Discovering Complex Othello Strategies Through Evolutionary Neural Networks

David E. Moriarty and Risto Miikkulainen
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712-1188
moriarty,risto@cs.utexas.edu

Abstract

An approach to develop new game playing strategies based on artificial evolution of neural networks is presented. Evolution was directed to discover strategies in Othello against a random-moving opponent and later against an α-β search program. The networks discovered first a standard positional strategy, and subsequently a mobility strategy, an advanced strategy rarely seen outside of tournaments. The latter discovery demonstrates how evolutionary neural networks can develop novel solutions by turning an initial disadvantage into an advantage in a changed environment.

1 Introduction

Game playing is one of the oldest and most extensively studied areas of artificial intelligence. Games require sophisticated intelligence in a well-defined problem where success is easily measured. Games have therefore proven to be important domains for studying problem solving techniques.

Most research in game playing has centered on creating deeper searches through the possible game scenarios. Deeper searches provide more information from which to evaluate the current board position. This approach, however, is different from the play of human experts. In a psychological study, DeGroot (1965) found that game playing experts did not search any more alternative moves than novices. Experts in fact use a much greater knowledge of the game together with sophisticated pattern recognition to focus the search on the important paths (Charness 1976; Frey and Adesman 1976).

This paper presents a more "human-like" approach to game playing by evolving artificial game-playing neural networks through genetic algorithms. The networks were required to learn the game of Othello without any previous knowledge of the game. Without hand-coded rules or heuristics, the networks were free from any bias in their decision of where to move. The strategies evolved purely from discovery through playing the game.

The networks were afforded no search mechanism, and were thus forced to rely of pattern recognition on the current board configuration to achieve good play. The goal was to see what strategies the networks would develop. The networks were first evolved against a random mover, and they quickly developed a *positional strategy* similar to those often used by novice players of Othello.

A more sophisticated *mobility strategy* is often employed by tournament players because it produces much stronger play. It is, however, considered to be very hard to learn (Billman and Shaman 1990). After a positional strategy was encoded into an α-β search program and the networks were allowed to compete with the α-β program, they evolved to exploit their initial material disadvantage and discovered the mobility strategy.

The neural networks were encoded genetically based on a marker-based approach originally proposed by Fullmer and Miikkulainen (1992). For an empirical comparison, another population of networks was evolved using a fixed architecture encoding scheme, however, only the marker-based scheme turned out sufficiently powerful in this task.

The first section reviews the game of Othello. The rules are presented for the reader not familiar with the game. The next section discusses the application of genetic algorithms to this problem and the encoding of the neural networks in the marker-based scheme. Section 4 presents the main experimental results. The significance of evolving the mobility strategy is discussed in section 5 and further research is proposed.

2 The Game of Othello

2.1 Previous work

Othello has traditionally been very popular in Japan, second only to Go. It was introduced to America in the mid 1970's and soon attained international popularity. It is enjoyed by novices and experts, for its rules are simple, but complex strategies must be mastered to play the game well.

Rosenbloom (1982) created one of the first master-level Othello programs called Iago. This program was based on α-β search techniques with kill tables. Lee and Mahajan (1990) developed a successor to Iago named Bill, which was based on similar search techniques, but also implemented Bayesian learning to optimize its evaluation function. While Bill does use more knowledge in evaluating board positions, its backbone is still the α-β search.

2.2 Rules of Othello

Othello is a two-player game played on an 8×8 board. All pieces (or tiles) are identical with one white side and one black side. The initial board setup is shown in Figure 1(a). Each player takes turns placing pieces on the board with his color face up. A player may only move to an open space that causes an opponent's piece or pieces to be flanked by the new piece and another one of the player's own pieces. Pieces may be captured vertically, horizontally, or diagonally. Figure 1(b) shows the legal moves for black for the given board pattern. Once a move is made, the captured pieces are flipped. Figure 1(c) shows the board layout resulting from a move by black in the sixth row of the sixth column. The game is continued until there are no legal moves available for either player. If a player has no legal move, he must pass. The winner is the player with the most pieces in the final board configuration.

2.3 Strategies

A game of Othello can be broken down into three phases: the beginning game, the middle game, and the end game. The beginning game can be adequately handled by an opening book. The

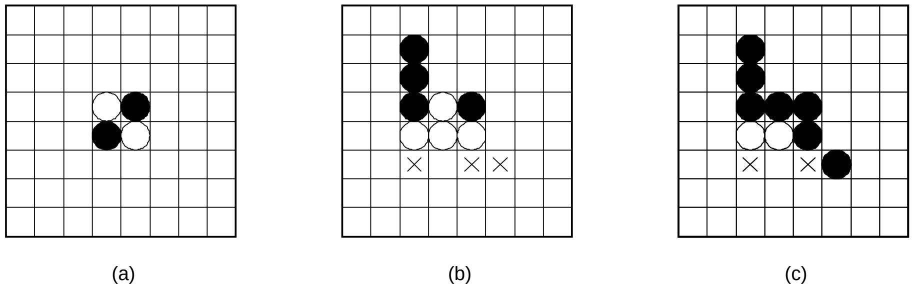

Figure 1: The Othello board: (a) The initial setup. (b) After four moves (the legal moves for black are marked with X's). (c) After black has moved to the rightmost X.

end game is simply played by maximizing your pieces while minimizing your opponent's. A good strategy for the middle game, however, is much more elusive. The goal of the middle game is to strategically position your pieces on the board so that they can be converted into a large number of permanent pieces during the end game.

There are two basic midgame strategies in Othello. A positional strategy stresses the importance of specific positions and piece configurations on the board. Places such as corners and edges are considered valuable, while others are avoided. Corners are especially valuable because once taken, they can never be recaptured. Normally, a person using a positional strategy tries to maximize his valuable pieces while minimizing his opponent's. Positional strategies are easily understood and implemented; they are often developed independently by beginners studying the game.

A much more powerful set of midgame strategies exist under the name "mobility". Even the most sophisticated positional strategy will fail against a basic understanding of mobility ideas. Here, corner capture is still considered an important mid-term goal, while taking edges and other specific formations is not. Mobility strategies are built around the idea that the easiest way to capture a corner is to force your opponent to make moves that surrender that corner. Mobility strategies often involve short term goals such as keeping a low piece count and clustering pieces. Mobility is one of the core ideas that forms the basis of all modern tournament play.

Mobility has been shown to be much harder to learn than a positional strategy (Billman and Shaman 1990). Unlike many good ideas that are often discovered independently by several people, it is widely believed that mobility was discovered only once in Japan and has since been introduced to America and Europe through American players in contact with the Japanese (Billman and Shaman 1990). Being able to independently discover a mobility strategy through evolutionary neural networks would therefore be a significant demonstration of the potential power of neuro-evolution.

3 Implementation

3.1 Game-playing Neural Networks

Our approach was to evolve a population of neural networks in the game of Othello. Each network sees the current board configuration as its input and indicates the goodness of each possible move as the output. In other words, instead of searching through the possible game scenarios for the best move, the neural networks rely on pattern recognition in deciding which move appears the most

$< start >< label >< value >< key_0 >< label_0 >< w_0 > \ldots < key_n >< label_n >< w_n >< end >$

$< start >$	- Start marker.
$< label >$	- Label of the node.
$< value >$	- Initial value of the node.
$< key_i >$	- Key that specifies whether connection is from/to an input/output unit or from another hidden unit.
$< label_i >$	- Label of the unit where connection is to be made.
$< w_i >$	- Weight of connection.
$< end >$	- End marker.

Figure 2: The definition of a hidden node in marker-based encoding. Nodes are separated by start and end markers.

promising in the current situation.

For each board space there are three possible states: it may be occupied by the network's piece, it may be occupied by the opponent's piece, or it may be unoccupied. For each board space, two input units were used. If the first unit is on, the space is occupied by the network's piece. If the second input unit is on, the space is occupied by the opponent's piece. If they are both off, the space is unoccupied. The two input units are never both on. The total number of input units was therefore 128.

Each network contained 64 output units. Each output unit corresponded directly to a space on the board. The activity of each output unit was interpreted as how strongly the network suggested moving into that space.

The network architectures were encoded in artificial chromosomes and evolved through genetic algorithms (Goldberg 1989; Holland 1975). Each chromosome was a string of 8-bit integers ranging from -128 to 127. A marker-based encoding scheme was used to encode the topology and weights of each neural network.

3.2 Marker-based Encoding

The marker-based encoding scheme is inspired by the biological structure of DNA. In DNA, strings of nucleotide triplets specify strings of amino acids that make up a protein. Since multiple proteins may be defined on the same DNA strand, certain nucleotide triplets have a special status as markers that indicate the start and the end of a protein definition (Griffiths et al. 1993).

Artificial genes can similarly use markers to define separate nodes in a neural network. Each node definition contains a start integer and an end integer. The integers in-between specify the node. Figure 2 outlines the node definition in the marker-based scheme. The start and end markers are determined by their absolute value. If the value of an integer MOD 25 equals 1, it is a start marker. If the value MOD 25 equals 2, it is an end marker. Any integer between a start marker and an end marker is always part of the genetic code. Therefore, an integer whose value MOD 25 equals 1 may exist between two markers and will not be treated as a marker.

Since 8-bit integers for the connection labels only allow for 256 distinct connection labels, using 128 for the input units and 64 for the output units would only leave 64 labels for the hidden units. To avoid this restriction, in our implementation of the marker-based encoding each connection label consists of two 8-bit integers. The *key* integer specifies whether the connection is to be made with the input/output layers or with another hidden unit. If the key is positive, the second integer, *label*,

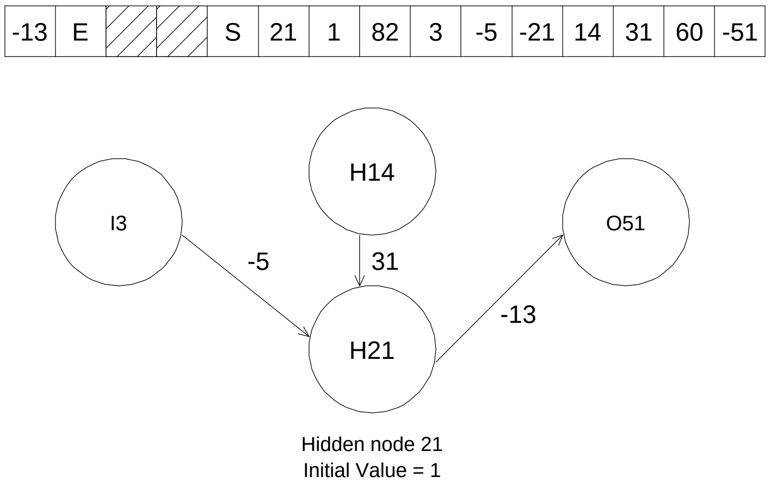

Figure 3: An example node definition in a marker-based gene. For example, the first connection has $key = 82$, $label = 3$, $w = -5$. The key and label are both positive so the connection is to be made from input unit 3. The last connection wraps around to the beginning of the chromosome.

specifies a connection from the input layer (if the label is > 0) or to the output layer (if the label is < 0). If the key is negative, the label specifies an input connection from another hidden unit. For the input and output layer connections, the identity of the input or output unit is determined by the value of the label MOD N, where N is the number of units in the layer. For the connections within the hidden layer, the connection is made from the hidden unit whose label value is closest to the connection label. If two or more hidden units have the same label, the first hidden unit on the chromosome is chosen for the connection.

The chromosome is treated as a continuous circular entity. A node definition may begin at the end of the chromosome and wrap around to the beginning. Such a node definition is terminated by an end marker or when the first start marker in the chromosome is encountered. Figure 3 shows an example gene and the network information it encodes.

The hidden units are activated once per network activation in the order specified on the chromosome. Hidden units are allowed to retain their values after each activation. This allows the networks to possibly use their hidden units as short-term memory.

The power of the marker-based scheme, compared to other neuro-evolution techniques, comes from two sources. The first is the ability to evolve the network architecture along with the weights. Some environments may be best served by a large number of hidden units with few connections per unit. Other environments may require less hidden units with a larger number of connections per unit. The marker-based encoding can adapt the network architecture to best fit the environment. The second advantage is that the interpretation of each allele is not dependent on its absolute location (locus) on the chromosome. In most neuro-evolution approaches, each position on the chromosome corresponds directly to a weight in the network (Belew et al. 1991; Jefferson et al. 1991; Werner and Dyer 1991; Whitley et al. 1990). A marker-based system, however, evaluates alleles according to their position relative to a start marker, which gives the genetic algorithm more freedom to explore useful schemata. For example, two particular hidden neuron definitions may together form an extremely useful building block. In the standard approach, the two definitions may be spread out over a large portion of the chromosome making them difficult to propagate to future generations. In the marker-based approach, however, the two neurons can be evolved in sequential node definitions, which results in a much smaller schema defining length. Short, highly functional schemata of this kind are hypothesized to be essential for efficient operation of genetic algorithms (Goldberg 1989).

The marker-based scheme presented above is a successor of Fullmer and Miikkulainen (1992). Whereas they defined output nodes explicitly like all other nodes in the network, our version of marker-based encoding only requires the definition of hidden nodes. The connections to output units are given as part of the hidden node definitions, resulting in a more compact encoding when the output layer is large.

3.3 Evolution

A Population of 50 networks was evolved, each with a 5000 integer chromosome. A two point crossover was employed to mate the top 12 networks. A mate for each network was selected randomly among all networks with a greater or equal fitness value. Two offspring were produced per mating resulting in 24 new networks per generation. The new networks replaced the worst networks in the population. Mutation, at the rate of 0.4%, was implemented at the integer level rather than the bit level by adding a random value to an allele if mutation was to occur. A number that exceeded the [-128,127] range was "wrapped around."

The networks were given the current board configuration as input. Out of all output units that represented a legal move in a given situation, the one with the highest value was selected as the network's move. In other words, the networks were not required to decide which moves were legal, but only to differentiate between legal moves. This strategy turned out to speed up the evolution a great deal, while still allowing the networks to evolve good game-playing skills.

The networks were initially evolved against a random move maker and later against an α-β search program. To create different games, an initial state was selected randomly among the 244 possible board positions after four moves. Each network's fitness was determined by the number of games won out of ten played against the α-β search program searching three levels down. The α-β program used a positional strategy similar to Iago's (Rosenbloom 1982). Iago's evaluation function also contained a complex mobility strategy, which was purposely left out to provide a weakness that the networks could exploit. The networks' hidden units were reset to their initial values after each game.

4 Results

Experiments on artificial evolution are computationally expensive, and many different populations were evolved throughout this research. Typically, the populations required 24 hours of CPU time on an IBM RS6000 25T workstation to evolve significant behavior.

The networks playing against a random mover immediately began to develop a positional strategy. Whenever possible they would maximize their edge pieces, while trying to minimize those of their opponent's. The networks' middle games consisted of taking edge and corner pieces with little regard to the middle squares. Besides recognizing valuable positions, the networks also learned which positions should be avoided, like those next to corners. These positions are undesirable if the corner is not already captured because they immediately lead to corner capture by the opponent. Within 100 generations, the best networks could defeat a random mover 97% of the time.

To better determine how the networks were playing, ten games in which a network won were

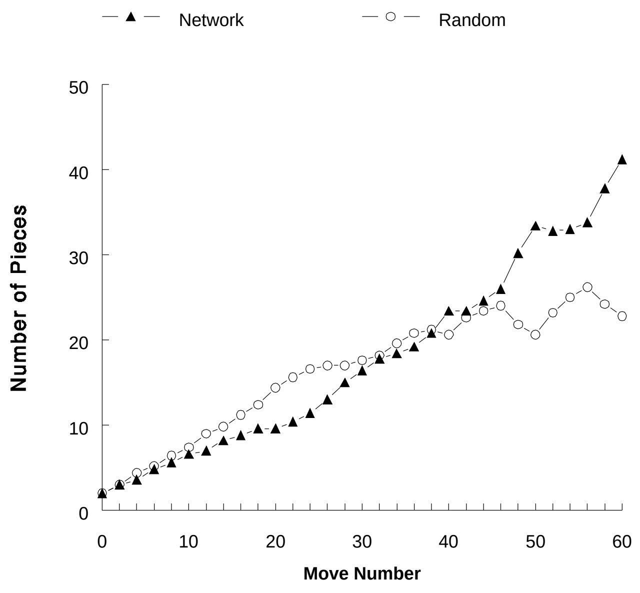

Figure 4: The average number of pieces throughout the game for the random mover and the network evolved through 100 generations against a random mover. There are 60 total moves in a game. The network and random mover monotonically increase their piece count until move 45, when the network's stronger positions allow it to efficiently capture the random mover's pieces.

chosen at random and analyzed. Figure 4 shows the average number of pieces at each point of the game for the network and the random mover. Throughout most of the game, the network and random mover monotonically increased their piece counts, which is typical of most games played using a positional strategy. However, since the random mover does not prefer stronger positions, it is unable to provide a strong defense of its pieces in the end game. Thus, around move 45 the random mover's piece count begins to decline and the network's piece count rises more rapidly. The network's strategically anchored positions allow efficient capture of the random mover's pieces without the need to sacrifice many of the network's own pieces.

The same population of networks obtained against the random mover was further evolved against the α-β search program (also called the searcher below). Initially the networks performed very poorly. As the populations evolved, however, their performance began to improve, and after 2000 generations the best networks were winning 70% of their games against the searcher. An analysis of the networks' games showed that the networks' strategies had changed drastically. They were not as interested in taking edge pieces as before. Instead, their attention was focused on moves that captured fewer discs and controlled the center of the board position. However, the value attached to corner squares had not changed, and if presented a legal corner move the networks would immediately capture it.

Figure 5 shows the average piece counts throughout the game for the network and the searcher during ten games won by the network. The results show that the networks have indeed changed their strategy from that used against the random mover and are playing a remarkably different middle game than the searcher. While the searcher is taking a greedy positional approach maximizing its positions, the networks are keeping the number of their pieces relatively small. The turning point in the games come around move 54, which is normally when strategic mobility advantages are converted into stable, permanent discs. As shown in figure 5 the network's pieces dramatically increase while the searcher's decrease.

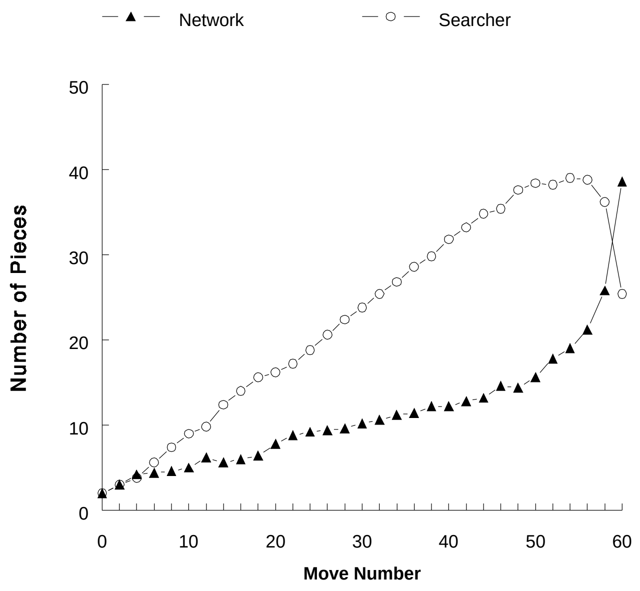

Figure 5: The average number of pieces throughout the game for the α-β search program and the network further evolved through 2000 generations against the α-β program. Throughout most of the game, the searcher attempts to maximize its positions and consequently its piece count, while the network keeps a relatively low piece count. A dramatic shift in power occurs near the end of the game, which is typical in mobility play.

Such a dramatic change could only have resulted from the searcher making a vulnerable move. Since the searcher does not normally give up good positions, it must have had no alternative. Figure 6 shows that during the crucial middle and end games the network had an average of twice as many moves to choose from than the searcher.

Figure 7 shows an actual game the authors played against one of the networks with the network playing white. Figure 7(a) shows the board configuration near the end of the middle game with white to move next. From a positional sense, the game looks very good for black. Black has conquered almost all of the edge pieces (although no corners) and is dominating the board. However, while black is positionally strong, his mobility is very limited. After white moves to f2, the only legal moves for black are b2, g2, and g7, which each lead to a corner capture by the network. By limiting the available moves (mobility), the network forces black to make a bad move. Figure 7(b) shows the final board where the network has taken every corner and captured all of black's edge pieces. The final score was 48-16 in the network's favor. The statistical results and the analysis of individual games clearly indicate that the network is using the mobility strategy to defeat the weaker positional strategy.

To verify that the network would apply the same strategy against other opponents as well, it was also played against the random mover. Figure 8 shows the number of pieces throughout a game for the mobility network and random mover averaged over 10 games. The network continued to keep a low piece count until move 52 when a shift in power occurred. Since the random mover has no built-in strategy to maximize its pieces, its high piece count must be the result of the mobility network's strategy to keep its own piece count low. This strategy is quite different from the original (positional) network's games against the random mover (figure 4). For example, at move 40 the positional network averaged 25 pieces compared to 20 pieces for the random mover, whereas the mobility network averaged only 16 pieces compared to 29 for the random mover. Since the random-

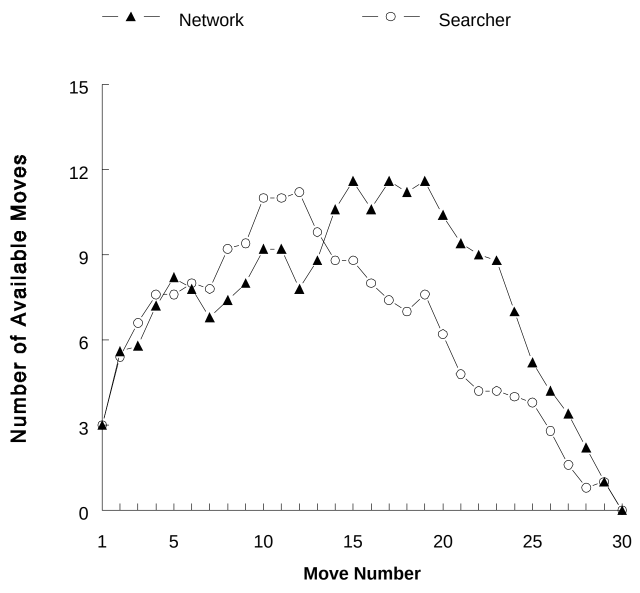

Figure 6: The average number of available moves for each player. Each player makes 30 moves throughout the game. During the middle and end game, the searcher's move choices become more limited.

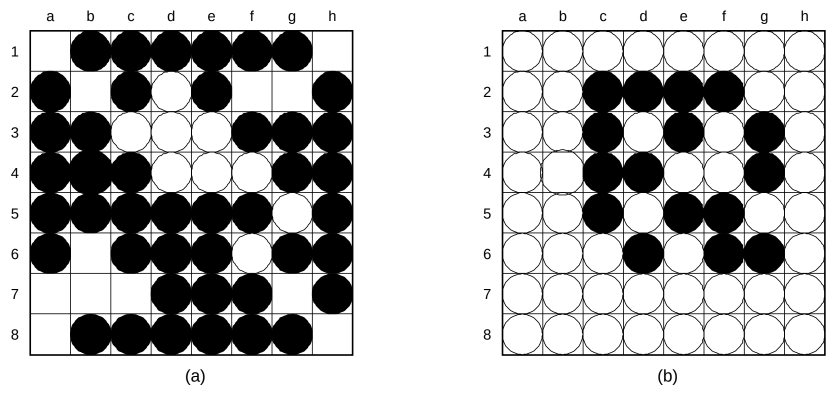

Figure 7: A game the authors played against a network. (a) A situation in the middle game with the network playing white. (b) The final board configuration. The network allowed the authors to dominate the board and forced us to make a vunerable move. The network then systematically captured the majority of our pieces.

mover is playing at the same level in both cases, we can conclude that the two networks (positional and mobility) are indeed employing different middle game strategies to beat the same opponent.

Although such quantitative observations are illuminating, it is also important to look at individual games and verify that the networks are indeed following a recognizable strategy. To this end, transcripts of the mobility network's games were analyzed by David Shaman, the 1993 world Othello champion. To help characterize the networks' play, Shaman divides middle game moves into 6 types:

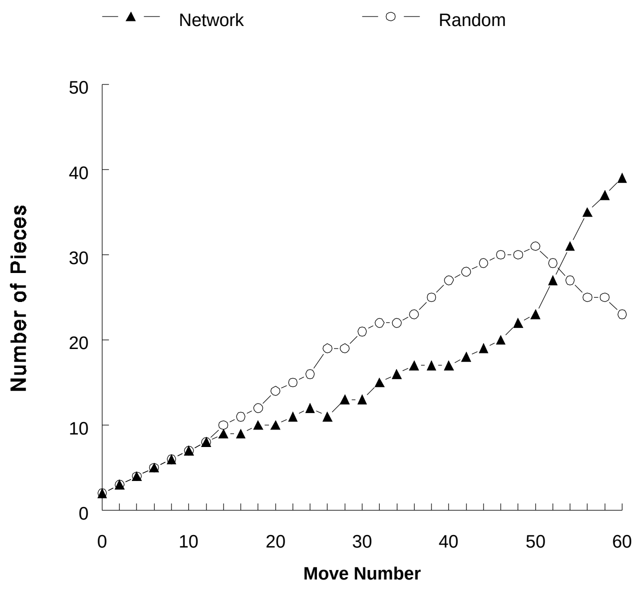

Figure 8: The average number of pieces for the mobility network and the random mover. The mobility network is the best network found after evolving against the α-β search program. The network continues to keep a low piece count, while allowing the opponent to monopolize the board during the beginning and middle game. Comparisons to figure 4 show that the networks are indeed using a different strategy.

1. Moves that all world class players would agree are best.
2. Moves that are good for mobility and have no obvious disadvantages.
3. Moves that look good from a mobility standpoint, but must be ruled out for other reasons.
4. Moves that are bad from a mobility standpoint, but have some other legitimate justification.
5. Moves that are bad from a mobility standpoint, but may have positional justifications.
6. Moves that are bad and inexplicable based on any known ideas.

From the game transcripts against the α-β search program, Shaman concluded that the networks did indeed exhibit play based on mobility. Most of the time the networks favored moves of type 2, occasionally selecting moves of type 1, but also sometimes types 3 and higher. The middle games were characterized by a low piece count, connected pieces, and control of the center, which all support mobility. Finally, if told that the games were played by a human, Shaman would would characterize the player as follows:

> This is someone who has been playing for a while and thought about the game. They've just been introduced to the idea of mobility. They are not very good yet. They are usually choosing the right type of move, but only occasionally choosing the best move. Unfortunately, sometimes they seem at a bit of a loss as to what to do – they then often revert to positional play or even just play an inexplicable bad move.

Unlike a human, however, these networks were not introduced to the idea of mobility, but rather discovered it independently by playing against an imperfect opponent. It is the ability to discover difficult, counterintuitive strategies that makes the neuro-evolution approach significant.

5 Discussion

Marker-based encoding turned out crucial for evolving the desired behavior. As a baseline comparison, another population of game-playing networks was evolved with a more standard fixed-architecture encoding. The networks consisted of three layers (input, output, and one hidden layer) that were fully connected. To keep the chromosomes the same relative length as with the marker-based encoding, 40 hidden units were used, resulting in a chromosome length of 5160 integers. Each hidden unit had a recursive connection to itself to allow short-term memory to develop. The chromosome was simply the concatenation of the weights in the network. Similar fixed architecture encoding techniques have been shown to be effective in domains such as evolving communication (Werner and Dyer 1991), processing sonar signals (Montana and Davis 1989), and trail-following (Jefferson et al. 1991). However, fixed encoding turned out inadequate for this task. The fixed-architecture networks achieved similar performance (97% winning percentage) against a random mover, although it took 1000 generations. Against the α-β search program, the best network was able to win only 7% of its games after 2000 more generations. It seems that in the fixed-architecture encoding approach, the genetic algorithm was less successful in generating novel solutions, which is required to turn the initial disadvantage into a winning strategy.

The best marker-based networks in the 10 experiments had an average of 110 hidden units organized in multiple layers with a high degree of recurrency. In general, recurrent connections are advantageous because they allow the network to incorporate knowledge of previous activations into the current activation. In this task, however, only pattern recognition based on the current activation is required, and good performance can be evolved even with a marker-based encoding of feed-forward networks. Recurrency was allowed in these simulations to determine whether a more general encoding would produce effective networks. Recurrent connections may emerge as a powerful asset if the networks are required to play against a wide range of opponents. The short-term memory, provided by the recurrent connections, could help identify the opponent's strategy and incorporate this knowledge into future move decisions.

Since the networks were not required to learn the concept of legal moves, at first glance it seems that they would not evolve to play the game correctly. For example, often a network would output a high value in a position (e.g. in a corner) which would otherwise be a good move, but illegal in the current configuration and therefore not deserve consideration. Interestingly, such behavior has a counterpart in human play. Humans will often look at a board position and recognize desirable moves that are illegal. They realize that certain moves, such as corners in Othello, are almost always advantageous and keep them in mind until they become legal. Through ranking all moves legal or illegal, the networks' play seems to go through a similar process.

The networks develop a positional strategy very much like human beginners. Most novices recognize the importance of corner pieces early on and somewhat later that of other pieces that can also be stabilized. Such positional strategy is very concrete and easy to understand. It is also easier to learn and sufficient to defeat a random mover and was thus employed by the networks early on. The mobility strategy, however, is difficult for humans to discover because it is counterintuitive. Intuitively, most human game players feel they should try to conquer as much territory as possible. The artificial networks have no such bias, and are much more flexible in considering alternative strategies.

The mobility strategy evolved quite naturally as a response to the environment. The α-β search program used a strong positional strategy and was allowed to search several moves ahead. The networks' positional strategy was developed against a random mover and was not nearly as sophisticated. As a result, the networks' piece count remained low throughout the game. However, the networks discovered that they could often win in such situations by improving their mobility instead of their positional game. Effectively, the networks turned around the search program's strategy and used it against itself. The evolution was able to turn the initial disadvantage into a novel advantage. A similar process often appears to take place in successful natural evolution and adaptation into a changing environment.

Such novel strategies are very difficult for human players to discover. Human novices often try to mimic the strategies of better players to achieve better play. A human in the same situation would have tried to improve his positional strategy to make the games closer, preventing him from seeing the advantages of mobility. If neuro-evolution had been applied to Othello in the early 1970's, it is possible that the positional strategy could have been shown vulnerable years before any human discovered its weaknesses.

Discovering a known counterintuitive strategy demonstrates the power of neuro-evolution. In principle it should be possible to develop truly new strategies as well. In preliminary experiments, we have replaced the positional opponent described in this paper with the Bill program (Lee and Mahajan 1990), which contains sophisticated positional and mobility strategies optimized through Bayesian learning. To date, the networks have been unable to discover a strategy that can defeat Bill. This result suggests that there may be no novel strategy in Othello that can exploit weaknesses in players with strong positional and mobility components. The neuro-evolution approach, however, is certainly not limited to Othello or even game playing. Domains like natural language processing, planning, and automatic theorem proving also rely on extensive search. By forcing neural networks to compete with current heuristic search methods, better evaluation techniques could be discovered and implemented to better guide a search.

Another question to explore in future research is whether networks can evolve significant play by playing each other. Co-evolving populations would preclude the need for a strong, searching opponent, which is typically the bottleneck in these simulations. Additionally, the networks should evolve more general game-playing strategies since the opponent's strategy is not constant. A difficult issue that must be addressed, however, is how to judge performance of one network relative to other networks in the population. Since the strength of the opponents will vary, a weak network may exhibit strong play simply because its opponents were sub-par. Such fitness evaluation noise could be averaged out by playing each network against more opponents, however, this would greatly increase the time needed for evolution.

6 Conclusion

Artificial evolution of neural networks provides a promising new paradigm for developing new problem-solving strategies. In Othello, a strategy that eluded experts for years was evolved in a week. The marker-based approach for encoding neural networks proved to very effective at discovering novel solutions and adapting to changes in the environment. It should be possible to use this same approach to find new strategies and heuristics in other domains such as planning, theorem proving, and natural language processing as well.

Acknowledgments

The authors are grateful to David Shaman for providing both invaluable Othello expertise and helpful comments on the manuscript.

References

Belew, R. K., McInerney, J., and Schraudolph, N. N. (1991). Evolving networks: Using genetic algorithm with connectionist learning. In Farmer, J. D., Langton, C., Rasmussen, S., and Taylor, C., editors, *Artificial Life II.* Reading, MA: Addison-Wesley.

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). Evolving finite state behavior using marker-based genetic encoding of neural networks. In *Proceedings of the First European Conference on Artificial Life.* Cambridge, MA: MIT Press.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning.* Reading, MA: Addison-Wesley.

Griffiths, A. J. F., Miller, J. H., Suzuki, D. T., Lewontin, R. C., and Gelbart, W. M. (1993). *An introduction to Genetic Analysis.* W. H. Freeman.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence.* Ann Arbor, MI: University of Michigan Press.

Jefferson, D., Collins, R., Cooper, C., Dyer, M., Flowers, M., Korf, R., Taylor, C., and Wang, A. (1991). Evolution as a theme in artificial life: The genesys/tracker system. In Farmer, J. D., Langton, C., Rasmussen, S., and Taylor, C., editors, *Artificial Life II.* Reading, MA: Addison-Wesley.

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

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

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

Werner, G. M., and Dyer, M. G. (1991). Evolution of communication in artificial organisms. In Farmer, J. D., Langton, C., Rasmussen, S., and Taylor, C., editors, *Artificial Life II.* Reading, MA: Addison-Wesley.

Whitley, D., Starkweather, T., and Bogart, C. (1990). Genetic algorithms and neural networks: Optimizing connections and connectivity. *Parallel Computing*, 14:347–361.