Competitive Coevolution through Evolutionary Complexification

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

Technical Report UT-AI-TR-02-298

December, 2002

Abstract

Two major goals in machine learning are the discovery of complex multidimensional solutions and continual improvement of existing solutions. In this paper, we argue that *complexification*, i.e. the incremental elaboration of solutions through adding new structure, achieves both these goals. We demonstrate the power of complexification through the NeuroEvolution of Augmenting Topologies (NEAT) method, which evolves increasingly complex neural network architectures. NEAT is applied to an open-ended coevolutionary robot duel domain where robot controllers compete head to head. Because the robot duel domain supports a wide range of sophisticated strategies, and because coevolution benefits from an escalating arms race, it serves as a suitable testbed for observing the effect of evolving increasingly complex controllers. The result is an arms race of increasingly sophisticated strategies. When compared to the evolution of networks with fixed structure, complexifying networks discover significantly more sophisticated strategies. The results suggest that in order to realize the full potential of evolution, and search in general, solutions must be allowed to complexify as well as optimize.

1 Introduction

Evolutionary algorithms are a class of algorithms that can be applied to open-ended learning problems in Artificial Intelligence. Traditionally, such algorithms evolve fixed-length genomes under the assumption that the space of the genome is sufficient to encode the solution. A genome containing n genes encodes a single point in an n-dimensional search space. In many cases, a solution is known to exist somewhere in that space. For example, the global maximum of a function of three arguments *must* exist in the three dimensional space defined by those arguments. Thus, a genome of three genes can encode the location of the maximum.

However, many common structures are defined by an arbitrary number of parameters. In particular, those solution types that can contain a variable number of parts can be represented by *any* number of genes. For example, the number of parts in neural networks, cellular automata, and electronic circuits can vary (Miller et al. 200a; Mitchell et al. 1996; Stanley and Miikkulainen 2002d). In fact, two neural networks with

different numbers of connections and nodes can approximate the same function (Cybenko 1989). Thus, it is not clear what number of genes is appropriate for solving a particular problem. Because of this ambiguity, researchers evolving fixed-length genotypes must use heuristics, such as smaller neural networks generalizing better than larger ones, in order to estimate *a priori* the appropriate number of genes to encode such structures.

A major obstacle to using fixed-length encodings is that heuristically determining the appropriate number of genes becomes impossible for very complex problems. For example, how many nodes and connections are necessary for a neural network that controls a ping-pong playing robot? Or, how many bits are needed in the neighborhood function of a cellular automata that performs information compression? The answers to these questions can hardly be based on empirical experience or analytic methods, since little is known about the solutions. One possible approach is to simply make the genome extremely large, so that the space it encodes is extremely large and a solution is likely to lie somewhere within. Yet the larger the genome, the higher dimensional the space that evolution needs to search. Even if a ping-pong playing robot lies somewhere in the 10,000 dimensional space of a 10,000 gene genome, searching such a space may take prohibitively long.

Even more problematic are open-ended problems where phenotypes are meant to improve indefinitely and there is no known final solution. For example, in competitive games, estimating the complexity of the "best" possible player is difficult because making such an estimate implicitly assumes that no better player can exist. How could we ever know that? Moreover, many artificial life domains are aimed at evolving increasingly complex artificial creatures for as long as possible (Maley 1999). Fixing the size of the genome in such domains also fixes the maximum complexity of evolved creatures, defeating the purpose of the experiment.

In this paper, we argue that the right way to evolve arbitrarily structured phenotypes is to start evolution with a population of small, simple genomes and systematically *complexify* solutions over generations by adding new genes. That way, evolution begins searching in a small easily-optimized space, and adds new dimensions as necessary. This approach is more likely to discover highly complex phenotypes than an approach that begins searching directly in the intractably large space of complete solutions. In fact, natural evolution has itself utilized this strategy, occasionally adding new genes that lead to increased phenotypic complexity (Martin 1999; Section 2). In biology, this process is called *complexification*, which is why we use this term to describe our approach as well.

When a good strategy is found in a fixed-length genome, 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 contrast, complexification *elaborates* on the existing strategy by adding new structure without changing the existing representation. Thus the strategy does not only become different, but the number of possible responses to situations it may face increases (figure 1).

This idea is implemented in a method for evolving increasingly complex neural networks, called NeuroEvolution of Augmenting Topologies (NEAT; Stanley and Miikkulainen 2002b,c,d). 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. We use NEAT to demonstrate the power of complexification.

NEAT was tested in a competitive robot control domain with and without complexification. Coevolution was used to evolve robot controllers against each other to find better strategies. We chose this domain because it is open-ended; there is no known optimal strategy but it is possible to come up with increasingly more sophisticated strategies indefinitely. The main results were that (1) evolution did complexify when possible, (2) complexification led to elaboration, and (3) significantly more sophisticated and successful

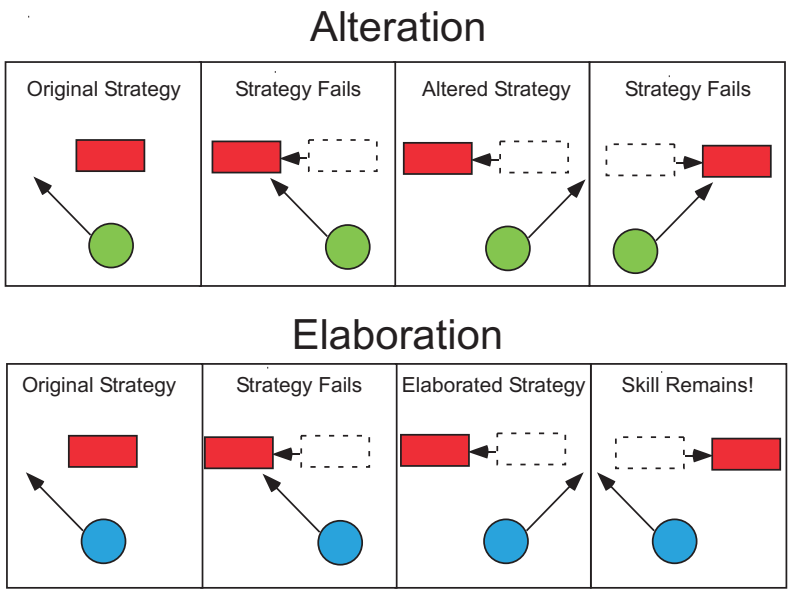

Figure 1: **Alteration vs. elaboration example.** A robot (depicted as a circle) evolves to avoid an obstacle. In the alteration scenario (top), the robot first evolves a strategy to go around the left side of the obstacle. However, the strategy fails in a future generation when the obstacle begins moving to the left. Thus, the robot alters its strategy by evolving the tendency to move right instead of left. However, when the obstacle later moves right, the new, altered, strategy fails because the robot did not retain its old ability to move left. In the elaboration scenario (bottom), the original strategy of moving left also fails. However, instead of altering the strategy, it is *elaborated* by adding a new ability to move right as well. Thus, when the obstacle later moves right, the robot still has the ability to avoid it by using its original strategy. Elaboration is necessary for a coevolutionary arms race to emerge and it can be achieved through complexification.

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 reviewing biological support for complexification, as well as past work in coevolution, followed by a description of the NEAT method, and experimental results.

2 Background

2.1 Complexification in Nature

Mutation in nature not only results in optimizing existing structures: New genes are occasionally added to the genome, allowing evolution to perform a complexifying function over and above optimization. In addition, complexification is protected in nature in that interspecific mating is prohibited. Such speciation creates important dynamics differing from standard GAs. In this section, we discuss these important characteristics of natural evolution as a basis for our approach to utilize them computationally in genetic algorithms.

Gene duplication is a special kind of mutation in which one or more parental genes are copied into an offspring's genome more than once. The offspring then has redundant genes expressing the same proteins. Gene duplication has been responsible for key innovations in overall body morphology over the course of natural evolution (Amores et al. 1998; Carroll 1995; Force et al. 1999; Martin 1999).

A major gene duplication event occurred around the time that vertebrates separated from invertebrates. The evidence for this duplication centers around *HOX genes*, which determine the fate of cells along the anterior-posterior axis of embryos. HOX genes are crucial in shaping the overall pattern of developmental in embryos. In fact, differences in HOX gene regulation explain a great deal of arthropod and tetrapod diversity (Carroll 1995). Amores et al. (1998) explain that since invertebrates have a single HOX cluster while vertebrates have four, cluster duplication must have significantly contributed to elaborations in vertebrate body-plans. The additional HOX genes took on new roles in regulating how vertebrate anterior-posterior axis develops, considerably increasing body-plan complexity. Although Martin (1999) argues that the additional clusters can be explained by many single gene duplications accumulating over generations, as opposed to massive whole-genome duplications, researchers agree that gene duplication contributed significantly to important body-plan elaboration.

A detailed account of how duplicate genes can take on novel roles was given by Force et al. (1999): Base pair mutations in the generations following duplication *partition* the initially redundant regulatory roles of genes into separate classes. Thus, the embryo develops in the same way, but the genes that determine overall body-plan are confined to more specific roles, since there are more of them. The partitioning phase completes when redundant clusters of genes are separated enough that they no longer produce identical proteins at the same time. After partitioning, mutations within the duplicated cluster of genes alter different steps in development than mutations within the original cluster. In other words, duplication creates more points at which mutations can occur. In this way, developmental processes complexify.

In order to implement this idea in artificial evolutionary systems we are faced with two major challenges. First, such systems evolve variable-length genomes, which can be difficult to cross over without losing information. For example, depending on when duplications occurred in the ancestral histories of two different genomes, the same gene may exist at different positions. Conversely, different genes may exist at the same position. Thus, artificial crossover may lose essential genes through misalignment. Second, it may be difficult for a variable-length genome GA to find innovative solutions; Optimizing many genes takes longer than optimizing only a few, meaning that more complex genotypes may be eliminated from the population before they have a sufficient opportunity to be optimized.

How has nature solved these problems? First, nature has a mechanism for aligning genes with their proper counterparts during crossover, so that data is not lost nor obscured. This alignment process has been most clearly observed in *E. coli* (Radding 1982; Sigal and Alberts 1972). A special protein called *RecA* takes a single strand of DNA and aligns it with another strand by attaching the strands at genes that express the same traits, which are called *homologous genes*. The process by which RecA protein aligns homologous genes is called *synapsis*. In experiments in vitro, researchers have found that RecA protein does not complete the process of synapsis on fragments of DNA that are not homologous (Radding 1982).

Second, innovations in nature are protected through speciation. Organisms with significantly divergent genomes never mate because they are in different species. Thus, organisms with larger genomes compete for mates among their own species, instead of with the population at large. That way, organisms that may initially have lower fitness than the general population still have a chance to reproduce, giving novel concepts a chance to realize their potential without being prematurely eliminated.

It turns out complexification is also possible in evolutionary computation if abstractions of synapsis and speciation are made part of the genetic algorithm. The NEAT method (section 3) is an implementation of this idea: the genome is complexified by adding new genes which in turn encode new structure in the phenotype, as in biological evolution.

Complexification is especially powerful in open-ended domains where the goal is to continually generate more sophisticated strategies. Competitive coevolution is a particularly important such domain, as will be reviewed in the next section.

2.2 Competitive Coevolution

In competitive coevolution, individual fitness is evaluated through direct competition with other individuals in the population, rather than through an objective fitness measure. In other words, fitness signifies only the relative strengths of solutions; an increased fitness in one solution leads to a decreased fitness for another. Ideally, competing solutions will continually outdo one another, leading to an ”arms race” of increasingly better solutions (Dawkins and Krebs 1979; Rosin 1997; Van Valin 1973).

In practice, it is difficult to establish an arms race. 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 1993; Ficici and Pollack 2001; Noble and Watson 2001; 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. Recently, Ficici and Pollack (2001) and Noble and Watson (2001) introduced a promising method called *Pareto coevolution*, which finds the best learners and the best teachers in two populations by casting coevolution as a multiobjective optimization problem.

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. For example, no matter how well selection is performed, or how well competitors are chosen, if no better solution exists in the fixed solution space, elaboration is impossible since the global optimum has already been reached. Moreover, it may occasionally be easier to escape a local optimum by adding a new dimension of freedom to the search space than by jumping back out into the same space that led to the optimum in the first place.

Complexification is an appropriate technique for establishing a coevolutionary arms race. Complexification naturally elaborates strategies by adding new dimensions to the search space. Thus, progress can be made indefinitely long: even if a global optimum is reached in the search space of solutions, new dimensions can be added, opening up a higher-dimensional space in which even higher optima may exist.

To test this idea experimentally, we chose a robot duel domain that combines predator/prey interaction and food foraging in a novel head-to-head competition (Section 4). We use this domain to demonstrate how NEAT uses complexification to continually elaborate on strategies. The next section describes the NEAT neuroevolution method, followed by a description of the robot duel domain and a discussion of the results.

3 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 2002b,c,d). 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? 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* so the most efficient solutions will be discovered? Our solution is to start from a minimal structure and grow only when necessary. In this section, we explain how NEAT addresses each challenge.

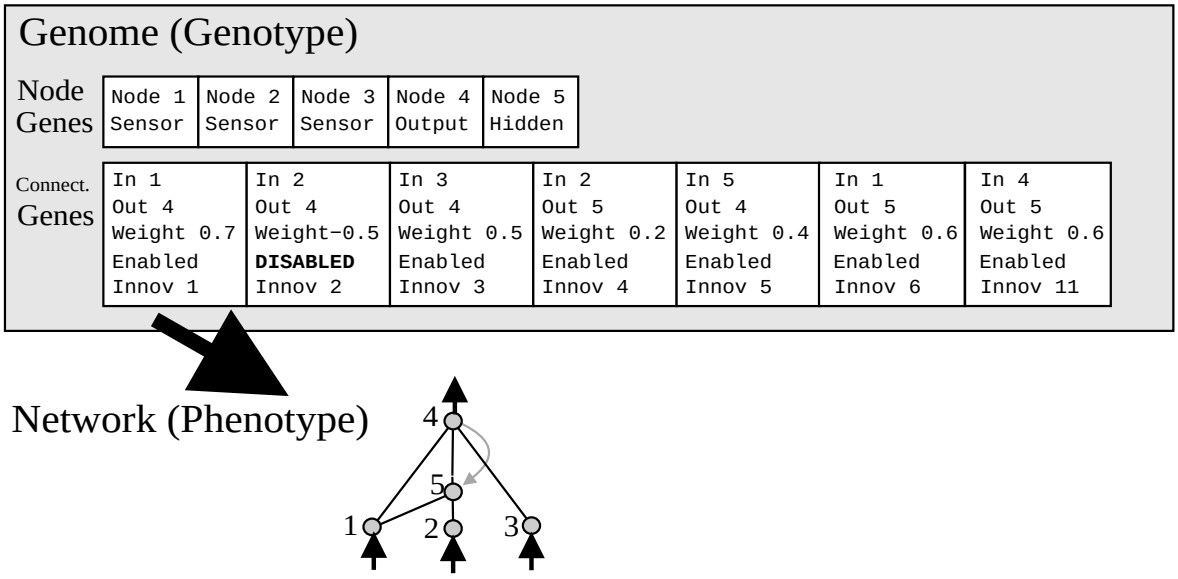

Figure 2: **A NEAT genotype to phenotype mapping example.** A genotype is depicted that produces the shown phenotype. There are 3 input nodes, one hidden, and one output node, and seven connection definitions, one of which is recurrent. The second gene is disabled, so the connection that it specifies (between nodes 2 and 4) is not expressed in the phenotype. In order to allow complexification, genome length is unbounded.

3.1 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. (figure 2). 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 3). 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. The new nonlinearity in the connection changes the function slightly, but new nodes can be immediately integrated into the network, as opposed to adding extraneous structure that would have to be evolved into the network later.

Through mutation, the genomes in NEAT will gradually get larger. Genomes of varying sizes will result, sometimes with different connections at the same positions. A difficult problem for crossover, with numerous differing topologies and weight combinations, is an inevitable result of allowing genomes to grow unbounded. Yet such growth is necessary if complexification is to occur. How can NE cross over differently sized genomes in a sensible way? The next section explains how NEAT addresses this problem.

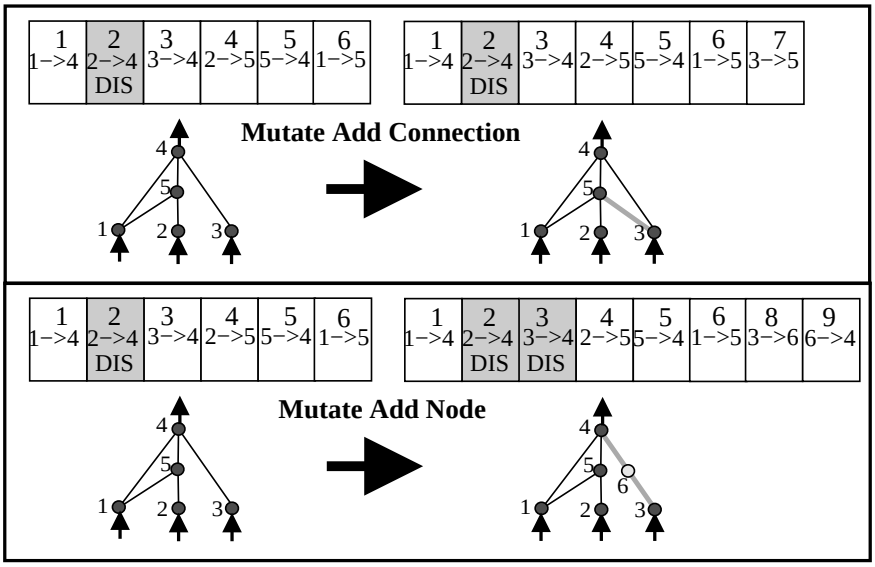

Figure 3: **The two types of structural mutation in NEAT.** Both types, adding a connection and adding a node, are illustrated with the genes above their phenotypes. The top number in each genome is the *innovation number* of that gene. The bottom two numbers denote the two nodes connected by that gene. 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.

3.2 Tracking Genes through Historical Markings

It turns out that there is unexploited information in evolution that tells us exactly which genes match up between *any* individuals in the population. That information is the historical origin of each gene in the population. Two genes with 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 3 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.

A possible problem is that the same structural innovation will receive different innovation numbers in the same generation if it occurs by chance more than once. However, by keeping a list of the innovations that occurred in the current generation, it is possible to ensure that when the same structure arises more than once through independent mutations in the same generation, each identical mutation is assigned the same innovation number. Thus, there is no resultant explosion of innovation numbers.

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

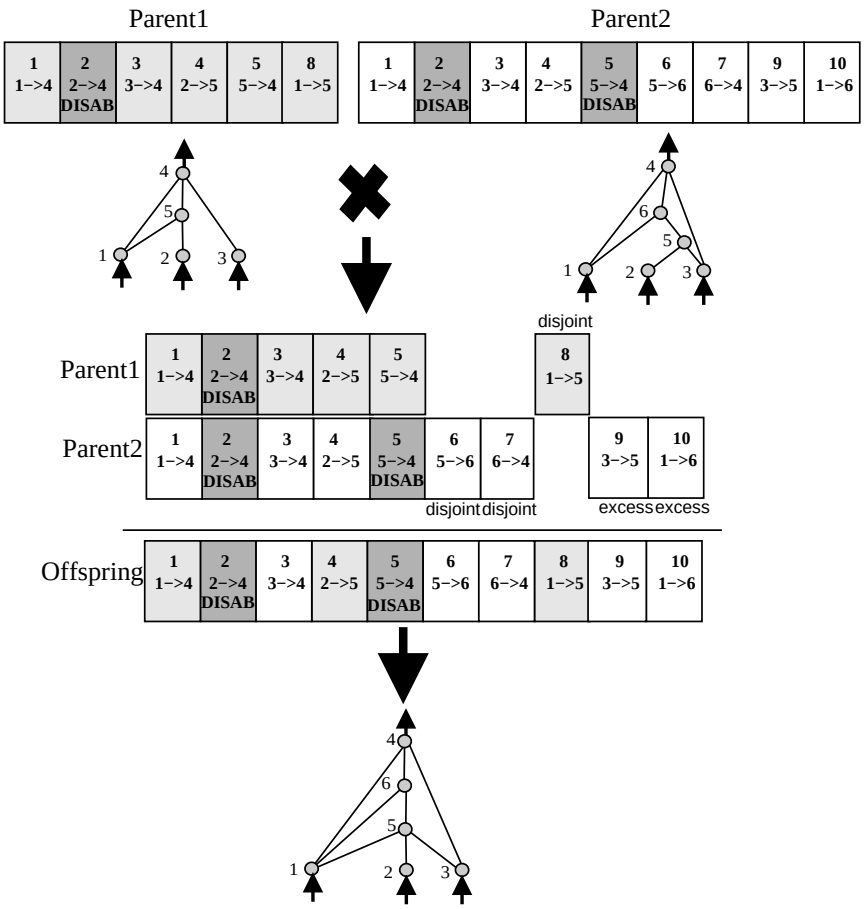

Figure 4: **Matching up genomes for different network topologies using innovation numbers.** Although Parent 1 and Parent 2 look different, their innovation numbers (shown at the top of each gene) tell us which genes match up with which without the need for topological analysis. Even without any topological 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, equal fitnesses are assumed, so the disjoint and excess genes from both parents are inherited randomly. The disabled genes may become enabled again in future generations: there's a preset chance that an inherited gene is disabled if it is disabled in either parent.

4). 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.

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 comparing different topologies (Radcliffe 1993) is essentially avoided. Such compatibility is essential in order to complexify structure. However, it turns out that a population of varying complexities cannot maintain topological innovations on its own. 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.

3.3 Protecting Innovation through Speciation

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. Protecting innovation through speciation follows the philosophy that new ideas must be given time to reach their potential before they are prematurely eliminated.

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 ($\overline{W}$):

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

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 (N can be set to 1 if both genomes are small, i.e., consist of fewer than 20 genes). 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 from the *previous generation* where this condition is satisfied, so that no genome is in more than one species. If a genome is not compatible with any existing species, a new species is created.

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 adjusted fitness f'_i for organism i is calculated according to its distance δ from every other organism j in the population:

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

The sharing function sh is set to 0 when distance $\delta(i,j)$ is above the threshold δ_t; otherwise, $\mathrm{sh}(\delta(i,j))$ is set to 1 (Spears 1995). Thus, $\sum_{j=1}^{n} \mathrm{sh}(\delta(i,j))$ reduces to the number of organisms in the same species as organism i. This reduction is natural since species are already clustered by compatibility using the threshold δ_t. Every species is assigned a potentially different number of offspring in proportion to the sum of adjusted fitnesses f'_i of its member organisms. Species then reproduce by first eliminating the lowest performing members from the population. The entire population is then replaced by the offspring of the remaining organisms in each species.

The net effect of speciating the population is that structural 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 complexification from a simple starting structure, as detailed in the next section.

3.4 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

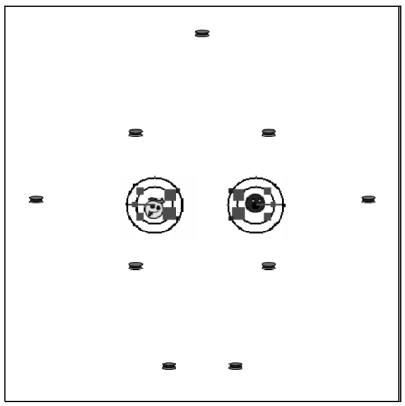

Figure 5: **The Robot Duel Domain.** The robots begin on opposite sides of the board facing away from each other as shown by the lines pointing away from their centers. The concentric circles around each robot represent the separate rings of opponent sensors and food sensors available to each robot. Each ring contains five sensors, which appear larger or smaller depending on their activations. The robots lose energy when they move around, and gain energy by consuming food (shown as small sandwiches). 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 such evolved strategies are available at **www.cs.utexas.edu/users/kstanley/demos.html**.

introduced over evolution. Thus, 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, and ensuring that networks become no more complex than necessary. This gradual production of increasingly complex structures constitutes *complexification*. In other words, NEAT searches for the optimal topology by complexifying when necessary.

4 The Robot Duel Domain

Our hypothesis is that complexification allows discovering more sophisticated strategies, i.e. strategies that are more effective, flexible, and general, and include more components and variations than do simple strategies. To demonstrate this effect, we need a domain where it is possible to develop increasingly sophisticated strategies, and where sophistication can be readily measured. A coevolution domain is particularly appropriate because a sustained arms race should lead to increasing sophistication.

In choosing the domain, it is difficult to strike a balance between being able to evolve complex strategies and being able to analyze and understand them. 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 develop 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 5). The two robots begin on opposite sides of a rectangular room facing away from each other. As the robots move, they lose

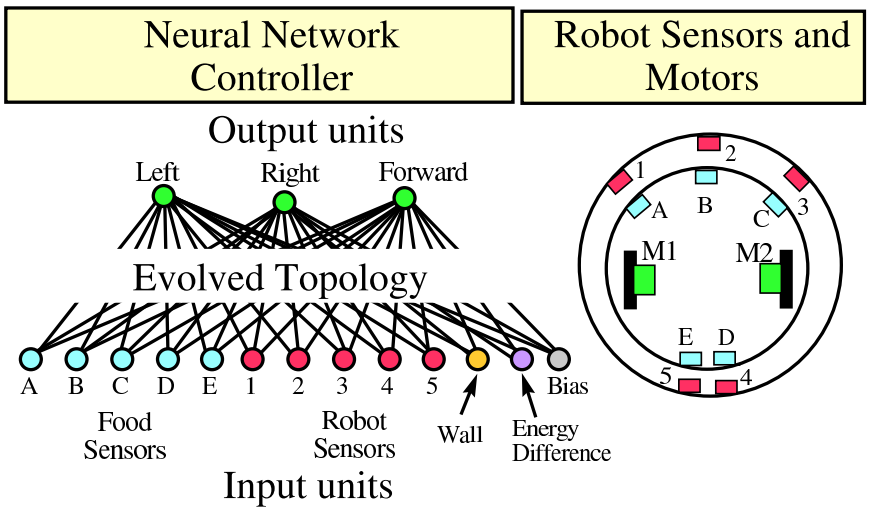

Figure 6: **Robot Neural Networks.** Five food sensors and five robot sensors detect the presence of objects around the robot. A single wall sensor indicates proximity to walls, and the energy difference sensors tells the robot how its energy level differs from that of its opponent. This difference is important because the robot with lower energy loses if the robots collide. The three motor outputs are mapped to forces that control the left and the right wheel.

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 wins when it collides 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. 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; figure 6). 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. These three values are then translated into forces to be applied to the left and right wheels of the robot.

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

5 Experiments

In order to demonstrate how complexification enhances performance, we ran 23 500-generation runs of coevolution in the robot duel domain. The full NEAT method was enabled in 13 runs, and complexification

was turned off in the remaining ten, in order to see how complexified networks compared to those with fixed topologies throughout evolution. The competitive coevolution setup is described first, followed by an overview of the dominance tournament method for monitoring progress.

5.1 Competitive Coevolution Setup

The evolution in the robot duel domain is *open-ended* in that it never reaches a final solution. Thus, the question in such a domain is whether *continual coevolution* will take place, i.e. whether increasingly sophisticated strategies will appear throughout the run or eventually stop being produced. Complexification should allow continual coevolution. The experiment has to be set up carefully for this process to emerge, and to be able to identify it when it does.

In competitive coevolution, every network should play a suitable number of games to establish a good measure of fitness. To encourage interesting and sophisticated strategies, networks should play a diverse and high quality sample of possible opponents. One way to accomplish this goal is to evolve two separate populations. 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 one based on 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 3.3). 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 competent competitive coevolution methodology. Appendix A contains the NEAT system parameter values used in the experiments.

It should be noted that complexification does not depend on the particular coevolution methodology. For example *Pareto coevolution* (Ficici and Pollack 2001; Noble and Watson 2001) could have been used as well, and the advantages of complexification would be the same. However, Pareto coevolution requires every member of one population to play every member of the other, and the running time in this domain would have been prohibitively long.

In order to interpret experimental results, a method is needed for analyzing progress in competitive coevolution. The next section describes such a method.

5.2 Monitoring Progress in Competitive Coevolution

A competitive coevolution run returns a record of every generation champion from both populations. The question is how can a sequence of increasingly sophisticated strategies be identified in this data, if one exists? This section describes the *dominance tournament* method for monitoring progress in competitive coevolution (Stanley and Miikkulainen 2002a) that allows us to do that.

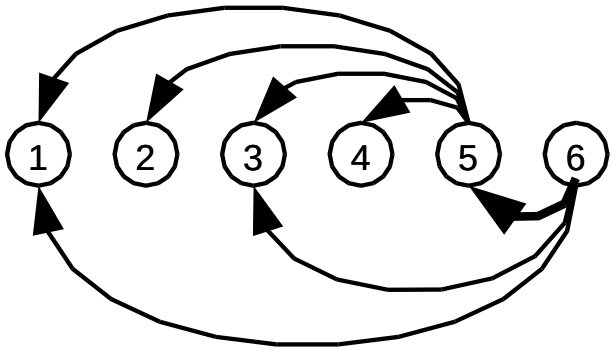

Figure 7: **Master tournament ambiguity.** Champions of generations one through six are depicted. An arrow pointing from champion x to champion y denotes that x is superior to y in a direct 288-way comparison. The master tournament reports that champion 5 defeats four other champions, while champion 6 only defeats three others. These results imply that 5 is superior to 6. However, when 5 and 6 are compared directly against each other, 6 actually wins. This way, master tournament results can be misleading. A method that captures the order of dominance among different strategies avoids such a confusion.

First we need a method for performing individual comparisons, i.e. whether one strategy is better than another. Because the board configurations can vary during the game, champion 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. Thus, the one who wins the majority of the 288 total games has demonstrated its superiority in many different scenarios, including those beginning with a disadvantage. 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 *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 counts how many strategies can be defeated by each generation champion, without identifying which ones. Thus, it can fail to detect cases where strategies that defeat fewer previous champions are actually superior (figure 7). 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.

The *dominance tournament* method of tracking progress in competitive coevolution meets this goal (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 over evolution. Then dominance is defined recursively starting from the first generation and progressing to the last:[1]

[1]Our definition of dominance is similar though not identical to the definition of a *transitive chain* (Rosin 1997, p.19).

- 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. We call d_n the nth dominant strategy of the run. A network c from a later generation that defeats d_4 but loses to d_3 has circular superiority, since it defeats newer strategies but loses to older ones. Since c is circular, it would not be entered into the dominance hierarchy. 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 tournament allows us to identify a sequence of increasingly more sophisticated strategies. They also require significantly fewer comparisons than the master tournament (Stanley and Miikkulainen 2002a).

Armed with the appropriate coevolution methodology and a measure of success, we can now ask the question: Does the complexification result in more successful competitive coevolution?

6 Results

Each of the 23 evolution runs lasted 500 generations, and took between 5 and 10 days on a 1GHz Pentium III processor, depending on the progress of evolution and sizes of the networks involved. The NEAT algorithm itself used less than 1% of this computation: most 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?

6.1 Evolution of Complexity

NEAT was run thirteen times, each time from a different seed, to verify that the results were consistent. The highest levels of dominance achieved were 17, 14, 17, 16, 16, 18, 19, 15, 17, 12, 12, 11, and 13, averaging at 15.15 (sd = 2.54).

At each generation where the dominance level increased in at least one of the thirteen runs, we averaged the number of connections and number of nodes in the current dominant strategy across all runs (figure 8). Thus, the graphs represent a total of 197 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 six. In a smooth trend over the first 200 generations, the number of connections in the dominant strategy grows by 50%. 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

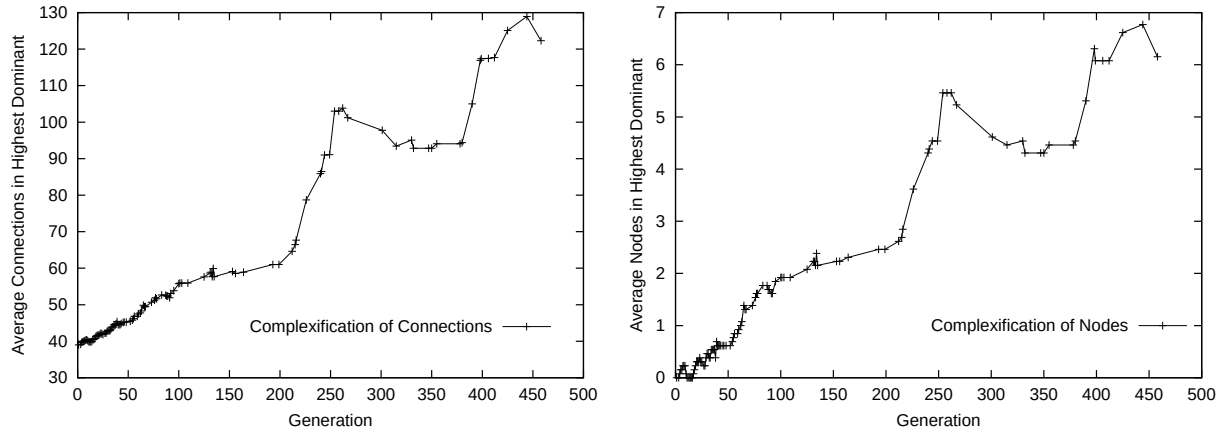

Figure 8: **Complexification of connections and nodes over generations.** The graphs depict the average number of connections and the average number of hidden nodes in the highest dominant network in each generation. Averages are taken over 13 complexifying runs. A hash mark appears every generation in which a new dominant strategy emerged in at least one of the 13 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 ($p < 0.001$).

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. Eventually, even more elaborate structures will emerge and again outperform the older structures.

Thus, there are two underlying forces of progress: the building of new structures, and the continual optimization of prior structures in the background. The product of these two trends is a gradual stepped progression towards increasing complexity.

Importantly, the dominant strategies did not have to complexify; Even though NEAT adds new structure to networks over generations, because of speciation many species remain simple even as some species become more complex. Thus, 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 increasing complexity.

6.2 Sophistication through Complexification

To see how complexification contributes to evolution, let us observe how a sample dominant strategy develops over time. While many complex networks evolved in the experiments, we follow the species that produced the winning network d_{17} in the third run because its progress is rather typical and easy to understand. 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 9a), 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

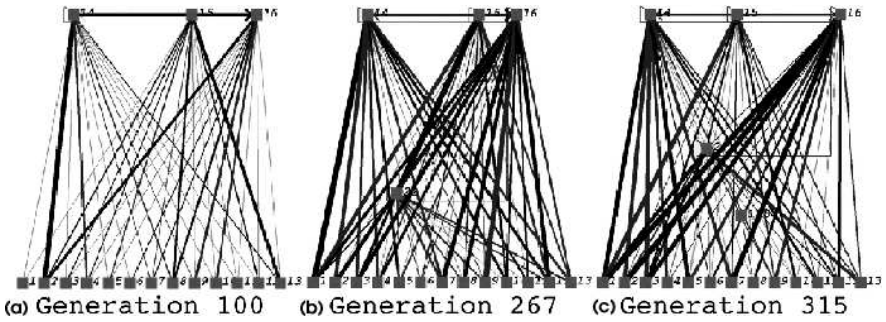

Figure 9: **Complexification of a Winning Species.** The best networks in the same species are depicted at landmark generations. Nodes are depicted as squares beside their node numbers, and line thickness represents the strength of connections. Over time, the networks became more complex and gained skills. (a) The champion from generation 10 had no hidden nodes. (b) The addition of h_{22} and its respective connections gave new abilities. (c) The appearance of h_{172} refined existing behaviors.

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.

- S_{200}. During the next 100 generations, S evolved a *resting* strategy, which it used when it had significantly lower energy than its opponent. In such a situation, the robot stopped moving, while the other robot wasted energy running around. By the time the opponent got 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 energy difference sensor, thereby slightly modifying intensity of an existing behavior.

- In S_{267} (figure 9b), a new hidden node, h_{22}, appeared. This node 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, allowing it to collect more 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 9c), another new hidden node, 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 increasing sophistication.

In order to illustrate the level of sophistication displayed during competition, we conclude this section with a description of an observed competition between two sophisticated strategies, S_{210} and S_{313}, from another run of evolution. At the beginning of the competition, S_{210} and S_{313} collected most of the available food until their energy levels were about equal. Two pieces of food remained on the board in locations distant

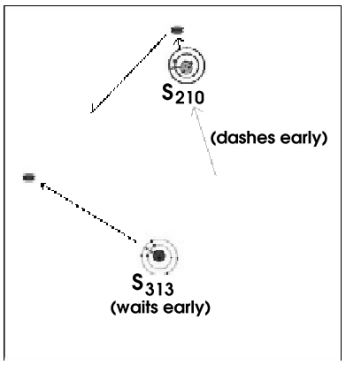

Figure 10: **Sophisticated Endgame.** Robot S_{313} dashes for the last piece of food while S_{210} is still collecting the second-to-last piece. Although it appeared that S_{313} would lose because S_{210} got the second-to-last piece, (gray arrow), it turns out that S_{210} ends with a disadvantage. It has no chance to get to the last piece of food before S_{313}, and S_{313} has been saving energy while S_{210} wasted energy traveling long distances. This way, sophisticated strategies evolve through complexification, combining multiple objectives, and utilizing weaknesses in the opponent's strategy.

from both S_{210} and S_{313} (figure 10). Because of the danger of colliding with similar energy levels, the naive strategy would be to rush for the last two pieces of food. In fact, S_{210} did exactly that. However, S_{313} cleverly forfeit this race for the second-to-last piece, opting to sit still, even though its energy temporarily dropped below S_{210}'s. After S_{210} consumed the second-to-last piece, it headed towards the last piece, which was now far away. Even though S_{313} had less energy, it was closer and got there first. It received a boost of energy while S_{210} wasted its energy running the long distance from its current position. Thus, S_{313}'s strategy ensured that it had more energy when they finally met. Robot S_{313}'s behavior was surprisingly deceptive, showing that a high level of strategic sophistication had evolved.

This analysis of individual evolved behaviors shows that complexification indeed elaborates on existing strategies, and allows highly sophisticated behaviors to develop that balance multiple goals and utilize weaknesses in the opponent. The last question is whether complexification indeed is necessary to achieve these behaviors.

6.3 Complexification vs. Fixed-topology Evolution

To see whether complexifying coevolution is more powerful than standard coevolution in a fixed search space, we compared the two methods. Note that it is not possible to compare them using the standard crossvalidation methodology because no external performance measure exists in this domain. However, the evolved neural networks can be compared *directly* by playing a duel. Thus, a run of fixed-topology coevolution can be compared to a run of complexifying coevolution by playing the highest dominant strategy from the fixed-topology run against the entire dominance ranking of the complexifying run. The highest level strategy in the ranking that the fixed-topology strategy can defeat is a measure of its quality against complexifying coevolution.

By playing every fixed-topology champion against the dominance ranking from every complexifying run, we can measure the performance of non-complexifying evolution. In addition, we established a baseline for comparison by also playing the champions of the *complexifying* runs against the entire dominance ranking of every other complexifying run. The average levels reached by complexifying and fixed-topology coevolution can then be compared.

Accordingly, the performance of a particular run is the average highest dominance level of the strategies it can defeat among all complexifying runs, normalized by the total number of dominance levels for that

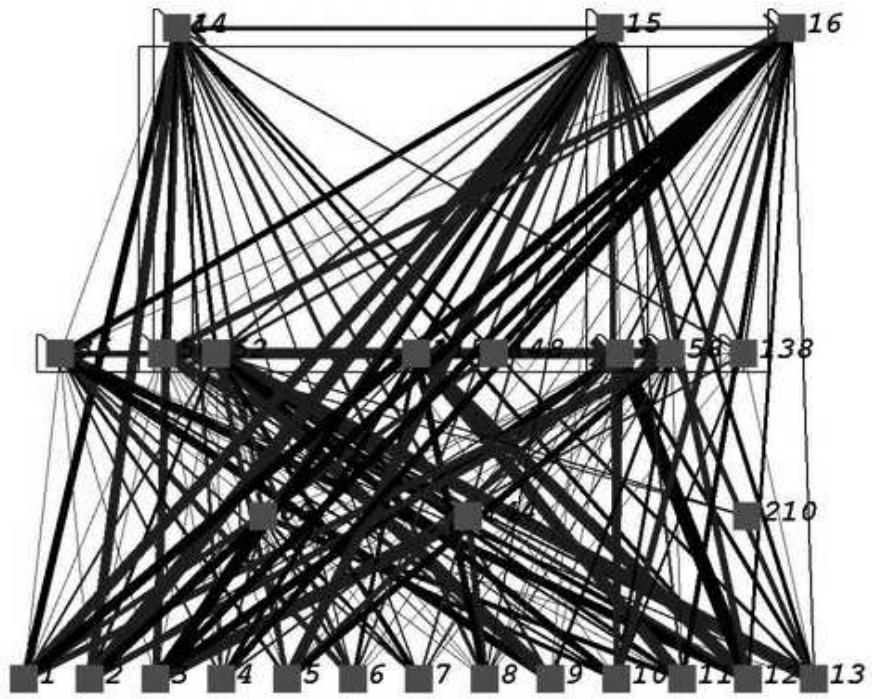

Figure 11: **The Best Complexifying Network.** The highest performing network from complexifying coevolution is depicted. The network has 11 hidden units and 202 connections, plotted as in figure 9. This network makes significant use of structure. While it still has basic direct connections, its strategy has been elaborated through the addition of new nodes and connections. Lateral and recurrent connections are used to make more refined decisions based on past experiences. While complexification makes good use of this high-dimensional space, it is difficult for fixed-topology evolution to take advantage of similarly complex structures.

run. For example, if a strategy can defeat up to and including the 13th dominant strategy out of 15, then its performance against that run is $\frac{13}{15} = 86.7\%$. The performance against all complexifying runs can then be averaged to obtain the overall performance of that run.

First, we establish the baseline performance by playing complexifying coevolution runs against themselves. We will then compare it with fixed-topology coevolution of both small and large networks.

6.3.1 Complexifying Coevolution

The highest dominant strategy from each of the 13 complexifying runs played the entire dominance ranking from every other run. Their average performances were 87.9%, 83.8%, 91.9%, 79.4%, 91.9%, 99.6%, 99.4%, 99.5%, 81.8%, 96.2%, 90.6%, 96.9%, and 89.3%. The fifth, sixth, and seventh runs thus were able to defeat almost the entire dominance ranking of every other run, indicating that these were the best of the complexifying runs. The highest dominant network from the best run (99.6%) is shown in figure 11. The average performance over all complexifying runs was 91.4% (sd=12.8%).

This result makes sense because it indicates that complexifying runs produce consistently good strategies. On average, the highest dominant strategies qualify for the top 10% of the other complexifying runs.

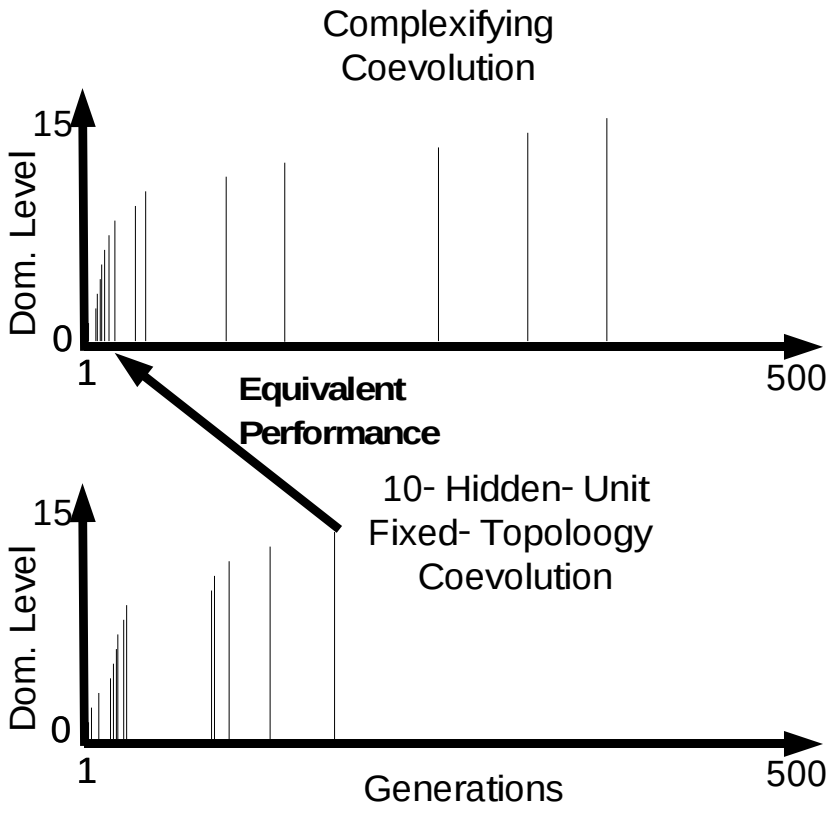

Figure 12: **Comparing Typical Runs of Complexifying Coevolution and 10-hidden-unit Fixed-Topology Coevolution.** Dominance levels are charted on the y-axis and generations on the x-axis. A line appears at every generation where a new dominant strategy arose in each run. The height of the line represents the level of dominance. The arrow shows that the highest dominant strategy found in 10-hidden-unit fixed-topology evolution only performs as well as the 8th dominant strategy in the complexifying run, which was found in the 19th generation! (Average = 24, sd = 8.8) In other words, only a few generations of complexifying coevolution are as effective as several hundred of fixed-topology evolution.

6.3.2 10-Hidden-Unit Fixed-Topology Coevolution

Fixed-topology coevolution uses fully-connected networks with a single hidden layer, and the number of hidden units must be chosen by the experimenter. Thus, in order to make the comparison between fixed-topology and complexifying coevolution fair, a reasonable size must be chosen for networks in the fixed-topology runs. One sensible approach is to choose a complexity that approximates the complexity of the highest dominant strategy from the highest performing complexifying run (figure 11). This strategy utilized 11 hidden units and 202 connections including both recurrent connections and direct connections from input to output. In order to approximate this complexity, we chose to use 10 hidden unit fully recurrent networks with direct connections from inputs to outputs, with a total of 263 connections. A network of this type should be able to approximate the functionality of the most effective complexifying strategy.

Three runs of fixed-topology coevolution were performed with these networks, and their highest dominant strategies were compared to the entire dominance ranking of every complexifying run. Their average performances were 29.1%, 34.4%, and 57.8%, for an overall average of 40.4%. Compared to the 91.4% performance of complexifying coevolution, it is clear that fixed-topology coevolution produced consistently much inferior solutions. As a matter of fact, no fixed-topology run could defeat any of the 13 highest

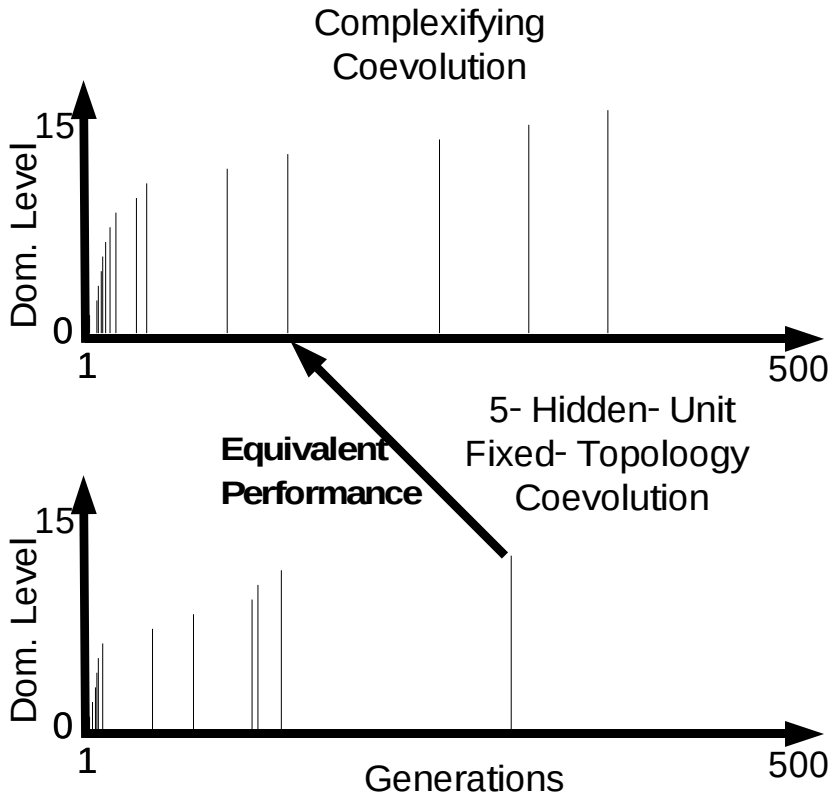

Figure 13: **Comparing Typical Runs of Complexifying Coevolution and 5-hidden-unit Fixed-Topology Coevolution.** As in figure 12 dominance levels are charted on the y-axis and generations on the x-axis, a line appears every generation where a new dominant strategy arose in each run, and the height of the line represents the level of dominance. The arrow shows that the highest dominant strategy found in 5-hidden-unit fixed-topology evolution only performs as well as the 12th dominant in the complexifying run, which was found in the 140th generation. (Average 159, sd = 72.0) Thus, even in the best configuration, fixed-topology evolution takes about twice as long to achieve the same level of performance.

dominant strategies from complexifying coevolution.

This difference in performance can be illustrated by computing the *average generation* of complexifying coevolution with the same performance as fixed-topology coevolution. This generation turns out to be 24 (sd = 8.8). In other words, 500 generations of fixed-topology coevolution reach on average the same level of dominance as only 24 generations of complexifying coevolution! In effect, the progress of the entire fixed-topology coevolution run is compressed into the first few generations of complexifying coevolution (figure 12).

6.3.3 5-Hidden-Unit Fixed-Topology Coevolution

One of the arguments for using complexifying coevolution is that starting the search directly in the space of the final solution may be intractable. This may explain why the attempt to evolve fixed-topology solutions at a high level of complexity failed. Thus, in the next experiment we chose instead to evolve fully-connected, fully-recurrent networks with five hidden nodes. After considerable experimentation, we found out that this level of complexity was the most productive for fixed-topology evolution, allowing it to find the best

Coevolution Type	Ave. Highest Dom. Level	Ave. Highest Generation	Average Performance	Equivalent Generation (out of 500)
Complexifying	15.2	353.6	91.4%	343
Fixed-Topology 10 Hidden Node	12.0	172	40.4%	24
Fixed-Topoogy 5 Hidden Node	13.0	291.4	80.3%	159

Table 1: **A comparison of fixed-topology coevolution with complexifying coevolution.** The second column shows how many levels of dominance were achieved in each type of coevolution on average. The third shows the average last generation where dominant strategies appeared, indicating how long innovation generally continues. The fourth column gives the highest dominance level in complexifying runs that the dominant strategy from the fixed-topology run could defeat and the fifth column shows its average generation. The differences in dominance level ($p < 0.05$), performance ($p < 0.001$) , and equivalent generation ($p < 0.001$) are all significant. The main result is that the level of sophistication reached by complexifying coevolution is significantly higher than that reached by fixed-topology coevolution.

solutions it can. These networks utilized 144 connections, constituting a significantly lower-dimensional search space than the 10-hidden-unit networks. It also is about the *average* number of hidden nodes used in the highest dominant strategy of complexifying runs.

A total of seven runs were performed with the 5-hidden-node networks, with average performances of 70.7%, 85.5%, 66.1%, 87.3%, 80.8%, 88.8%, and 83.1%. The overall average was 80.3% (sd=18.4%), which is significantly below the 91.4% performance of complexifying coevolution ($p < 0.001$).

The results with 5-hidden-unit fixed-topology coevolution are significantly better than with 10 hidden units, confirming the hypothesis that search is more difficult in higher-dimensional space. However, the solutions found this way were still inferior to the complexified solutions. The two most effective complexifying runs were never defeated by any of the fixed-topology runs. Also, because each dominance level is more difficult to achieve than the last, on average the fixed-topology evolution only reached the performance of the 159th complexifying generation (sd=72.0). Thus, even in the best case, fixed-topology coevolution on average only finds the level of sophistication that complexifying coevolution finds halfway through a run (figure 13).

6.3.4 Comparing Fixed-Topology vs. Complexifying Coevolution

Table 1 shows how the different methods of coevolution differ on the number of dominance levels, the generation of the highest dominance level, overall performance, and equivalent generation. The conclusion is that complexifying coevolution innovates longer and finds a higher level of sophistication both on average and absolutely.

It is important to ask whether standard non-complexifying coevolution could in principle reach the same level of sophistication as the most effective complexifying runs. As the results show, if fixed-topology coevolution is forced to search directly in the space of the most effective solutions, it cannot reach even 50% of the performance of complexifying coevolution. On the other hand, if it is allowed to optimize less complex networks, the most sophisticated solutions may not exist in that space. Thus, it is unclear which, if any, choice of topology is correct with fixed-topology evolution. In contrast, complexifying coevolution finds the appropriate level of complexity for the task, allowing it to discover more sophisticated strategies.

7 Discussion and Future Work

The results confirm that complexification makes it possible to find more 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 add functionality without sacrificing some of the knowledge that is already present. In contrast, if new genetic material can be added, sophisticated elaborations can be layered above existing structure. Indeed, dominant strategies are usually more complex than their predecessors, showing that evolution capitalizes on the capacity to complexify.

Adding new genes to the genome does more than expand the dimensionality of the search space. Before the addition, the values of the existing genes have already been optimized over preceding generations. Thus, after a new gene is added, the genome is *already* in a promising part of the new, higher-dimensional space. Thus, the search in the higher-dimensional space is not starting blindly as it would if evolution began searching in that space. It is for this reason that complexification can find high-dimensional solutions that would otherwise be difficult to discover.

Complexification can find solutions that would be difficult to obtain through evolving fixed structure: No fixed-sized network was able to defeat the best strategies from complexifying coevolution. 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. Moreover, *even if* complexity is guessed right, searching in the the high-dimensional space of the final solution may be intractable because search begins in a random part of the space. A complexifying system saves the user from such concerns.

In principle, if the complexification process is efficient enough, it will generate just enough structure to solve the task. However, if complexification is too greedy or inaccurate, it may be useful to counterbalance it with a process of simplification, in which genes are occasionally disabled or deleted from the genome. Such a process could be useful in principle to keep the evolved structures minimal. However, removing structure is potentially more disruptive than adding it, since entire modules can be accidentally cut off with even a small deletion. Moreover, removing a gene may initially improve fitness, even though the gene may be necessary to construct the final solution. In such a case, evolution may become stuck for several generations waiting for the gene to become reinstated. As a result, in preliminary experiments we did not find minimization to improve the performance of complexifying system. Whether these two processes can implemented so that they leverage each other is an interesting direction of future work.

The advantages of complexification do not imply that fixed-sized genomes cannot sometimes evolve increasingly complex *phenotypic behavior*. Recall that complexification refers to the addition of new structure to the solution (Section 2.1). Depending on the mapping between the genotype and the phenotype, it may be possible for a fixed, finite set of genes to represent solutions (phenotypes) with varying behavioral complexity. For example, such behaviors have been observed in Cellular Automata (CA). A cellular automaton is a lattice of cells that change their state as a function of their own current state and the state of other cells in their neighborhood. This neighborhood function can be evolved, in which case the size of the genome is 2^{n+1} (assuming n neighboring cells with binary state). Mitchell et al. (1996) were able to evolve such fixed-sized CA neighborhood functions to determine whether black or white cells were in the majority. The evolved CAs displayed complex global behavior patterns that converged on a single classification, depending on which cell type was in the majority. Over the course of evolution, the behavioral complexity of the CA rose even as the genome remained the same size.

In the CA example, the correct neighborhood size was chosen a priori. This choice is difficult to make, and highly critical in order to succeed. If the desired behavior had not existed within the chosen size, even if

the behavior would become gradually more complex, the system would never solve the task. Interestingly, such a dead-end could be avoided if the neighborhood (i.e. the genome) could be expanded during evolution. It is possible that CA could be more effectively evolved by complexifying (i.e. expanding) the genomes, and speciating to protect innovation, as in NEAT.

Moreover, not only can the chosen genome be too small to represent the solution, but it can also be unnecessarily large. Searching in a space of more dimensions than necessary can impede progress. Thus, for any particular genome size, the question can always be asked: does a solution possibly exist in the space of a smaller genome? For this reason, it makes sense to start evolution in a minimal space and complexify. In the example of the CA, if the desired function existed in a smaller neighborhood it could have been found with significantly fewer evaluations. Indeed, it is even possible that the most efficient neighborhood is not symmetric, or contains cells that are not directly adjacent to the cell being processed. Moreover, even the most efficient neighborhood may be too large a space in which to begin searching. Starting search in a small space and incrementing into a promising part of higher-dimensional space is more likely to find a solution. For these reasons, complexification can be an advantage, *even if* behavioral complexity can increase to some extent within a fixed space.

The preceding example raises the intriguing possibility that *any* structured phenotype can be evolved through complexification from a minimal starting point, historical markings, and the protection of innovation through speciation. In addition to neural networks and CA, electrical circuits (Miller et al. 200a,b), genetic programs (Koza 1992), robot body morphologies (Lipson and Pollack 2000), Bayesian networks (Mengshoel 1999), finite automata (Brave 1996), and building and vehicle architectures (O'Reilly 2000) are all structures of varying complexity that can benefit from complexification. By starting search in a minimal space and adding new dimensions as necessary, highly complex phenotypes can be discovered that would be difficult to find if search began in the intractable space of the final solution, or if it was prematurely restricted to too small a space.

Note that not all algorithms that grow structure are complexifying systems. In many cases, the goal of the growth is to form gradually better *approximations*. For example, methods like Incremental Grid Growing (Blackmore and Miikkulainen 1995), and Growing Neural Gas (Fritzke 1995) add neurons to a network until it approximates the topology of the input space reasonably well. On the other hand, complexifying systems do not have to be non-deterministic (like NEAT), nor do they need to be based on evolutionary algorithms. For example, Harvey (1993) introduced a deterministic algorithm where the chromosome lengths of the entire population increase all at the same time in order to expand the search space; Fahlman and Lebiere (1990) developed a supervised (non-evolutionary) neural network training method called cascade correlation, where new hidden neurons are added to the network in a predetermined manner in order to complexify the function it computes.

The conclusion is that complexification is an important general principle in Artificial Intelligence (AI). One of the primary and most elusive goals of the field is to create systems that *scale up*. In a sense, complexification *is* the process of scaling up. It is the general principle of taking a simple idea and elaborating it for broader application. Much of AI is concerned with search, whether over complex multi-dimensional landscapes, or through highly-branching trees of possibilities. However, intelligence is as much about deciding *what space* to search as it is about searching once the proper space has already been identified. For many problems, the hard question is not how to search, but rather, what should we be searching in? Currently, only humans are able to decide the proper level of abstraction for solving many problems. A program that can decide what level of abstraction is most appropriate for a given domain would be a highly compelling demonstration of Artificial Intelligence.

8 Conclusion

The experiments presented in this paper show that complexification of genomes leads to continual coevolution of increasingly sophisticated strategies. Three trends were found in the experiments: (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 a successful search for complex solutions. By both complexifying and optimizing, a system can both discover the appropriate representation for the solution as well as the location of the solution itself.

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.

A NEAT System Parameters

Each population had 256 NEAT networks, for a total of 512. The coefficients for measuring compatibility were $c_1 = 1.0$, $c_2 = 1.0$, and $c_3 = 2.0$. The initial compatibility distance was $\delta_t = 3.0$. However, because population dynamics can be unpredictable over hundreds of generations, we assigned a target of 10 species. If the number of species grew above 10, δ_t was increased by 0.3 to reduce the number of species. Conversely, if the number of species fell below 10, δ_t was decreased by 0.3 to increase the number of species. In order to prevent stagnation, the lowest performing species over 30 generations old was not allowed to reproduce. The champion of each species with more than five networks was copied into the next generation unchanged. There was an 80% chance of a genome having its connection weights 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 is tolerant to frequent mutations because of the protection speciation provides.) There was a 75% chance that an inherited gene was disabled if it was disabled in either parent. In each generation, 25% of offspring resulted from mutation without crossover. The interspecies mating rate was 0.05. The probability of adding a new node was 0.01 and the probability of a new link mutation was 0.1. We used a modified sigmoidal transfer function, $\varphi(x) = \frac{1}{1+e^{-4.9x}}$, at all nodes. These parameter values were found experimentally: links need to be added significantly more often than nodes, and an average weight difference of 0.5 is about as significant as one disjoint or excess gene. Performance is robust to moderate variations in these values: The dynamic compatibility distance measure caused speciation to remain stable.

References

Amores, A., Force, A., Yan, Y.-L., Joly, L., Amemiya, C., Fritz, A., Ho, R. K., Langeland, J., Prince, V., Wang, Y.-L., Westerfield, M., Ekker, M., and Postlethwait, J. H. (1998). Zebrafish HOX clusters and vertebrate genome evolution. *Science*, 282:1711–1784.

Angeline, P. J., and Pollack, J. B. (1993). Competitive environments evolve better solutions for complex tasks. In Forrest, S., editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, 264–270. San Francisco, CA: Morgan Kaufmann.

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.

Blackmore, J., and Miikkulainen, R. (1995). Visualizing high-dimensional structure with the incremental grid growing neural network. In Prieditis, A., and Russell, S., editors, *Machine Learning: Proceedings of the 12th Annual Conference*, 55–63. San Francisco, CA: Morgan Kaufmann.

Brave, S. (1996). Evolving deterministic finite automata using cellular encoding. In Koza, J. R., Goldberg, D. E., Fogel, D. B., and Riolo, R. L., editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, 39–44. Stanford University, CA, USA: MIT Press.

Carroll, S. B. (1995). Homeotic genes and the evolution of arthropods and chordates. *Nature*, 376:479–485.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314.

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.

Fahlman, S. E., and Lebiere, C. (1990). The cascade-correlation learning architecture. In Touretzky, D. S., editor, *Advances in Neural Information Processing Systems 2*, 524–532. San Francisco, CA: Morgan Kaufmann.

Ficici, S. G., and Pollack, J. B. (2001). Pareto optimality in coevolutionary learning. In Kelemen, J., editor, *Sixth European Conference on Artificial Life*. Berlin; New York: Springer-Verlag.

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

Force, A., Lynch, M., Pickett, F. B., Amores, A., lin Yan, Y., and Postlethwait, J. (1999). Preservation of duplicate genes by complementary, degenerative mutations. *Genetics*, 151:1531–1545.

Fritzke, B. (1995). A growing neural gas network learns topologies. In G.Tesauro, D.S.Touretzky, and T.K.Leen, editors, *Advances in Neural Information Processing Systems 7*, 625–632. Cambridge, MA: MIT Press.

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.

Harvey, I. (1993). *The Artificial Evolution of Adaptive Behavior*. PhD thesis, School of Cognitive and Computing Sciences, University of Sussex, Sussex.

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.

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

Lipson, H., and Pollack, J. B. (2000). Automatic design and manufacture of robotic lifeforms. *Nature*, 406:974–978.

Maley, C. C. (1999). Four steps toward open-ended evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-1999)*, 1336–1343. San Francisco, CA: Morgan Kaufmann.

Martin, A. P. (1999). Increasing genomic complexity by gene duplication and the origin of vertebrates. *The American Naturalist*, 154(2):111–128.

Mengshoel, O. J. (1999). *Efficient Bayesian Network Inference: Genetic Algorithms, Stochastic Local Search, and Abstraction*. PhD thesis, University of Illinois at Urbana-Champaign Computer Science Department, Urbana-Champaign, IL.

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.

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.

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.

Noble, J., and Watson, R. A. (2001). Pareto coevolution: Using performance against coevolved opponents in a game as dimensions for parerto selection. In et al, L. S., editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*. San Francisco, CA: Morgan Kaufmann.

O'Reilly, U.-M. (2000). Emergent design: Artificial life for architecture design. In *7th International Conference on Artificial Life (ALIFE-00)*. Cambridge, MA: MIT Press.

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

Radding, C. M. (1982). Homologous pairing and strand exchange in genetic recombination. *Annual Review of Genetics*, 16:405–437.

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. (1997). *Coevolutionary Search Among Adversaries*. PhD thesis, University of California, San Diego, San Diego, CA.

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

Sigal, N., and Alberts, B. (1972). Genetic recombination: The nature of a crossed strand-exchange between two homologous DNA molecules. *Journal of Molecular Biology*, 71(3):789–793.

Spears, W. (1995). Speciation using tag bits. In *Handbook of Evolutionary Computation*. IOP Publishing Ltd. and Oxford University Press.

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.

Stanley, K. O., and Miikkulainen, R. (2002d). Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127.

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.