Neuro-Evolution Through Augmenting Topologies Applied To Evolving Neural Networks To Play Othello

Timothy Andersen, Kenneth O. Stanley, and Risto Miikkulainen
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712 USA
`t.andersen@mai.utexas.edu`
`kstanley@cs.utexas.edu`
`risto@cs.utexas.edu`

Abstract

Many different approaches to game playing have been suggested including alpha-beta search, temporal difference learning, genetic algorithms, and coevolution. Here, a powerful new algorithm for neuroevolution, Neuro-Evolution for Augmenting Topologies (NEAT), is adapted to the game playing domain. Evolution and coevolution were used to try and develop neural networks capable of defeating an alpha-beta search Othello player. While standard evolution outperformed coevolution in experiments, NEAT did develop an advanced mobility strategy. Also we demonstrated the need for protection of long-term strategies in coevolution. NEAT established its potential to enter the game playing arena and illustrated the necessity of the mobility strategy in defeating a powerful positional player in Othello.

1 Introduction

Game playing is studied extensively in artificial intelligence because of its applications to many real world problems in economics, politics, biology, and countless other areas. Also, games are ideal for AI because they have well defined rules that make them easier to simulated on a computer. Studying how humans and machines play games provides insight into how humans and machines solve a larger class of important problems.

Most game playing algorithms make use of the well studied concept of search and tree pruning. These algorithms attempt to search through a large number of game scenarios, ignoring the unlikely ones, and choose the best possible path based on the evaluation of each different sequence of moves. Deepening the search has the positive effect of increasing performance but the drawback of exponentially increasing search time, requiring faster machines.

Unlike search algorithms, human experts do not scan any more moves than novices (DeGroot 1965). Instead, chess masters have been found to rely on advanced pattern recognition to eliminate all but the best moves from consideration (Frey and Adesman 1976; Charness 1976). Therefore, in trying to understand human problem solving, a more *human-like* approach is needed. Artificial Neural Networks represent such an approach because they must learn the game without prior knowledge of the best strategies or how to evaluate individual moves. The only positive or negative feedback the neural networks presented received was whether they had won or lost the game. This means that strategies could not be biased towards any particular feature of the game.

Traditionally neural networks have learned to play games such as Go using training algorithms such as temporal difference learning (Schraudolph et al. 1994). However, evolutionary algorithms have been demonstrated to be a more powerful tool for evolving innovative networks with advanced strategies (Moriarty and Miikkulainen 1995). Recently, Stanley and Miikkulainen (2002) have developed a new and extremely powerful evolutionary algorithm called Neuro-Evolution Through Augmenting Topologies (NEAT).

NEAT has several advantages over traditional evolutionary algorithms. It uses speciation to protect innovative networks from being eliminated before they develop to their full potentials. Also, NEAT grows its networks from a minimal topology, adding nodes and links as mutations, to obtain the most efficient networks possible. While the power of this new algorithm had been demonstrated in continuous control tasks such as non-Markov double pole-balancing, it had yet to be tried in the game playing domain (Stanley and Miikkulainen 2002).

Two different approaches were employed to test NEAT in the game playing domain. In each, networks were required to learn to play the game Othello without any prior understanding of the game.

In the first approach, networks were evolved to play against a random mover, during which they developed a *positional strategy* often used by novices. They were then evolved against an α-β search program where they learned a sophisticated *mobility* strategy similar to that seen in tournaments and used by all master Othello players. This phenomenon was first demonstrated in Moriarty and Miikkulainen (1995), where an evolutionary algorithm known as *Marker-based Encoding*, proposed by Fullmer and Miikkulainen (1992), was used to evolve networks to the mobility level of playing proficiency. The mobility strategy is considered to be very difficult to learn and requires a great deal of practice for humans to master it (Billman and Shaman 1990).

In the second approach, a technique known as competitive coevolution was used. This approach eliminates the need for a fixed opponent such as a search program in favor of pitting two population against one another. While Lubberts and Miikkulainen (2001) had shown promising results from this approach in playing Go, our results were limited.

In the first section we review the game of Othello for those non familiar. Then we discuss the details of NEAT and its advantages, as well as competitive coevolution. In the next section we present our experimental setup and our control experiment which involved eliminating NEAT's two distinctive features (speciation and topology building) and evolving against the random and alpha-beta player. Following that we present the results of our experiments. The significance of our results are discussed in Section 6. Section 7 details a plan for future work in this on-going project.

2 The Game Othello

2.1 History and Previous Work

Othello was first formalized by Goro Hasegawa in Japan in 1974, but has existed since the nineteenth century under the name Reversi. It is a derivative of the the Go family of games, emphasizing the capture of territory as a means of advancement. In Japan it is second only to Go in popularity. It has a few simple rules making it easily accessible to novices but to play it well requires players to develop complex strategies over a long period of time. For more information on Othello see Staff (1980).

One of the first master level Othello playing programs, Iago, was developed in the 1980's by Rosenbloom (1982). It used alpha-beta search with kill tables. Later, Lee and Mahajan (1990) developed a more powerful alpha-beta search based Othello player called Bill. Bill used Bayesian learning to optimize its evaluation function making it one of the strongest machine Othello players of its time.

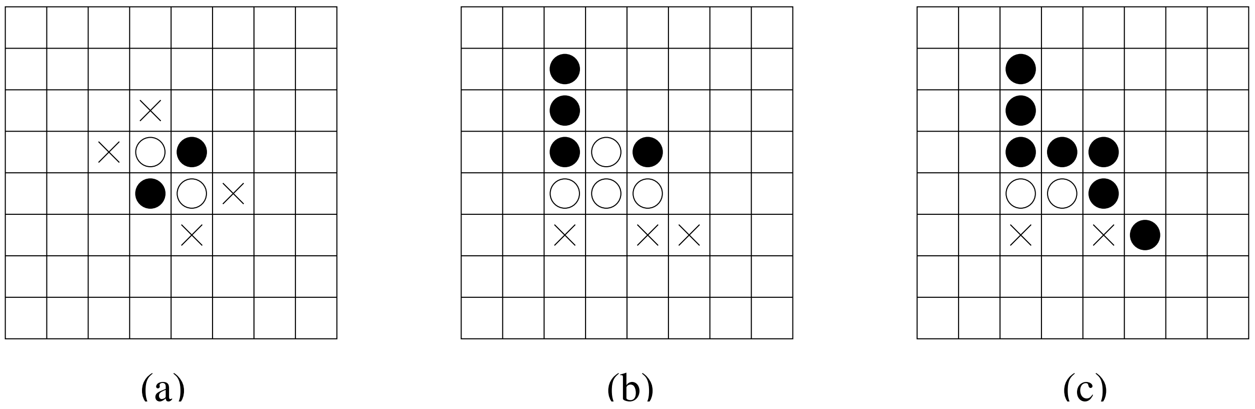

Figure 1: **Othello Board Configurations.** (a) Initial board. (b) After four moves (black's legal moves marked with X's) (c) After black has moved to the rightmost X.

2.2 Setup and Rules of Othello

Othello is a two player game played on an 8x8 board. One player plays white, the other black. All pieces are identical discs, with one side white and the other black. The initial board setup appears in Figure 1(a).

A player can only place a piece in an empty space such that the new piece and another of the player's pieces *brackets* one or more of the opponent's pieces horizontally, vertically, or diagonally. All the bracketed pieces are flipped over and become the color of the current player. Players take turns putting down pieces until the board is full or one of the players has had to pass twice in a row because no legal moves were available. The player with the most number of pieces at the end wins. Figure 1(b) shows the legal moves available to black. Figure 1(c) shows the result when black moves into the sixth row of the sixth column.

2.3 Strategies

The game of Othello can be divided into three phases, openning game, mid-game, and end game. There are no well defined boundaries between these phases, but they all require markedly different strategies. The openning game can be played using a book of favorable move sequences. In the end game, players simply try and gather as many pieces as possible so as to win the game. The mid-game, however, has attracted much attention over the years because the strategies required to play the mid-game are complex and difficult to program into a computer.

Two general classes of mid-game strategies exist in Othello, the positional strategy and the mobility stategy. The positional strategy is simpler than the mobility strategy but inferior. A player using the positional strategy has the immediate goal of capturing as many pieces as possible. To do this the player attempts to play the edge of the board so as to ring the opponent in. A positional player always captures the four corner spaces if possible because a piece in one of these spots cannot be captured. Against another positional player, this strategy will precipitate an arms race between the two players, each attempting to get the upper hand. This class of strategies is easy to program and easy for beginners to master.

A more difficult and superior strategy is known as the mobility strategy. In this strategy the player attempts to control the center of the board, forcing the opponent to surround the player's pieces. Mobility is based on the idea that to win, a player must force the opponent to give up available moves until the player is in a position to decide exactly where the opponent will have to move. The opponent will be forced to surrender corners and edges in the end game because of what the player does in the mid-game. The main features of the mobility strategy in the mid-game are a low piece count and a large number of available moves for the player. For the opponent, the opposite should be true.

Billman and Shaman (1990) have shown that mobility is a very difficult strategy to learn. This strategy is believed to have been developed only in Japan by one person or group of persons and was subsequently spread to Europe and the United States (Billman and Shaman 1990). If machines using evolutionary algorithms are able to discover this strategy more than once, that would imply that machines have the potential to serve as a tremendous resource for developing new approaches to problem solving and that the result in Moriarty and Miikkulainen (1995) is not an isolated phenomenon.

3 Method

3.1 Neuro-Evolution Through Augmenting Topologies

Neuro-Evolution Through Augmenting Topologies (NEAT) represents a novel collusion of a number of different ideas in evolutionary algorithms that have until recently remained separate. Its genetic encoding allows genomes to be lined up easily during mating, serving to eliminate non-viable offspring. *Innovation numbers* keep track of matching genomes by recording structural mutations so that networks with the same structural ancestry but different weights can be mated.

Mutation in NEAT not only alters connection weights but can add new nodes and links to the genome, effectively growing the topology of the network rather than simply changing the weights of a fixed topology. This aspect gives NEAT a performance gain over other neuro-evolutionary algorithms because small structures optimize more quickly than large ones. To take full advantage of this principle, NEAT uses a starting genome with minimal structure so as to allow the evolution to proceed without bias towards any conceivable topology.

While Stanley and Miikkulainen (2002) made use of a topology with no hidden nodes as a starting point in the small topologies with which they experimented, in larger networks a genome with a small number of hidden nodes is, in fact, more minimal than one with no hidden nodes because it requires fewer links.

Speciation is another important feature of NEAT. Speciation can have dire consequences with noisy evaluation functions as will be discussed below under *Noise Sensitivity in NEAT*. However, for topological innovation to occur, speciation is absolutely necessary. Because small structures optimize more quickly than larger ones, it is easy to see how structure could never evolve in a system like NEAT unless these topological innovations were protected by only mating with their own kind. In each generation NEAT partitions the population into species based on how the organisms differ structurally (i.e. whether they have an *excess* of links or nodes or *disjoint* links or nodes) and based on their link weight differences. This partitioning assures that only structurally similar organisms mate and allows topologies to be thoroughly investigated before they are discarded (Stanley and Miikkulainen 2002).

In continuous control tasks, such as pole-balancing and double pole-balancing with and without velocity information, NEAT was found to achieve proficiency remarkably faster than other evolutionary algorithms (Stanley and Miikkulainen 2002). We hypothesized that NEAT could bring its formidable arsenal to bear on game playing as well. However, as illustrated in the Results section, several factors posed challenges to overcome in adapting NEAT to play Othello.

3.2 Competitive Coevolution

In most experiments involving neuro-evolution an opponent based on alpha-beta search or some other algorithm is necessary for the population to evolve against. The opponent provides an environment of conflict in which only the fittest survive. Competitive coevolution attempts to do away with this traditional model by allowing two populations, called the *hosts* and the *parasites*, to compete in an arms race for superiority. Rosin and Belew (1997) demonstrated the feasibility of competitive coevolution in the game playing domain using three techniques: shared sampling, hall of fame, and competitive fitness sharing. We use a different form of sampling, neither shared nor random, and competitive fitness sharing is not used because NEAT already has a form of fitness sharing within its species. However, we do use the Hall of Fame concept.

Since it would be extremely inefficient for every host to play against every parasite, we only choose a sample of parasites for the hosts to play against. This concept is called sampling. In our form of sampling we simple count off 4 species from the top of the parasite population's species list, and have the current host play against the champion networks from each of these species. The results of these games are then added to the host's total fitness measurement. This technique has the advantage of allowing the hosts to compete against different strategies, because each species, in a sense, represents a different strategy. Only evaluating against the best of the parasites, forces the arms race between the elite members of each population.

Populations as small as ours (only about 150 organims), have a very short memory (Rosin and Belew 1997). Old gains are quickly forgotten causing the arms race to move in a circular pattern where strategies are forgotten as others take their place, and then they are rediscovered later. To prevent this Rosin and Belew (1997) propose an approach called *Hall of Fame* where the best networks are saved at the end of every generation. By saving these networks, we save the strategies of older generations. During a host organism's evaluation, the host is tested against a sample of size 8 from the hall of fame. To ensure that the sampling is even, the hall of fame list is uniformly partitioned and one organism is chosen at random from each partition. The scores from these evaluations are also added to the host's total fitness.

Through the use of sampling and hall of fame, we ensure an arms race is precipitated between two populations. However, for coevolution to progress, new strategies must develop quickly. There is no opportunity for networks to sit outside the heat of conflict for long and optimize a strategy to the point where it can defeat a weaker, but more quickly developed strategy. This lack of protection for burgeoning genomes means that competitive coevolution can miss important avenues of advancement, favoring short-term gains over long-term progression. This important idea is elaborated in the Discussion Section.

4 Experiments

4.1 Game Playing Neural Networks

Each network received as input the entire board configuration using the scheme proposed in Moriarty and Miikkulainen (1995). The board was condensed to a vector of 128 floating-point numbers. Each of the 64 board spaces corresponded to a pair of numbers in the vector. Empty spaces were encoded as two zeros, player spaces as a one and a zero, and enemy spaces as a zero and a one. Nothing corresponded to two ones. No other information was given to the networks. So, rather than searching through possible move sequences like a search algorithm, the neural networks rely on pattern matching to convert the board configuration into a specific move. The information on how to play comes from repeated exposure to different game scenarios as the networks evolve. This learning is similar to how a human novice becomes a master by playing the game over and over again.

Each network has 64 outputs, one for each board square. After the network was fully activated, the output with the highest activation corresponding to a legal move became the network's move. Networks were not required to distinguish between legal and non-legal moves.

Networks are activated incrementally in NEAT. In a continuous control task where time is counted in seconds rather than turns, NEAT would simply activate each layer once a second. Inputs from previous timesteps would propagate through the network over several activations. This scheme gives the network a kind of memory because it can consider inputs from several timesteps at once by evolving topologies that cause inputs to reach outputs at different times. Although it makes sense to propagate inputs over time in a real-time task, in a turn based game such as Othello, inputs from previous turns are irrelevant to the decision of where to move. Therefore, the network was fully activated each turn. In Stanley and Miikkulainen (2002), the XOR function also fully activated the network before using the output. To determine the maximum depth of the network, as recursive function was used. This functionwas found to be unusable in large networks because it was too computationally expensive. Instead, we developed a function that recorded when any node changed its activation. If a node did change activation then the network was activated again. We continued to activate until all the nodes in the network settled.

Since NEAT is based on the concept of starting all the networks in the population minimally, there is a starter genome that is passed to the algorithm telling it the architecture to start with (i.e.

number of inputs, outputs, hidden nodes and link connections). Stanley and Miikkulainen (2002) used starter genomes with no hidden nodes and with inputs fully connected to outputs. With large networks this approach turns out to be non-minimal because the number of links is considerably more than if there were only a few hidden nodes. While the most minimal network would have only one hidden node, we decided to start with 12 hidden nodes to save time.

4.2 Evolution

4.3 Standard Evolution

In the standard evolution case a population of 250 networks was used and the number of species was kept to around five to ensure a sufficient mating pool. Other experiments used 150 networks and ten species but in these cases the species were too small to allow the population to progress due to NEAT's noise sensitivity (discussed below).

Nodes were added at a rate of 2% and links at 6%. The survival threshold representing the number of organims to survive each generation and mate was set at 25%. Weights were mutated at a rate of 0.4%. While rates this low are common in evolving game playing networks (Moriarty and Miikkulainen 1995; Lubberts and Miikkulainen 2001), Stanley and Miikkulainen (2002) used a much higher rate of 50% in double pole-balancing and other tasks. It is unclear why this should be necessary. Nevertheless, NEAT can neither succeed in double-pole balancing with a low mutation rate, nor can it succeed in Othello with a high one.

Two opponent players were employed, a random mover and an alpha-beta mover. The alpha-beta mover was similar to the one used in Rosenbloom (1982). However, it had no mobility strategy whatsoever, giving the networks a weakness to exploit. Networks were first evolved against the random mover for 100 generations and then the final population from that was used to seed the evolution against alpha-beta. This strategy was necessary because a gradient could not be established against alpha-beta without giving the networks some preparation against an easier opponent.

To create different games (because the alpha-beta mover is deterministic), a set of 244 initial boards was created representing all possible board configurations after four moves into the game. Of these five were chosen during each evaluation. Each of the five was played twice, once with the network as white and once as black giving a total of 10 games per evaluation. The number of games that the network won became its fitness. Draws were counted as losses.

As a control test a population of 150 networks with only 1 species and no structural growth (only weight mutations) was evolved as a comparison with full NEAT. This setup effectively reduced NEAT to a traditional neuro-evolutionary algorithm with fixed topology networks and no speciation. We hypothesized that based on the results of experiments conducted with NEAT and other neuro-evolution methods in Stanley and Miikkulainen (2002) that full NEAT would outperform the control experiment due to its advanced features such as speciation and growth from a minimal topology.

4.4 Coevolution

In the coevolutionary case the same mutation and survival threshold parameters were used as in standard evolution. Two populations of 150 genomes were evolved, each with about eight species. (Noise was not a problem with the evaluation function used in coevolution.)

During the evaluation of a host, opponents were changed every two games. Therefore, unlike in standard evolution, networks could play starting from the traditional starting board configuration. Each host played each of its opponents once as black and once as white. With four parasites and eight hall of famers to play, the maximum possible fitness was 24. While Rosin and Belew (1997) generally used a sampling of 50 parasites and 50 hall of famers in their game playing trials, expediency required us to use smaller subsets.

We expected that the results from this experiment would be similar to those in Lubberts and Miikkulainen (2001), with coevolution far exceeding its standard evolution counterpart.

4.5 Noise Sensitivity

Noise is an important problem facing many different areas of neural networks research. In our experiments understanding where noise is present and how is can be countered is crucial to evolving good Othello players. For us, noise occured in the evaluation function of the standard evolution experiment. When the five initial board configurations were chosen, they were chosen out of a body of 244. Therefore, the fitness assigned to a network was not be consistent from one evaluation to the next even though the network itself did not change.

It is entirely possible that a poor network (from the perspective of its performance over the entire set of 244 boards) could play from five initial boards and win every game. It is equally likely that a good network could lose every game. Normally, a large enough population would overcome this difficulty because one would expect a larger number of good players to have high fitnesses than bad players. However, in NEAT this becomes a problem because, even though the population as a whole is large enough, each species might have only about 10 members. Since genomes rarely (0.001% of the time) mate outside their species and only 25% of the species' highest performing members actually mate and survive to influence the next generation, the set of two or three members that are chosen to mate out of a species of size 10 would almost certainly contain a bad player. Increasing the survival threshold was not a viable solution because it would only allow more bad players to move on to the next generation. Two other possibilities remained, increase the number of games per evaluation to make the fitness measurement less noisy or increase the size of the species. Both of these solutions inflicted severe slow downs.

After exploring both possibilities we found that increasing the species size gave us the best chance of generating results because we could not be sure how many games per evaluation would reduce the noise to a tolerable level. On the other hand, since Moriarty and Miikkulainen (1995) used a population size of 50 to evolve networks to the mobility level of play, we knew that a species size of 50 was sufficient. Although the correct *number* of species was uncertain, for the sake of expediency, we used five species.

5 Results

Experiments involving genetic algorithms are computationally time consuming. Many different populations were evolved in the course of this research. Once performance enhancements were introduced to the NEAT algorithm part-way through this project such as altering the test for full activation of a network (see Section "Experiments", Subsection "Game Playing Neural Networks" for details), populations took about 25 minutes per generation on a 1 Ghz PC running Linux. Coevolution took longer, about 50 minutes per generation, because two populations needed to be evolved instead of one.

5.1 Standard Evolution

Networks evolved against the random mover immediately developed a positional strategy achieving win rates as high as 90% for some champions after 100 generations. Corner pieces were always captured and edge pieces were maximized when possible.

This same population was then evolved against the alpha-beta search player for 1050 generations. Initially, networks performed extremely poorly but over the first 500 generations gradually improved to the point where the best networks were winning 22% of their games. In sharp contrast to the results in Moriarty and Miikkulainen (1995) at this point our networks stopped progressing further and remained at 22% for the final 550 generations. However, despite this unusual behavior some interesting results did occur in the first 500 generations.

To demonstrate these results, ten games where a network won were taken from the tests of the 500th generation and analyzed. Figure 2 shows an average of the number of pieces each player had after each move of the game. Throughout the game the alpha-beta opponent methodically built up its piece count, but close to the end of the game it lost everything as the network moved in for the kill. The dramatic downfall of the alpha-beta's piece count indicates that it must have made itself vulnerable earlier in the game. Furthermore, the network had learned to exploit these vulnerabilities.

5.2 Competitive Coevolution

Here networks never really got off the ground when tested against the alpha-beta mover. Win rates stayed below 5%, and a mobility strategy never developed. An interesting phenomenon occurred when coevolution was seeded with a population from a standard evolution run that had already developed a mobility strategy to some degree and was defeating alpha-beta 30% of the time.

Figure 3 shows how the efficacy of the populations' best networks against alpha-beta decreased over time. The fragile mobility strategy gradually disappears, replaced by a positional strategy. This behavior indicates a serious problem in the coevolution experiment setup, leading the algorithm to decimate the populations' mobility.

5.3 Control Experiment

Networks in this experiment were evolved against alpha-beta from the same seed as standard evolution. As in standard evolution these networks gradually improved for 500 generations and then stagnated for 1500 more generations, never rising about 35% wins against alpha-beta. Here too the mobility strategy developed as a defense against the searcher's strong positional play.

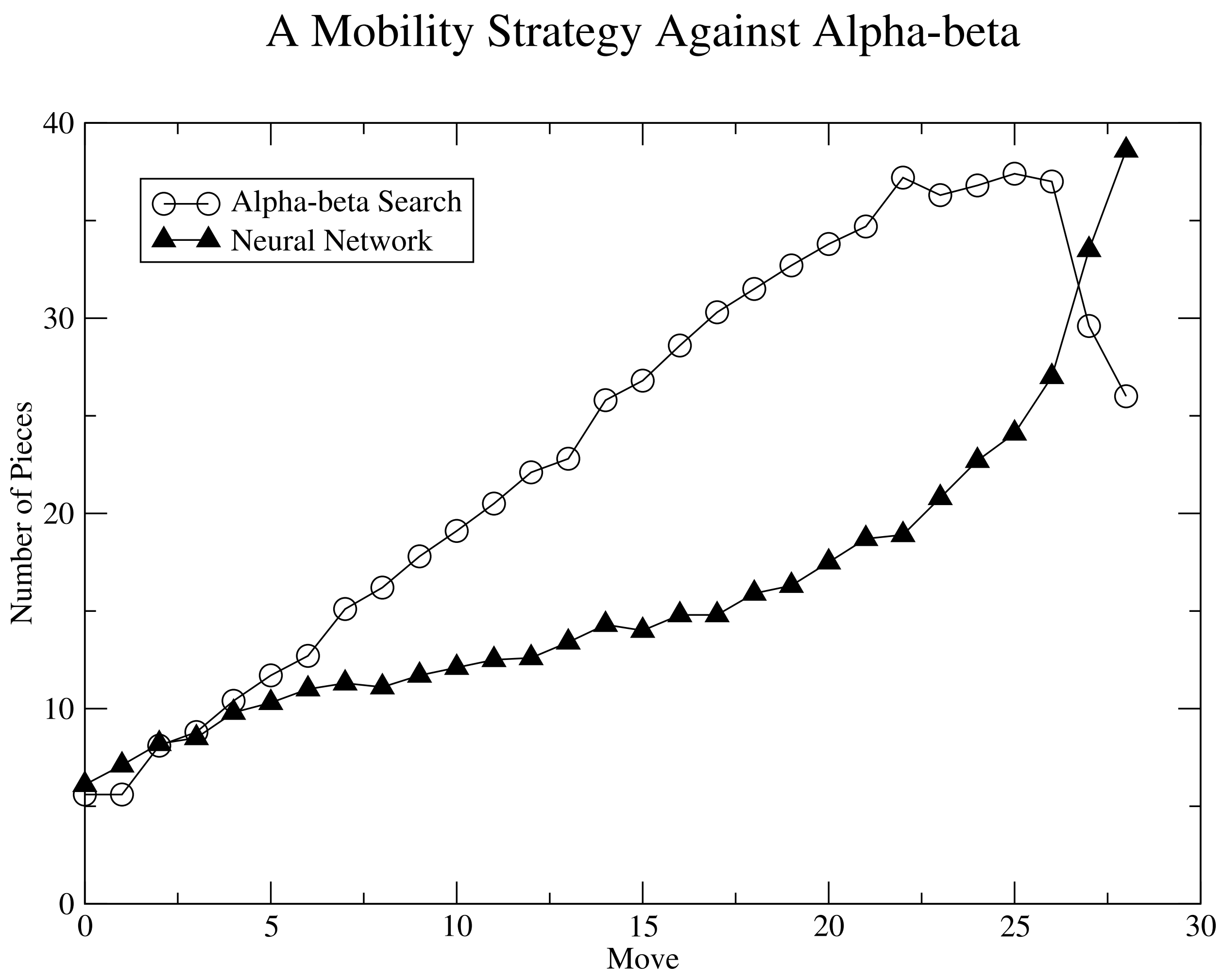

Figure 2: **The Mobility Strategy.** Alpha-beta builds up its piece count using a positional strategy but fails to notice how it has made itself vulnerable to the mobility strategy.

6 Discussion

From the results presented in the last section, it is easy to see than none of our expectations were met and that, in fact, we were faced with the opposite of what we had hypothesized.

6.1 Full NEAT versus Control Experiment

The problems that appear in the results of the standard evolution and control evolution experiments are best summed up in two questions: why is the control experiment superior to full NEAT? and Why does full NEAT stagnate? It is easy to explain the stagnation of the control experiment as the result of a lack of growth in the network. Essentially the network's memory capacity became full. Since it uses only 12 hidden nodes, this sort of behavior is expected. A larger network might

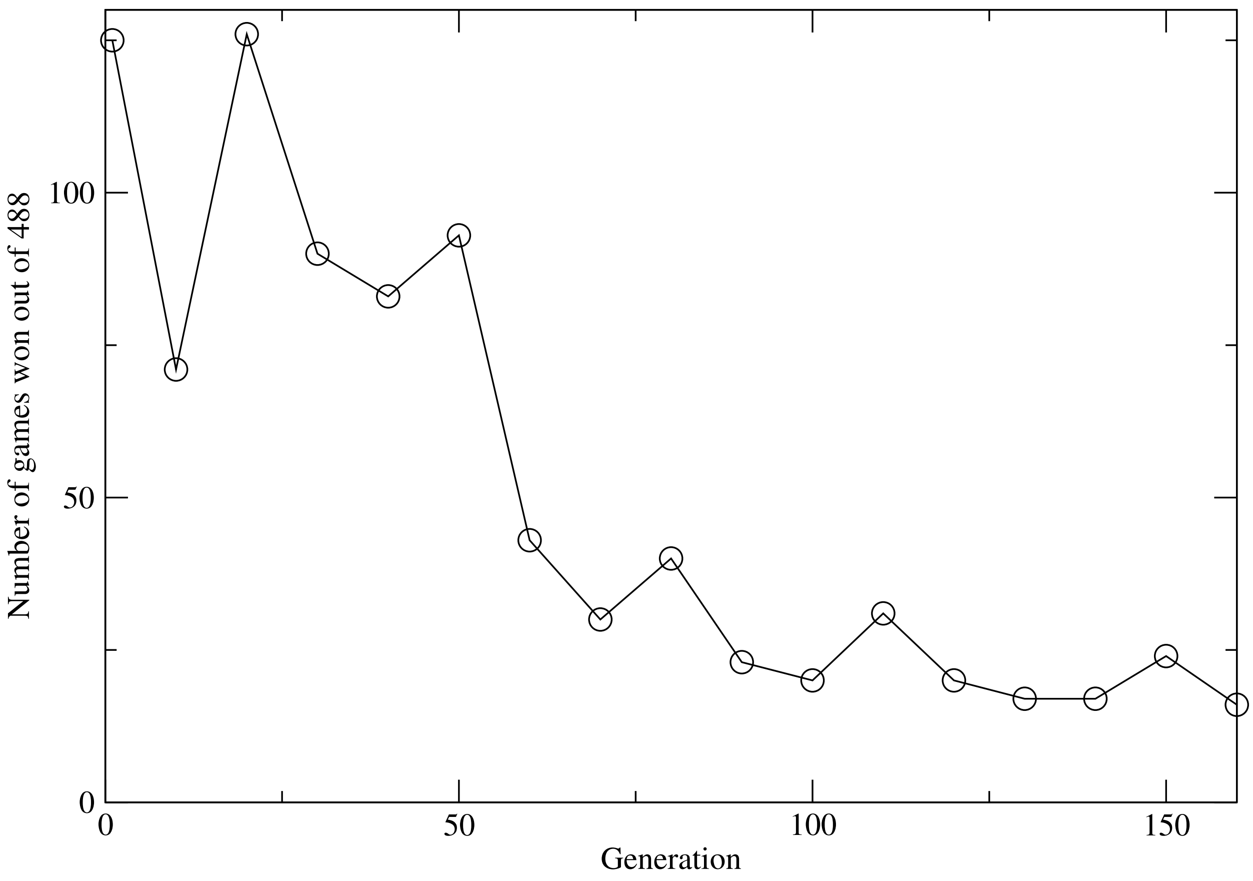

Figure 3: **Competitive Coevolution Against Alpha-beta.** As the two populations are coevolved their ability to defeat alpha-beta decreases.

be able to do even better against alpha-beta. If so, it is arguable that NEAT's advanced features such as growing topology and protecting innovations are not necessary to evolve an Othello player capable of defeating alpha-beta.

6.1.1 Why is the control experiment superior to full NEAT?

That the control experiment did so well is surprising because Moriarty and Miikkulainen (1995) point out that when a fixed topology network was evolved against alpha-beta the best networks did no better than 7% wins. Our result of 35% wins seems to stem from some aspect of NEAT that has nothing to do with growing structure *or* speciation, such as its mating algorithm.

Compared to a 35% win rate, 22% is significantly lower. Stanley and Miikkulainen (2002) points out that larger networks take longer to optimize. By the 500th generation full NEAT had evolved networks with an average of about 20 hidden nodes and 30 additional links. These are significantly larger than the networks being optimized in the control experiment. Therefore, they should take more generations to optimize. One would expect them to continue to improve beyond the control networks' limited capacity. However, this does not happen.

6.1.2 Why does full NEAT stagnate?

One possible explanation for this phenomenon is that NEAT simply did not have the number of species necessary to carry out the difficult task of evolving an Othello playing network. Since NEAT relies on its species to provide innovative strategies, the five species it had to work with may simply have all hit dead ends. If so, it may be necessary to eliminate noise using a different evaluation function rather than larger species (see Section ”Experiments”, Subsection ”Noise Sensitivity” for details), since increasing the number of species without decreasing their size would result in too much of a slow down in run time.

Another possible explanation is that the networks simply were not large enough to progress further. In Moriarty and Miikkulainen (1995) the best networks used against alpha-beta ”had an average of 110 hidden units organized in multiple layers with a high degree of recurrency.” Since our largest networks only had about 23 hidden nodes and no recurrency whatsoever, it is possible that they could not progress beyond 22%. Recurrency could be an asset in identifying the opponent's strategy over a series of moves. This hypothesis and the preceding one together could explain the stagnation.

6.2 Coevolution

The results from coevolution are easier to understand than those from standard evolution because we possess data on how a population actually loses its mobility due to the pressures of competition.

Because the mobility strategy is much more difficult to learn than the positional strategy (Billman and Shaman 1990), we expect that it will require many more generations to develop than a positional strategy. Therefore, there is a period during evolution when a strong positional strategy has developed and only a weak mobility strategy exists. These positional players would almost certainly overwhelm the burgeoning mobility players in competitive coevolution because fitness is measured in how well a host can beat a sampling of opponents, and it is entirely likely that once a good positional strategy evolves it could consistenly defeat any of the weak mobility players. The mobility players would have lower fitnesses than the positional players and would be eliminated from the population before they could develop a winning strategy. While NEAT's speciation provides some protection to developing strategies (Stanley and Miikkulainen 2002), it is difficult to protect a strategy that requires hundreds of generations to prevail. Thus, the positional strategy represents an irresistable basin of attraction in coevolution from which there is no escape.

Coevolution, rather than building upon an existing mobility strategy, takes advantage of short-term gains and develops a positional strategy capable of defeating the weak mobility strategy. This explains why in experiments coevolution slowly eliminates its populations' ability to defend against the alpha-beta mover. Since the alpha-beta mover represents a nearly optimal positional strategy, it almost always defeats another positional opponent. Only mobility can beat it.

This idea has strong implications for competitive coevolution because it can be generalized to any task where a hierarchy of strategies exists, and some are 10 or 100 times more difficult to master. Coevolution runs into a barrier with these higher stategies. It ignores strategies that pay off in the long-term in favor of short-term gains.

6.3 Mobility

One of the most important results of this research is that we see the development of the mobility strategy as a required step towards defeating alpha-beta search. The development of this sophisticated strategy, even at an intermediate level, suggests that evolutionary algorithms like those presented in Moriarty and Miikkulainen (1995) and here have the inherent ability to seek out weaknesses in their opponents and develop strategies to exploit them. That evolutionary networks have repeatedly discovered this same important breakthrough indicates the potential of these algorithms to develop problem solving strategies that might otherwise go unnoticed by human beings.

7 Future Work

In future research we will investigate the challenge presented by coevolution, and we will attempt to protect the mobility strategy and nurture it during evolution. It may be that by explicitly seeking it out and protecting it we could gain insight into how to protect difficult strategies over several hundred or thousand generations in the general case. If such a method could be developed it might have a profound impact on the study of coevolution as applied to problem solving.

Several new directions are suggested in the discussion of the standard evolution experiment. We will try using a different, less noisy evaluation function in future simulations, rather than extremely large species. This strategy is more consistent with the original research on NEAT where species were small and evaluation functions noiseless (Stanley and Miikkulainen 2002). Also we intend to introduce recurrent connections into the networks NEAT evolves and investigate faster node addition rates that would give rise to larger networks in a shorter time. Attempting the fixed topology, control experiment with a large network of as many as 100 hidden nodes could also generate interesting results.

8 Conclusion

Artificial neural networks, coupled with genetic algorithms, coevolution and other new ideas represent a set of powerful new tools in game playing and general problem solving. NEAT proved able to exploit its opponents and adapt to its environment to a point. Future research will almost certainly generate even more interesting results and be able to better explain that which we see here. Game playing provides us with a unique domain in which to study problem solving, abstracting away the richness of the real world to give us insights into how that world can be manipulated for our benefit.

References

Billman, D., and Shaman, D. (1990). Strategy knowledge and strategy change in skilled performance: A study of the game Othello. *American Journal of Psychology*, 103:145–166.

Charness, N. (1976). Memory for chess positions; resistance to interference. *Journal of Experimental Psychology*, 2:641–653.

DeGroot, A. D. (1965). *Thought and Choice in Chess.* The Hague, The Netherlands: Mouton.

Frey, P. W., and Adesman, P. (1976). Recall memory for visually presented chess positions. *Memory and Cognition*, 4:541–547.

Fullmer, B., and Miikkulainen, R. (1992). Using marker-based genetic encoding of neural networks to evolve finite-state behaviour. In Varela, F. J., and Bourgine, P., editors, *Toward a Practice* *of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, 255–262. Cambridge, MA: MIT Press.

Lee, K.-F., and Mahajan, S. (1990). The development of a world class Othello program. *Artificial Intelligence*, 43:21–36.

Lubberts, A., and Miikkulainen, R. (2001). Co-evolving a go-playing neural network. In *Coevolution: Turning Adaptive Algorithms Upon Themselves, Birds-of-a-Feather Workshop, Genetic and Evolutionary Computation Conference (GECCO-2001).*

Moriarty, D. E., and Miikkulainen, R. (1995). Discovering complex Othello strategies through evolutionary neural networks. *Connection Science*, 7(3):195–209.

Rosenbloom, P. (1982). A world championship-level Othello program. *Artificial Intelligence*, 19:279–320.

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

Schraudolph, N. N., Dayan, P., and Sejnowski, T. J. (1994). *Temporal Difference Learning of Position Evaluation in the Game of Go.* San Francisco: Morgan Kaufmann.

Staff, C. C. (1980). Background and origins of othello. *Personal Computing*, 87–8.

Stanley, K. O., and Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2). In press.