Method	Evaluations	Failure Rate
No-Growth NEAT	**30,239**	80%
Non-speciated NEAT	**25,600**	25%
Initial Random NEAT	**23,033**	5%
Full NEAT	**3,578**	0

Table 2. **NEAT Ablations Summary.** The table compares the average number of evaluations for a solution in the double pole balancing *with* velocities task (an easier task than that in Table 1). Results are averages over 20 runs, except full NEAT, which is an average over 120 runs. Each ablation leads to a weaker algorithm, showing that each component is necessary.

Table 2 shows the results of all the ablation studies, in terms of average evaluations required to find a solution. The main result is that the system performs significantly worse ($p \leq 0.001$) for every ablation. We will explain how each ablation was performed and interpret the results.

B. *No-growth Ablation*

In order to make no-growth NEAT comparable to fixed-topology NE, it was allowed to start with a fully-connected hidden layer of 10 hidden units, the same number as in past fixed-topology NE experiments [15, 19]. Without growth, the system was only able to use weight differences to speciate the population. Given 1,000 generations to find a solution, the ablated system could only find a solution 20% of the time! When it did find a solution, it took 8.5 times more evaluations than full NEAT. Clearly, speciation and historical markings alone do not account for full NEAT's performance; growing and complexifying solutions is a significant factor as well.

C. *Initial Random Ablation*

TWEANNs other than NEAT typically start with a random population [2, 7, 21]. Whether starting minimally is an advantage was tested by starting NEAT with random topologies. Each network in the initial population received between 1 and 10 hidden neurons with random connectivity (as implemented by Pujol and Poli [13]). Random-starting NEAT was 7 times slower than full NEAT on average. The random-starting system also failed to find a solution within 1,000 generations 5% of the time. The result suggests that starting randomly forces NE to search higher-dimensional spaces than necessary, thereby wasting time. If topologies are to grow, they should start out as small as possible.

D. *Non-speciated Ablation*

We have argued that speciation is important because it protects innovation and allows search to proceed in many different spaces simultaneously. To test this claim, speciation was ablated from the system. Because starting minimally *without* speciation stifles innovation, the non-speciated NEAT must

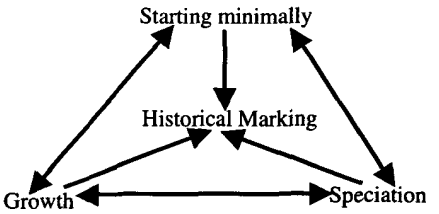

Fig. 4. **Dependencies among NEAT components.** Strong interdependencies can be identified among the different components of NEAT.

be started with an initial random population in order to provide diversity. The resulting non-speciated NEAT was able to find solutions, although it failed in 25% of the attempts. When it found a solution, it was 7 times slower on average than full NEAT.

The reason for the dramatic slowdown is that without speciation, the population quickly converges on whatever topology happens to initially perform best. Thus, a lot of diversity is drained immediately (within 10 generations). On average, this initially best-performing topology has about 5 hidden nodes. It is likely that the initial best-performing topology has more connections than necessary to solve the task. Thus, the population tends to converge to a relatively high-dimensional search space, even though the smaller networks in the initial population would have optimized faster. The smaller networks just do not get a chance because being small offers no immediate advantage in the initially random weight space. This result shows that speciation is crucial for innovation to be protected in a population of diverse topologies.

E. *Ablations Summary*

An important conclusion is that all of the parts of NEAT contribute to its performance. No single component works well without the aid of the other components (figure 4), and they are all needed to utilize the power of topology evolution. None of the system can work without historical markings because all of NEAT's functions utilize historical markings. If growth from minimal structure is removed, speciation can no longer help NEAT find spaces with minimal dimensionality. If speciation is removed, growth from minimal structures cannot proceed because structural innovations do not survive. When the system starts with a population of random topologies without speciation, the system quickly converges onto a non-minimal topology that just happens to be one of the best networks in the initial population. Thus, each component is necessary to make NEAT work.

V. DISCUSSION

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.

A system that can add new structure to already-optimized solutions has several potentially powerful properties. Speed

is one benefit. Smaller structures optimize faster, so the system is able to optimize the minimal number of connections necessary to obtain a solution. This benefit contrasts with the goal of a minimal *finished product* [22]. Our results show that the topology of the final solution is less important than the topologies of the networks along the way. Each increase in complexity resulting from new structure leads to a promising part of a higher dimensional space because most of the existing structure is already optimized.

A second benefit of complexification is an additional way to escape local optima. Not only can NEAT search the fitness landscape with mating and mutation, but it can *alter* the landscape itself with new structure. Thus, when a species in NEAT is on a local optimum, it is possible that by adding a new connection, a new dimension of freedom may open up, leading to a path away from the local optimum.

A parallel can be drawn between structure evolution in NEAT and incremental evolution [5, 19]. Incremental evolution is a method used to train a system to solve harder tasks than it normally could by training it on incrementally more challenging tasks. The idea is that NE is likely to get stuck on a local optimum when attempting to solve the harder task directly. However, after solving the easier version of the task first, the population is likely to be in a part of fitness space closer to the solution to the harder task, allowing it to avoid local optima. The difference between the incrementality of adding structure and general incremental evolution is that adding structure is *automatic* in NEAT whereas a sequence of progressively harder tasks requires human design.

VI. CONCLUSION

NEAT incrementally elaborates structure in a stochastic manner from a minimal starting point. Smaller structures optimize faster, so NEAT is able to find solutions faster than other neuroevolution methods. The ablation studies demonstrate that historical markings, speciation, and incremental growth from minimal structure are all integral components of efficient evolution of network structure. NEAT strengthens the analogy between GAs and natural evolution by both optimizing *and complexifying* solutions simultaneously.

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

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

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

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

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

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

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

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

[8] S. W. Mahfoud. *Niching Methods for Genetic Algorithms*. PhD thesis, U. of Illinois at Urbana-Champaign, May 1995.

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

[10] D. E. Moriarty. *Symbiotic Evolution of Neural Networks in Sequential Decision Tasks*. PhD thesis, Department of Computer Sciences, The University of Texas at Austin, 1997. Technical Report UT-AI97-257.

[11] D. E. Moriarty and Risto Miikkulainen. Efficient reinforcement learning through symbiotic evolution. *Machine Learning*, 22:11–32, 1996.

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

[13] J. C. F. Pujol and R. Poli. Evolving the topology and the weights of neural networks using a dual representation. *Special Issue on Evolutionary Learning of the Applied Intelligence Journal*, 8(1):73–84, January 1998.

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

[15] N. Saravanan and D. B. Fogel. Evolving neural control systems. *IEEE Expert*, pages 23–27, June 1995.

[16] K. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. Tech. Report AI2001-290, Dept. of Computer Sciences, The U. of Texas at Austin, 2001.

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

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

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

[20] A. P. Wieland. Evolving controls for unstable systems. In David S. Touretzky, Jeffrey L. Elman, Terrence J. Sejnowski, and Geoffrey E. Hinton, editors, *Connectionist Models: Proceedings of the 1990 Summer School*, pages 91–102. San Francisco, CA: Morgan Kaufmann, 1990.

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

[22] B. Zhang and H. Muhlenbein. Evolving optimal neural networks using genetic algorithms with Occam's razor. *Complex Systems*, 7:199–220, 1993.