Continual Coevolution through Complexification

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

In competitive coevolution, the goal is to establish an "arms race" that will lead to increasingly sophisticated strategies. However, in practice, the process often leads to idiosyncrasies rather than continual improvement. Applying the NEAT method for evolving neural networks to a competitive simulated robot duel domain, we will demonstrate that (1) as evolution progresses the networks become more complex, (2) complexification elaborates on existing strategies, and (3) if NEAT is allowed to complexify, it finds dramatically more sophisticated strategies than when it is limited to fixed-topology networks. The results suggest that in order to realize the full potential of competitive coevolution, genomes must be allowed to complexify as well as optimize over the course of evolution.

1 INTRODUCTION

In competitive coevolution, two or more populations of individuals evolve simultaneously in an environment where an increased fitness in one population leads to a decreased fitness for another. Ideally, competing populations will continually outdo one another, leading to an "arms race" of increasing sophistication (Dawkins and Krebs 1979; Van Valin 1973). In practice, evolution tends to find the simplest solutions that can win, meaning that strategies can switch back and forth between different idiosyncratic yet uninteresting variations (Darwen 1996; Floreano and Nolfi 1997; Rosin and Belew 1997). Several methods have been developed to encourage the arms race (Angeline and Pollack 1994; Rosin and Belew 1997). For example, a "hall of fame" can be used to ensure that current strategies remain competitive against strategies from the past. Although such techniques improve the performance of competitive coevolution, they do not directly encourage continual coevolution, i.e. creating new solutions that maintain existing capabilities. Much time is wasted evaluating solutions that are deficient in this way.

The problem is that in general genomes have a fixed set of genes mapping to a fixed phenotypic structure. Once a good strategy is found, the entire representational space of the genome is used to encode it. Thus, the only way to improve it is to *alter* the strategy, thereby sacrificing some of the functionality learned over previous generations.

In this paper, we propose a novel solution to this problem. The idea is to *complexify* or add structure to the dominant strategy, so that it does not merely become different, but rather *more elaborate*. This idea is implemented in a method for evolving increasingly complex neural networks, called NeuroEvolution of Augmenting Topologies (NEAT; Stanley and Miikkulainen 2001, 2002b,c). NEAT begins by evolving networks without any hidden nodes. Over many generations, new hidden nodes and connections are added, resulting in the complexification of the solution space. This way, more complex strategies elaborate on simpler strategies, focusing search on solutions that are likely to maintain existing capabilities.

NEAT was tested in a competitive robot control domain with and without complexification. The main results were that (1) evolution did complexify when possible, (2) complexification led to elaboration, and (3) significantly more sophisticated and successful strategies were evolved with complexification than without. These results imply that complexification allows coevolution to continually elaborate on successful strategies, resulting in an arms race that achieves a significantly higher level of sophistication than is otherwise possible.

We begin by describing the NEAT neuroevolution method, followed by a description of the robot duel domain and a discussion of the results.

2 NEUROEVOLUTION OF AUGMENTING TOPOLOGIES (NEAT)

The NEAT method of evolving artificial neural networks combines the usual search for appropriate network weights with complexification of the network structure. This approach is highly effective: NEAT outperforms other neuroevolution (NE) methods, e.g. on the benchmark double pole balancing task by a factor of five (Stanley and Miikkulainen 2001, 2002b,c). The NEAT method consists of solutions to three fundamental challenges in evolving neural network topology: (1) What kind of genetic representation would allow 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* so the most efficient solutions will be discovered? In this section, we explain how NEAT addresses each challenge.[1]

2.1 GENETIC ENCODING

Evolving structure requires a flexible genetic encoding. In order to allow structures to complexify, their representations must be dynamic and expandable. Each genome in NEAT 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 during crossover.

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 form the basis of complexification, occur in two ways (figure 1). 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.

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.

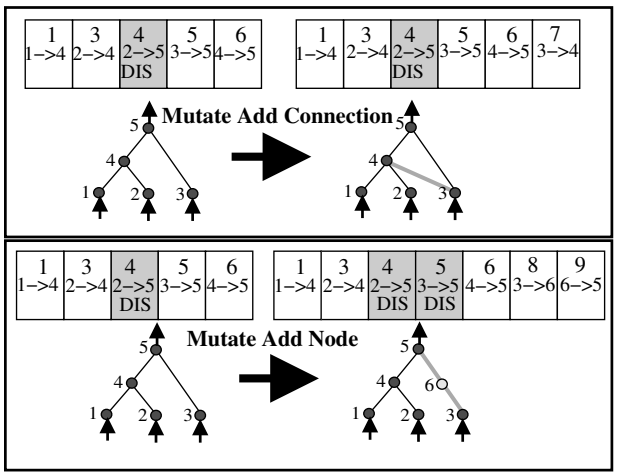

Figure 1: **The two types of structural mutation in NEAT.** Both types, adding a connection and adding a node, are illustrated with the genes above their phenotypes. The top number in each genome is the *innovation number* of that gene. The bottom two numbers denote the two nodes connected by that gene. The weight of the connection, also encoded in the gene, is not shown. The symbol *DIS* means that the gene is disabled, and therefore not expressed in the network. The figure shows how connection genes are appended to the genome when a new connection is added to the network and when a new node is added. Assuming the depicted mutations occurred one after the other, the genes would be assigned increasing innovation numbers as the figure illustrates, thereby allowing NEAT to keep an implicit history of the origin of every gene in the population.

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 1 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.

Through innovation numbers, the system now knows exactly which genes match up with which. 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

[1] A more comprehensive description of the NEAT method is given in Stanley and Miikkulainen (2001, 2002c).

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.

Historical markings allow NEAT to perform crossover without the need for expensive topological analysis. Genomes of different organizations and sizes stay compatible throughout evolution, and the problem of competing conventions (Radcliffe 1993) is essentially avoided. Such compatibility is essential in order to complexify structure.

2.2 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.

Historical markings make it possible for the system to divide the population into species based on topological similarity. We can measure the distance δ between two network encodings 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. 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, a compatibility threshold, 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* (Goldberg and Richardson 1987), where organisms in the same species must share the fitness of their niche, preventing any one species from taking over the population.

2.3 MINIMIZING DIMENSIONALITY THROUGH COMPLEXIFICATION

Unlike other systems that evolve network topologies and weights (Angeline et al. 1993; Gruau et al. 1996; Yao 1999; Zhang and Muhlenbein 1993), NEAT begins with a uniform population of simple networks with no hidden nodes. Speciation protects new innovations, allowing topological diversity to be gradually introduced over evolution.

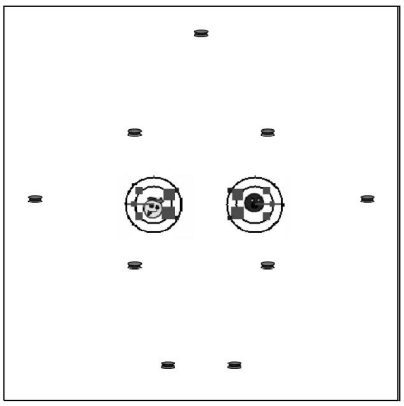

Figure 2: **The Robot Duel Domain.** The robots begin on opposite sides of the board facing away from each other as shown by the lines pointing away from their centers. The concentric circles around each robot represent the separate rings of opponent sensors and food sensors available to each robot. Each ring contains five sensors, which appear larger or smaller depending on their activations. From this initial position, neither robot has a positional advantage. The robots lose energy when they move around, yet they can gain energy by consuming food (shown as black dots). The food is placed in a horizontally symmetrical pattern around the middle of the board. The objective is to attain a higher level of energy than the opponent, and then collide with it. Because of the complex interaction between foraging, pursuit, and evasion behaviors, the domain allows for a broad range of strategies of varying sophistication. Animated demos of the robot duel domain are available at **www.cs.utexas.edu/users/nn/pages/research/neatdemo.html**.

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, and ensuring that networks become no more complex than necessary. In other words, NEAT searches for the optimal topology by *complexifying* when necessary.

3 THE ROBOT DUEL DOMAIN

To demonstrate the effect of complexification on competitive coevolution, a domain is needed where it is possible to develop increasingly sophisticated strategies and where the sophistication can be readily measured. A balance between the potential complexity of evolved strategies and their analyzability is difficult to strike. Pursuit and evasion tasks have been utilized for this purpose in the past (Gomez and Miikkulainen 1997; Jim and Giles 2000; Miller and Cliff 1994; Reggia et al. 2001), and can serve as a benchmark domain for competitive coevolution as well. While past experiments evolved either a predator or a prey, an interesting coevolution task can be established if the agents are instead equal and engaged in a duel. To win, an agent must de-

velop a strategy that outwits that of its opponent, utilizing structure in the environment.

In the robot duel domain, two simulated robots try to overpower each other (figure 2). The two robots begin on opposite sides of a rectangular room facing away from each other. As the robots move, they lose energy in proportion to the amount of force they apply to their wheels. Although the robots never run out of energy (they are given enough to survive the entire competition), the robot with higher energy can win by colliding with its competitor. In addition, each robot has a sensor indicating the difference in energy between itself and the other robot. To keep their energies high, the robots can consume food items, arranged in a symmetrical pattern in the room.

The robot duel task supports a broad range of sophisticated strategies that are easy to observe and interpret without expert knowledge. The competitors must become proficient at foraging, prey capture, and escaping predators. In addition, they must be able to quickly switch from one behavior to another. The task is well-suited to competitive coevolution because naive strategies such as forage-then-attack can be complexified into more sophisticated strategies such as luring the opponent to waste its energy before attacking.

The simulated robots are similar to Kheperas (Mondada et al. 1993). Each has two wheels controlled by separate motors. Five rangefinder sensors can sense food and another five can sense the other robot. Finally, each robot has an energy-difference sensor, and a single wall sensor.

The robots are controlled with neural networks evolved with NEAT. The networks receive all of the robot sensors as inputs, as well as a constant bias that NEAT can use to change the activation thresholds of neurons. They produce three motor outputs: Two to encode rotation either right or left, and a third to indicate forward motion power.

This complex robot-control domain allows competitive coevolution to evolve increasingly sophisticated and complex strategies, and can be used to benchmark coevolution methods.

4 EXPERIMENTS

In order to demonstrate how complexification contributes to continual coevolution, we ran four evolution trials with full NEAT and three trials with complexification turned off. The methodology is described below.

4.1 COMPETITIVE COEVOLUTION SETUP

In each evolution trial, 2 populations, each containing 256 genomes, were evolved simultaneously. In each generation, each population is evaluated against an intelligently chosen sample of networks from the other population. The population currently being evaluated is called the *host* population, and the population from which opponents are chosen is called the *parasite* population (Rosin and Belew 1997). The parasites are chosen for their quality and diversity, making host/parasite evolution more efficient and more reliable than random or round robin tournament.

A single fitness evaluation included two competitions, one for the east and one for the west starting position. That way, networks needed to implement general strategies for winning, independent of their starting positions. Host networks received a single fitness point for each win, and no points for losing. If a competition lasted 750 time steps with no winner, the host received 0 points.

In selecting the parasites for fitness evaluation, good use can be made of the speciation and fitness sharing that already occur in NEAT. Each host was evaluated against the champions of four species with the highest fitness. They are good opponents because they are the best of the best species, and they are guaranteed to be diverse because their compatibility must be outside the threshold δ_t (section 2.2). Another eight opponents were chosen randomly from a Hall of Fame (Rosin and Belew 1997) that contained population champions from all generations. Together, speciation, fitness sharing, and Hall of Fame comprise a state of the art competitive coevolution methodology. However, as our experimental results will show, complexification is the most important ingredient in establishing continual coevolution.

4.2 MONITORING PROGRESS IN COMPETITIVE COEVOLUTION

In order to track progress in coevolution, we need to be able to tell whether one strategy is better than another. Because the board configurations can vary during the game, networks were compared on 144 different food configurations from each side of the board, giving 288 total comparisons. The food configurations included the same 9 symmetrical food positions used during training, plus an additional 2 food items, which were placed in one of 12 different positions on the east and west halves of the board. Some starting food positions give an initial advantage to one robot or another, depending on how close they are to the robots' starting positions. We say that network a *is superior* to network b if a wins more comparisons than b out of the 288 total comparisons.

Given this definition of superiority, progress can be tracked. The obvious way to do it is to compare each network to others throughout evolution, finding out whether later strategies can beat more opponents than earlier strategies. For example, Floreano and Nolfi (1997) used a measure called

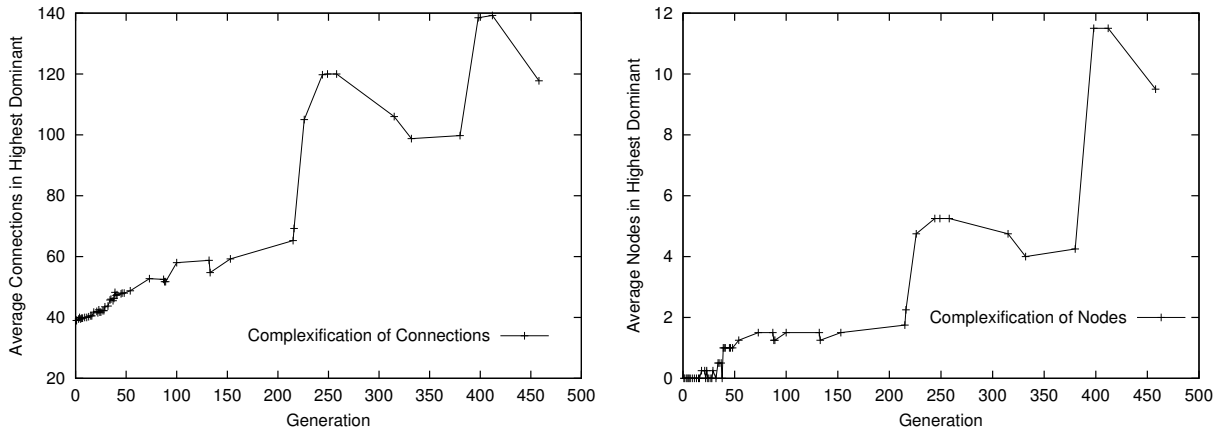

Figure 3: **Complexification of connections and nodes over generations.** The graphs depict the average number of connections and the average number of hidden nodes in the highest dominant network in each generation. Averages are taken over four complexifying runs. A hash mark appears every generation in which a new dominant strategy emerged in at least one of the four runs. The graphs show that as dominance increases, so does complexity level on average. The differences in complexity between the average final dominant and first dominant strategies are statistically significant for both connections and nodes ($p < 0.05$).

master tournament, in which the champion of each generation is compared to all other generation champions. Unfortunately, such methods are impractical in a time-intensive domain such as the robot duel competition. Moreover, the master tournament only shows how often each champion wins against other champions. In order to track strategic innovation, we need to identify *dominant strategies*, i.e. those that defeat *all previous* dominant strategies. This way, we can make sure that evolution proceeds by developing a progression of strictly more powerful strategies, instead of e.g. switching between alternative ones.

To meet this goal, we developed the *dominance tournament* method of tracking progress in competitive coevolution (Stanley and Miikkulainen 2002a). Let a *generation champion* be the winner of a 288 game comparison between the two population champions of a single generation. Let d_j be the jth dominant strategy to appear in the evolution. Then dominance is defined recursively:

- The first dominant strategy d_1 is the generation champion of the first generation;
- dominant strategy d_j, where $j > 1$, is a generation champion such that for all $i < j$, d_j is superior to (wins the 288 game comparison with) d_i.

This strict definition of dominance prohibits circularities. For example, d_4 must be superior to strategies d_1 through d_3, d_3 superior to both d_1 and d_2, and d_2 superior to d_1. The entire process of deriving a dominance hierarchy from a population is a *dominance tournament*, where competitors play all previous dominant strategies until they either lose a 288 game comparison, or win every comparison to previous dominant strategies, thereby becoming a new dominant strategy. Dominance tournaments require significantly fewer comparisons than the master tournament.

The question tested in the experiments is: Does the complexification of networks help attain high levels of dominance?

5 RESULTS

Each of the seven evolution trials lasted 500 generations, and took between 5 and 10 days on a 1GHz PIII processor, depending on the progress of evolution and sizes of the networks involved. The NEAT algorithm itself used less than 1% of this computation: the rest of the time was spent in evaluating networks in the robot duel task. Evolution of fully-connected topologies took about 90% longer than structure-growing NEAT because larger networks take longer to evaluate.

We define *complexity* as the number of nodes and connections in a network: The more nodes and connections there are in the network, the more complex behavior it can potentially implement. The results were analyzed to answer three questions: (1) As evolution progresses does it also continually complexify? (2) How is complexification utilized to create more sophisticated strategies? (3) Does complexification allow better strategies to be discovered than does evolving fixed-topology networks?

5.1 EVOLUTION OF COMPLEXITY

NEAT was run four times, each time from a different seed, to verify consistency of results. The highest levels of dominance achieved were 17, 14, 17, and 16, averaging at 16.

At each generation where the dominance level increased in at least one of the four runs, we averaged the number of connections and number of nodes in the current dominant strategy across all runs (figure 3). Thus, the graphs represent a total of 64 dominance transitions spread over 500 generations. The rise in complexity is dramatic, with the average number of connections tripling and the average number of hidden nodes rising from 0 to almost 10. In a smooth trend over the first 200 generations, the number of connections in the dominant strategy nearly doubles. During this early period, dominance transitions occur frequently (fewer prior strategies need to be beaten to achieve dominance). Over the next 300 generations, dominance transitions become more sparse, although they continue to occur.

Between the 200th and 500th generations a staircase pattern emerges, where complexity first rises dramatically, then settles, then abruptly increases again. The reason for this pattern is speciation. While one species is adding a large amount of structure, other species are optimizing the weights of less complex networks. While it is initially faster to grow structure until something works, such ad hoc constructions are eventually supplanted by older species that have been steadily optimizing for a long period of time. Thus, spikes in complexity occur when structural elaboration leads to a better strategy, and complexity slowly settles when older structures optimize their weights and overtake more recent structural innovations.

The results show more than just that the champions of each generation tend to become complex. The dominant strategies, i.e. the networks that have a strictly superior strategy to every previous dominant strategy, tend to be more complex the higher the dominance level. Thus, the results verify that continual progress in evolution is paired with increase in complexity.

5.2 SOPHISTICATION THROUGH COMPLEXIFICATION

To see how complexification contributes to evolution, let us observe the development of a sample dominant strategy, i.e. the evolution of the species that produced the winning network d_{17}, in the third run. Let us use S_k for the best network in S at generation k, and h_l for the lth hidden node to arise from a structural mutation over the course of evolution. We will track both strategic and structural innovations in order to see how they correlate. Let us begin with S_{100} (figure 4, left), when S had a mature zero-hidden-node strategy:

- S_{100}'s main strategy was to follow the opponent, putting it in a position where it might by chance collide with its opponent when its energy is up. However, S_{100} followed the opponent even when the opponent had more energy, leaving S_{100} vulnerable to attack. S_{100} did not clearly switch roles between foraging and chasing the enemy, causing it to miss opportunities to gather food.

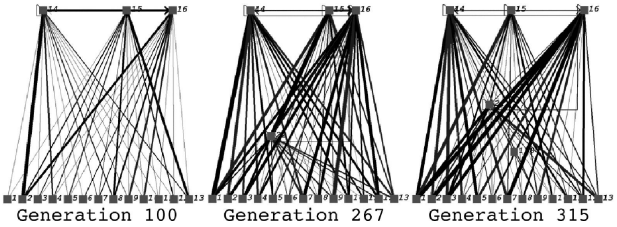

Figure 4: **Complexification of a Winning Species.** The best networks in the same species are depicted at landmark generations. Over generations, the networks in the species complexified and gained skills.

- S_{200}. During the next 100 generations, S evolved a *resting* strategy, which it used when it had significantly lower energy than the enemy. In such a situation, the robot stopped moving, while the other robot wasted energy running around. By the time the opponent gets close, its energy was often low enough to be attacked. The resting strategy is an example of improvement that can take place without complexification: it involved increasing the inhibition from the enemy difference sensor, thereby slightly modifying intensity of an existing behavior only.

- In S_{267} (figure 4, middle), a new hidden node, h_{22}, appeared. Node h_{22} arrived through an interspecies mating, and had been optimized for several generations already, Node h_{22} gave the robot the ability to change its behavior at once into a consistent all-out attack. Because of this new skill, S_{267} no longer needed to follow the enemy closely at all times, leaving it to focus on collecting food. By implementing this new strategy through a new node, it was possible not to interfere with the already existing resting strategy, so that S now switched roles between resting when in danger to attacking when high on energy. This way, the new structure resulted in strategic elaboration.

- In S_{315} (figure 4, right), h_{172} split a link between an input sensor and h_{22}. Replacing a direct connection with a sigmoid function greatly improved S_{315}'s ability to attack at appropriate times, leading to very accurate role switching between attacking and foraging. S_{315} would try to follow the opponent from afar focusing on resting and foraging, and only zoom in for attack when victory was certain. This final structural addition shows how new structure can greatly improve the accuracy and timing of existing behaviors.

The analysis above shows that in some cases, weight optimization alone can produce improved strategies. However, when those strategies need to be extended, adding new structure allows the new behaviors to coexist with old strategies. Also, in some cases it is necessary to add complexity to make the timing or execution of the behavior more accurate. These results show how complexification can be utilized to produce sophistication in competitive coevolution.

5.3 COMPLEXIFICATION VS. FIXED-STRUCTURE EVOLUTION

To see whether complexifying coevolution is more powerful than standard non-complexifying coevolution, we ran three trials with fixed, fully-connected topologies. To make the comparison fair, the fixed-topology networks in the first two trials had 10 hidden nodes, as did the winning networks of complexifying runs on average. In the third trial, fixed-topology networks had only five hidden nodes, which gives them the same number of connections as the complexifying trials. In the first trial, the hidden nodes were fully connected to the outputs. In the other two trials, the *inputs* were also fully connected to outputs. In all standard runs, the hidden layer was fully recurrent, because complexifying runs were found to evolve recurrent connections. Although topologies were fixed, evolution continued to speciate using weight differences.

Fixed-topology Run	Highest Dom.	Equivalent Dom. Level (out of 16)	Equivalent Generation (out of 500)
1: 10 Hidden Node	12	5.5	17.75
2: 10 Hidden Node, Direct Connections	14	9.25	39
3: 5 Hidden Nodes, Direct Connections	10	10.75	65.5

Table 1: **Comparing the dominant strategies in the fixed-topology (i.e. standard) coevolution with those of complexifying coevolution.** The second column shows how many levels of dominance were achieved in the standard coevolution. The third column gives the highest dominance level in complexifying runs that the dominant from the standard run can defeat and the fourth column shows its average generation. The main result is that the level of sophistication reached by standard coevolution is significantly lower than that reached by complexifying coevolution.

In Table 1, the relative sophistication of the strategies developed are compared to those in complexifying coevolution. We compared the highest dominant network from each of the standard runs with the entire dominance hierarchies of all the complexifying runs. The table reports the highest dominance level *within the complexifying runs* that the best fixed-topology network can defeat on average. In all cases, the standard strategy reaches only the middle levels of the hierarchy, i.e. 5.5, 9.25, and 10.75 out of possible 16. Complexifying coevolution on average found 7 levels of dominance *above* the most sophisticated strategies of standard coevolution. Considering that high levels of dominance are much more difficult to attain than low levels, it is clear that complexifying coevolution develops a dramatically higher level of sophistication.

Another significant result is that NEAT developed equivalent strategies very early in evolution. For example, the second standard run stopped producing new dominant strategies after the 169th generation, followed by 331 consecutive generations without any additional dominant strategies. This network can defeat about the 9th dominant from complexifying coevolution, which was found on average in the 39th generation. In other words, standard coevolution is considerably slower in finding even the first few steps in the dominance hierarchy.

In summary, complexifying coevolution progresses faster and discovers significantly more sophisticated solutions than standard coevolution.

6 DISCUSSION AND FUTURE WORK

Evolution in nature acts as both an optimizer *and* a complexifier. Not only do existing genes express different alleles, but *new* genes are added occasionally through a process called gene amplification (Darnell and Doolittle 1986). Therefore, we should expect to find that complexification can also play a role in models of open-ended evolution, such as competitive coevolution, thus strenghening the analogy of evolutionary computation with nature.

Indeed, as the results confirm, complexification does enhance the capability of competitive coevolution to find sophisticated strategies. Complexification encourages continual *elaboration*, whereas evolution of fixed-structures proceeds primarily by *alteration*. When a fixed genome is used to represent a strategy, that strategy can be optimized, but it is not possible to complexify without sacrificing some of the knowledge that is already present. In contrast, if new genetic material can be added, then sophisticated elaborations can be layered above existing structure.

Complexification can find solutions that are difficult to find by evolving fixed structure. In fixed evolution, the complexity must be guessed just right: too little structure will make it impossible to solve the problem and too much will make the search space too large to search efficiently. A complexifying system saves the user from such concerns.

Complexification is a new and still largely unexplored research area. How complexifying systems work in general, and what the best ways are to describe such systems are open questions at this point. Although evolution is the best known complexifier, that does not mean it is the only one.

Organizations (such as corporations and governments) are also complexifying systems, with new positions being created that only have meaning relative to positions that previously existed. We need to develop an abstract description of complexification, from which we can derive theories and rules for understanding and utilizing complexification in different domains.

7 CONCLUSION

We hypothesized that complexification of genomes can lead to continual coevolution of increasingly sophisticated strategies. Experimental results showed three trends: (1) as evolution progresses, complexity of solutions increases, (2) evolution uses complexification to elaborate on existing strategies, and (3) complexifying coevolution is significantly more successful in finding highly sophisticated strategies than evolution of fixed structures. These results suggest that complexification is a crucial component of continual coevolution.

Acknowledgments

This research was supported in part by the National Science Foundation under grant IIS-0083776 and by the Texas Higher Education Coordinating Board under grant ARP-003658-476-2001.

References

Angeline, P. J., and Pollack, J. B. (1994). Competitive environments evolve better solutions for complex tasks. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, 264–270.

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.

Darnell, J. E., and Doolittle, W. F. (1986). Speculations on the early course of evolution. *Proceedings of the National Academy of Sciences, USA*, 83:1271–1275.

Darwen, P. J. (1996). *Co-Evolutionary Learning by Automatic Modularisation with Speciation*. PhD thesis, School of Computer Science, University College, University of New South Wales.

Dawkins, R., and Krebs, J. R. (1979). Arms races between and within species. *Proceedings of the Royal Society of London Series B*, 205:489–511.

Floreano, D., and Nolfi, S. (1997). God save the red queen! Competition in co-evolutionary robotics. *Evolutionary Computation*, 5.

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.

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.

Jim, K.-C., and Giles, C. L. (2000). Talking helps: Evolving communicating agents for the predator-prey pursuit problem. *Artificial Life*, 6(3):237–254.

Miller, G., and Cliff, D. (1994). Co-evolution of pursuit and evasion i: Biological and game-theoretic foundations. Technical Report CSRP311, School of Cognitive and Computing Sciences, University of Sussex, Brighton, UK.

Mondada, F., Franzi, E., and Ienne, P. (1993). Mobile robot miniaturization: A tool for investigation in control algorithms. In *Proceedings of the Third International Symposium on Experimental Robotics*, 501–513.

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

Reggia, J. A., Schulz, R., Wilkinson, G. S., and Uriagereka, J. (2001). Conditions enabling the evolution of inter-agent signaling in an artificial world. *Artificial Life*, 7:3–32.

Rosin, C. D., and Belew, R. K. (1997). New methods for competitive evolution. *Evolutionary Computation*, 5.

Stanley, K. O., and Miikkulainen, R. (2001). Evolving neural networks through augmenting topologies. Technical Report AI2001-290, Department of Computer Sciences, The University of Texas at Austin.

Stanley, K. O., and Miikkulainen, R. (2002a). The dominance tournament method of monitoring progress in coevolution. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002) Workshop Program*. San Francisco, CA: Morgan Kaufmann.

Stanley, K. O., and Miikkulainen, R. (2002b). Efficient evolution of neural network topologies. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC'02)*. IEEE.

Stanley, K. O., and Miikkulainen, R. (2002c). Efficient reinforcement learning through evolving neural network topologies. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002)*. San Francisco, CA: Morgan Kaufmann.

Van Valin, L. (1973). A new evolutionary law. *Evolution Theory*, 1:1–30.

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.