Evolving Neural Networks to Focus Minimax Search *

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

Abstract

Neural networks were evolved through genetic algorithms to focus minimax search in the game of Othello. At each level of the search tree, the focus networks decide which moves are promising enough to be explored further. The networks effectively hide problem states from minimax based on the knowledge they have evolved about the limitations of minimax and the evaluation function. Focus networks were encoded in marker–based chromosomes and were evolved against a full–width minimax opponent that used the same evaluation function. The networks were able to guide the search away from poor information, resulting in stronger play while examining fewer states. When evolved with a highly sophisticated evaluation function of the Bill program, the system was able to match Bill's performance while only searching a subset of the moves.

Introduction

Almost all current game programs rely on the minimax search algorithm (Shannon 1950) to return the best move. Because of time and space constraints, searching to the end of the game is not feasible for most games. Heuristic evaluation functions, therefore, are used to approximate the payoff of a state. Unfortunately, heuristics create errors that propagate up the search tree, and can greatly diminish the effectiveness of minimax (Korf 1988). Minimax also assumes that the opponent will always make the best move. It does not promote risk taking. Often in losing situations the best move may not be towards the highest min/max value, especially if it will still result in a loss. Knowledge of move probabilities could guide a search towards a more aggressive approach and take advantage of possible mistakes by the opponent.

Recently, several algorithms have emerged that are more selective than the standard fixed-depth minimax search (Korf and Chickering 1994; McAllester 1988; Rivest 1987). These algorithms allow moves that appear more promising to be explored deeper than others, creating nonuniform-depth trees. While these techniques have lead to better play, they still allow minimax to evaluate every unexplored board and are therefore vulnerable to errors in the evaluation function.

Most game programs overcome weak evaluation functions by searching deeper in the tree. Presumably, as the search frontier gets closer to the goal states, the heuristic evaluations become more accurate. While this may be true, there is no guarantee that deeper searches will provide frontier nodes closer to the goal states. Hansson and Mayer (1989) have shown that without a sound inference mechanism, deeper searches can actually cause more error in the frontier nodes. A more directed search, therefore, seems necessary.

An alternative to deeper searches is to decrease the errors in the evaluation function. Bayesian learning has been implemented to combine several heuristic estimates (Lee and Mahajan 1990) and to adjust the heuristic values based on values of other nodes in the tree (Hansson and Mayer 1989). The new estimates represent a measure of belief in the heuristic value. These methods have provided stronger play, although they do not address problems inherent in minimax such as no risk taking.

This paper presents a novel approach using evolutionary neural networks that can compensate for problems in the evaluation function as well as in the minimax algorithm. Artificial neural networks have proven very effective in pattern recognition and pattern association tasks, which makes them a good candidate for recognizing undesirable board situations. Genetic algorithms provide a powerful, general training tool for neural networks. Like natural evolution, artificial evolution is very good at discerning problems and finding ways to overcome them. Our approach is based on a marker-based encoding of neural networks which has been shown particularly effective in adapting to new challenges in complex environments (Fullmer and Miikkulainen 1992; Moriarty and Miikkulainen 1993).

Genetic algorithms were used to evolve *Focus networks* to direct a minimax search away from poor infor-

*Thanks to Kai-Fu Lee and Richard Korf for providing the source code for Bill's evaluation function.

In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*. Seattle, WA.

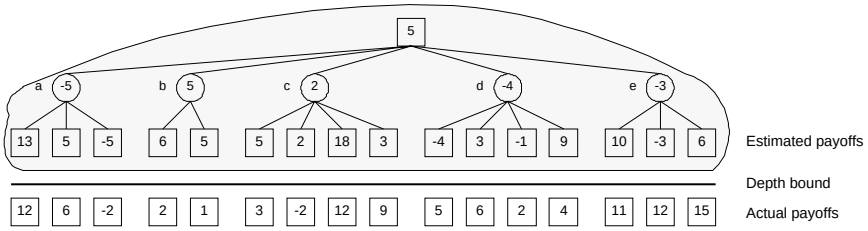

Figure 1: A full-width minimax search to level 2. All nodes in the shaded area are evaluated. The actual payoff values of the leaf states are listed below the depth bound. Their heuristic estimates are shown inside the leaf nodes. Min (circles) selects the lowest payoff and max (squares) the highest of min's choices. As a result, move b is selected for the root.

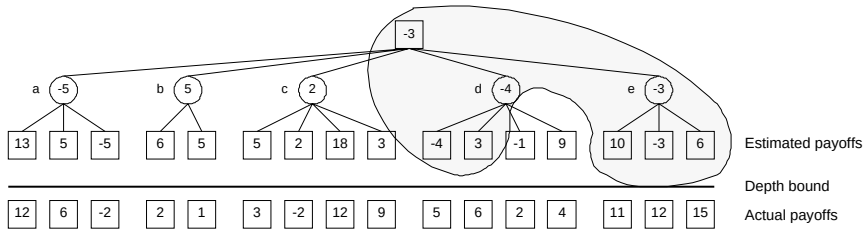

Figure 2: A focused minimax search. Only the states in the focus window (the shaded region) are evaluated. As a result, move e appears to be max's best choice.

mation. At each state in the search, the focus network determines which moves look promising enough to be further explored. The focus network is able to control which moves the minimax search can see, and can evolve to overcome limitations of the evaluation function and minimax by focusing the search away from problem states.

A population of focus networks was evolved in the game of Othello. The results show that the focus networks are capable of stronger play than full–width minimax with the same evaluation function, while examining fewer positions. Also, when evolved with the highly sophisticated evaluation function of the Bill program (Lee and Mahajan 1990), the focus networks were able to maintain Bill's level of play while searching through fewer states.

The next section describes the basic idea and implementation of the focus networks. Section 3 describes marker-based encoding and the specifics of the evolution simulations. The main experimental results are presented in section 4, and discussed in section 5.

Focus Networks

Selecting Moves for Minimax

Focus networks decide which moves in a given board situation are to be explored. At each level, the network sees the updated board and evaluates each move. Only those moves that are better than a threshold value will be further explored. This subset of moves can be seen as a window to the search tree returned by the focus network. The search continues until a fixed depth bound is reached. A static evaluation function is applied to the leaf states, and the values are propagated up the tree using the standard minimax method. The α-β pruning algorithm (Edwards and Hart 1963; Knuth and Moore 1975) is used as in a full–width search to prune irrelevant states.

To illustrate how such control of minimax might be beneficial, consider the following situation. Two moves, A and B, are considered in the current board configuration. Although move A returns, through minimax search, a higher evaluation value than move B, both moves appear to lead to losing situations. Move B, however, can result in a win if the opponent makes a mistake. By assuming that the opponent will always make the best move, minimax would choose A over B resulting in a sure loss. Focus networks, however, could learn that a win can sometimes be achieved by selecting move B, and they would thus not include A in their search window.

More generally, restricting the number of moves explored has two advantages: (1) the branching factor is reduced which greatly speeds up the search. As a result, searches can proceed deeper on more promising paths. (2) The focus networks are forced to decide which moves the minimax search should evaluate, and in order to play well, they must develop an understanding of the minimax algorithm. It is possible that they will also discover limitations of minimax and the evaluation function, and learn to compensate by not allowing minimax to see certain moves.

Figures 1 and 2 illustrate the focused search process. The current player has a choice of 5 moves (a through e). Figure 1 shows a basic minimax search with a depth bound of 2. The leaf states are evaluated according to

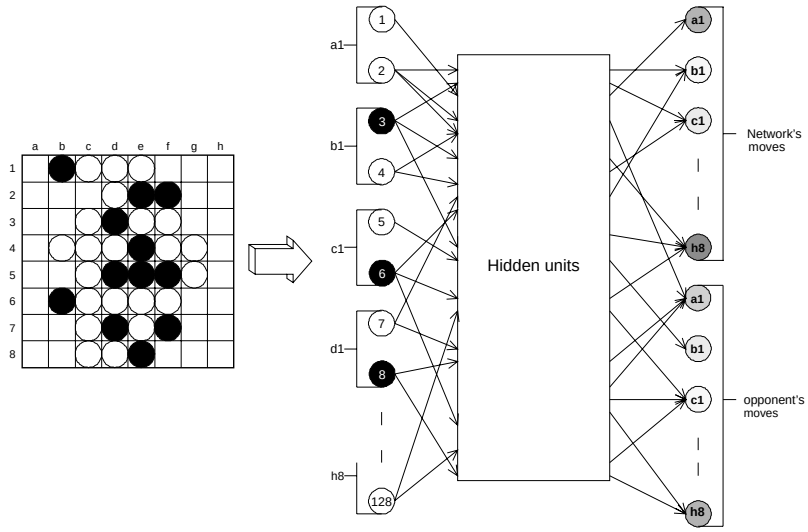

Figure 3: The architecture of the focus networks for Othello. Two inputs are used to encode each position on the board. The encoding of the first four spaces (a1, b1, c1, d1) for the given board with the network playing black are shown in the input layer. Both input nodes 1 and 2 are off since a1 is empty. Node 3 is on (i.e. dark) since b1 has the network's piece in it, and nodes 6 and 8 are on since the opponent has pieces in c1 and d1 (both nodes for the same position are never on simultaneously). The activation of the output layer is shown by the shading. The corners (such as a1 and h8) have high activations since corners are almost always good moves. Only the input and output encoding was prespecified for the network. The input and output connectivity and the number and connectivity of the hidden nodes were all evolved using genetic algorithms.

a static evaluation function. The actual payoff value of each leaf is shown below the depth bound. The difference between these values is the error or misinformation generated by the evaluation function. The best move is e, as it will generate a payoff of at least 11. Because of the misinformation, however, full-width minimax would choose move b. Figure 2 shows the same search tree but with the addition of a focus window. Only the nodes in the window are evaluated. By focusing the search away from the poor information, the best move (e) would be selected. The question is, how can we reliably form such a search window?

The evolutionary approach is attractive because no previous knowledge of minimax or the evaluation function is needed. The usual neural network learning algorithms such as backpropagation (Rumelhart et al. 1986) would require exact target values to be specified for each training example. Such information is very difficult to establish in the search focus task. In the neuro-evolution approach, however, evolutionary pressures will guide the networks toward providing good windows for the search. Networks will discover misinformation by associating certain board situations with winning and losing. Networks that prune out problem states will win more games, allowing them to survive and propagate their genes to future networks.

Implementation in Othello

Othello is a board game played on an 8×8 grid (figure 3). Each piece has one white and one black side. Players ("white" and "black") take turns placing pieces on the board with their own color facing up until there are no further moves. For a move to be legal, it must cause one or more of the opponent's pieces to be surrounded by the new piece and another of the player's pieces. All surrounded pieces are subsequently flipped to become the player's pieces. Several world championship-level Othello programs have been created using full-width minimax search (Lee and Mahajan 1990; Rosenbloom 1982). Like most advanced game programs, they achieve high performance through examining millions of positions per move.

In our implementation of focus networks, two input units were used to represent the type of piece in each board space. Each output unit corresponded directly to a space on the board. The activation of an output unit determined how strongly the network suggested moving to that position. Separate output units were used for the two players. Thus, the ranking for the network's moves may differ from the ranking of the opponent's moves. This distinction is beneficial since an aggressive player should not assume his opponent is equally aggressive and should take a more conservative approach when predicting his opponent's moves. Similarly, a defensive player should not presume defensive play from his opponents. The separation of player and opponent's output units allows offensive and defensive strategies to develop.

The number of hidden units and connections between them were determined through evolution. Each

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

$< start >$	- Start marker.
$< label >$	- Label of the node.
$< value >$	- Initial value of the node.
$< key_i >$	- Key that specifies whether connection is from an input unit/to an 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 4: The definition of a hidden node in marker–based encoding.

hidden unit used a linear threshold of 0 to determine it's output (either 0 or 1). Usually the networks contained about 120 hidden nodes and 600 connections with a large amount of recurrency. For each state to be explored in a search tree an activation was propagated through the network. The legal moves with activation greater than or equal to 0 were included in the search window.

Evolution

Each focus network's genetic representation was based on a marker–based encoding (Fullmer and Miikkulainen 1992) of the architecture and weights. The encoding is inspired by markers in DNA that separate protein definitions. Artificial markers in the chromosome are used to separate neural network node definitions. Alleles serve as start markers if their absolute value MOD 25 equals 1 and end markers if their absolute value MOD 25 equals 2. Any integer between a start marker and an end marker is always part of the genetic code. The interpretation of non-marker alleles depends on their location with respect to a start or an end marker. Figure 4 summarizes the structure of the hidden node definition in marker-based encoding.

Each chromosome consisted of 5000 8-bit integers ranging from -128 to 127. Two 8-bit integers were used for the connection definitions. 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*, 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. Figure 5 shows an example gene and the network information it encodes.

The chromosome is treated as a continuous circular entity. A node may begin on one end of the chromosome and end on the other. The final node definition is terminated, however, if the first start marker is encountered in the node definition. The hidden nodes were evaluated in the order specified in the chromosome.

A population of 50 networks was evolved using standard genetic algorithms (Goldberg 1988; Holland 1975). A two point crossover (figure 6) was used to produce two offspring per mating. Only the top 15 networks were allowed to mate with each other, creating 30 new offspring per generation. The new offspring replaced the least fit networks in the population. Traits that previously led to high fitness levels were passed to future generations, whereas traits that led to poor performance were selected against. Mutation, at the rate of 0.4%, was implemented at the integer level by adding a random value to an integer allele. The top 3 networks were not mutated.

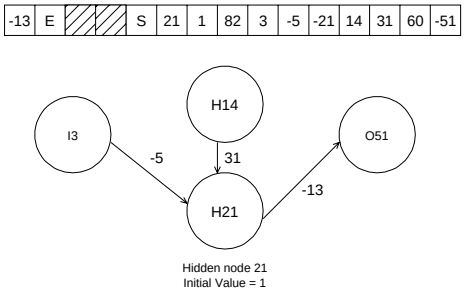

Figure 5: An example node definition in a marker-based gene. The first connection has $key = 82$, $label = 3$, $w = -5$. The key and label are both positive so the connection is to be made from input unit 3.

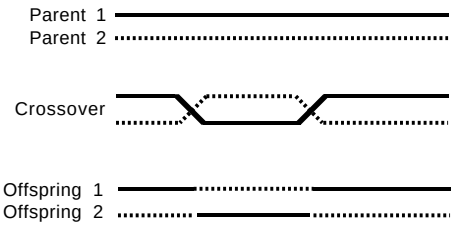

Figure 6: Two point crossover. Each offspring receives the front and rear part of one parent's chromosome and the middle of the other parent's chromosome.

To determine a network's fitness, it was inserted into an α-β search program and played against a full-width, fixed-depth minimax-α-β search. Both players were allowed to search through the second level. To optimize α-β pruning, node ordering was implemented based on the values of the evaluation function (Pearl 1984).

Both players always used the same evaluation func-

tion. One population was evolved based on the positional strategy of Iago (Rosenbloom 1982), one of the first championship-level Othello programs. Such an evaluation function is relatively weak as it only considers the merits of single spaces without taking mobility into account[1]. The goal was to see how well the focus networks could evolve to make use of weak heuristic information, and also to provide enough errors so that the effect of focus networks would be easily seen.

A separate population was evolved using the evaluation function from the Bill program (Lee and Mahajan 1990). Bill's evaluation has been optimized through Bayesian learning and is believed to be one of the best in the world. The goal was to see if the focus networks could achieve any improvement over such an already strong heuristic.

To create different games, an initial state was selected randomly among the 244 possible board positions after four moves. To prevent networks from expecting certain moves, the opponents moved randomly 10% of the time. The random moves also make risk taking a viable option in a losing situation since the opponent will not always make the best move. If the opponent's evaluation function returned the same value for two or more moves, a random selection was made between the equal-valued moves, further discouraging expectations. The number of wins over ten games determined each network's fitness.

Results

The networks were evolved for 1000 generations, which took about four days on a Sun Sparcstation 1. After evolution, the best focus network was again played against the full–width search program, but this time the program made no random moves. The performance was measured by the percentage of games won over all 244 opening games.

In the first test (figure 7), the focused search level was fixed at 2, and the full–width opponent's was varied. As a control, a 2-level, full–width minimax search was also played against the full–width opponent. Note that the focused (shaded bars) and full–width (white bars) searches are not playing against each other, but against another full–width opponent. The results show that a focused search to level 2 appears to be as strong as a full–width search to level 4.

In the second test (figure 8), the focused search level was increased with the full-width opponent's. The control full–width search (white bars) performs consistently at 50% because it is simply playing itself at each level. The results show that the focused search consistently outplays the full–width search even as the search level increases far beyond its training. The performance is strongest at level 2, where the focused network was actually trained, and is otherwise approximately constant at 65%. This result is important

[1]Iago also included a complex mobility strategy.

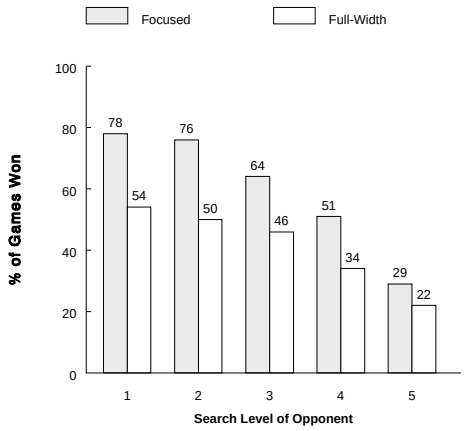

Figure 7: The winning percentage of two level search with and without a focus network against a variable-level full–width opponent.

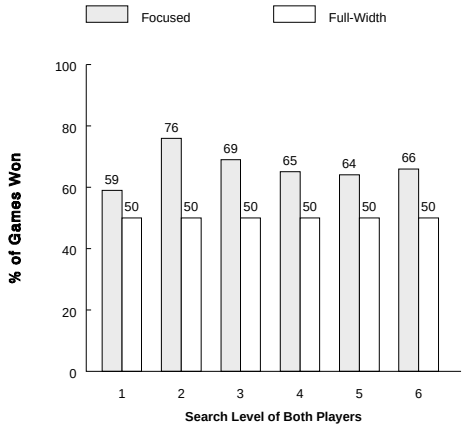

Figure 8: The winning percentage of a variable-level search with and without a focus network against a variable-level full–width opponent.

because it suggests that the focus network could be trained at any level, and would generalize well to other search depths.

It is also important to note that the focused searches were winning while looking at only a subset of the states that the full-width searches are examining. Figure 9 shows the average number of board positions examined per game for each search bound. Of all available legal moves, only 79% were included in the focus window. The full-width search must be receiving poor information from minimax, causing it to choose bad moves. Since the focused search is using the same evaluation function and is searching to the same depth, it appears that the focus network is shielding the root from this misinformation.

To better understand how the stronger play was achieved, the moves included in the focus window were further analyzed. 100 test games were played against a full–width search using the same evaluation function.

	1	2	3	4	5	6
Focus	189	662	3440	12172	63304	230487
Full	226	842	4042	16684	75696	330453

Figure 9: The average number of states examined per game for each depth bound.

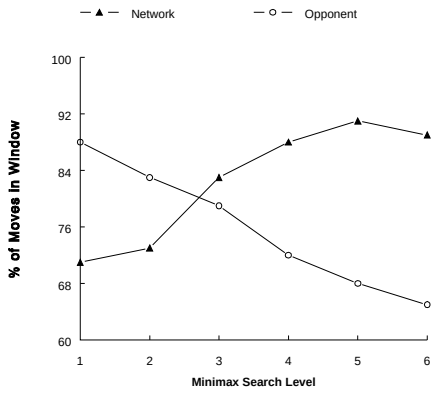

Figure 10: The percentage of moves returned by minimax as its choice that the focus network considers.

At each board position the moves in the focus window were compared with the move a full–width minimax search would return at the same position. Figure 10 shows the percentage of full–width minimax's moves that were included in the focus network's window. The graph thus reflects how often the focus network agrees with full–width minimax. The results show that the focus network is effectively looking far ahead. The moves in the network's window are similar to moves that a deep–searching, full–width minimax would return (black triangles in figure 10). However, since the network has only been evolved against a shallow–searching opponent, its predictions of the opponent's moves become less accurate as the opponent searches deeper (white circles in figure 10). The focus network's moves are strong because they are not tied to the moves that a full–width minimax search would choose. Instead, they reflect moves that have led to wins. It is this strong offense that allows the networks to scale with the search level. It is conceivable that eventually the network's diminishing defense will leave it vulnerable to a powerful opponent, however that was never observed in our experiments.

In the second population, evolved using the evaluation function from Bill, the best focus networks achieved a winning percentage of 51% over the full–width searches to the same level. Apparently, since Bill's evaluation function has very few errors, the focus networks were not able to improve the play very much. However, it is significant that the focused searches achieved this performance while examining only 84% of the moves that full-width Bill evaluated. It seems the focus networks were able to discover and prune unnecessary nodes even with a Bayes-optimized heuristic. In a game playing setting where time constraints must be taken into account, such an improved efficiency translates directly to better performance.

Discussion and Future Work

The results show that better play can be achieved through more selective search. Much like humans, focus networks selectively dismiss moves that have previously led to adverse situations. Whereas full-width minimax is very sensitive to inconsistencies in the evaluation function, focused searches can actually discover and discard unreliable information. The approach will be most useful in improving performance in domains where it is difficult to come up with good evaluation functions. The evolution system can take a weak heuristic and discover how to best use the information it provides. In this sense, the approach is similar to other recent improvements in game playing such as Bayesian optimization of evaluation functions (Hansson and Mayer 1990; Lee and Mahajan 1990). A comparison of these techniques and a study of how they perhaps could be combined would be most interesting.

In an earlier implementation of focus networks, a fixed-size focus window that always included the three best moves was used (Moriarty and Miikkulainen 1994). This strategy achieved performance comparable to the threshold-based window with an even more dramatic reduction in the number of states evaluated. However, the fixed window system was not able to generalize well to better opponents such as Bill. When evolved with Bill's evaluation function, the fixed window pruned too many nodes and performed very poorly. On the other hand, the threshold-based window allows the system to adjust the extent of pruning according to how much reliable information there is in the tree.

It seems to make little difference how deep the system is allowed to search during training (figure 8). The focus networks should therefore perform well in real game-playing situations where the search depth may vary significantly depending on the available time. However, the training opponent's search depth (and evaluation function) may have a significant effect on performance. It might be possible to evolve better play by improving the opponent gradually during training. If the opponent gets stronger as the networks evolve, the networks would have to compensate by improving their defensive strategy, and superior overall play should result.

In our implementation, focus networks searched only through uniform–depth trees. Focus networks could also be implemented with algorithms such as best–first minimax (Korf and Chickering 1994), where the tree is grown in non-uniform depths allowing more promising moves to be searched deeper. Whereas the standard best–first minimax considers all unexplored board positions in the decision of where to explore next, a selective window of the most important positions could

be maintained to focus the search.

Another application of neuro-evolution to game playing is to evolve networks to serve as the evaluation function. Interestingly, the results have been discouraging so far. Whereas the focus networks' output values only need to indicate above or below a threshold, the evaluation networks' output units must reflect an absolute value comparable to other board evaluations. It has proven very difficult for the networks to discover such global values.

While focus networks may be well suited for Othello, their implementation in more complex games like chess is not as straightforward. In our implementation, the output layer represented the entire move space. This is feasible in Othello, since there are only 60 possible moves. It is unrealistic to try to represent the entire move space of a game such as chess in a single output layer. A possible solution is to use two focus networks in the decision process. The first network's output layer would represent each piece and would decide which pieces to consider. The second network's output layer would represent each space on the board (as in the Othello networks). Given the current board and the piece to be moved, the second network could decide which moves of a given piece to consider. Such an extension constitutes a most interesting direction of future research.

Conclusion

Artificial evolution of neural networks is a promising paradigm for developing better search strategies. It is possible to identify unreliable information in the search tree and find ways to avoid it. Focus networks can overcome not only errors in the evaluation function but flaws inherent in minimax itself. Focused searches are cognitively more appealing since they produce more human-like search rather than systematic exhaustive search. In Othello, a focused search consistently outplayed full–width minimax while examining a subset of the moves. Even with a highly sophisticated evaluation function, the focus networks were able to create a more efficient search by pruning irrelevant nodes. Applications to more complex domains are more challenging, but not infeasible.

References

Edwards, D., and Hart, T. (1963). The alpha-beta heuristic. Technical Report 30, MIT.

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. (1988). *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley.

Hansson, O., and Mayer, A. (1989). Heuristic search as evidential reasoning. In *Proceedings of the Fifth Workshop on Uncertainty in AI*.

Hansson, O., and Mayer, A. (1990). Probabilistic heuristic estimates. *Annals of Mathematics and Artificial Intelligence*, 2:209–220.

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.

Knuth, D. E., and Moore, R. W. (1975). An analysis of alpha-beta pruning. *Artificial Intelligence*, 6:293–326.

Korf, R. E. (1988). Search: A survey of recent results. In Shrobe, H. E., editor, *Exploring Artificial Intelligence*. San Mateo, California: Morgan Kaufmann.

Korf, R. E., and Chickering, D. M. (1994). Best-first minimax search: Othello results. In *AAAI-94*.

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

McAllester, D. A. (1988). Conspiracy numbers for minmax search. *Artificial Intelligence*, 35:287–310.

Moriarty, D. E., and Miikkulainen, R. (1993). Evolving complex Othello strategies using marker-based genetic encoding of neural networks. Technical Report AI93-206, Department of Computer Sciences, The University of Texas at Austin.

Moriarty, D. E., and Miikkulainen, R. (1994). Improving game tree search with evolutionary neural networks. In *Proceedings of the First IEEE Conference on Evolutionary Computation*.

Pearl, J. (1984). *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Reading, MA: Addison-Wesley.

Rivest, R. L. (1987). Game tree searching by min/max approximation. *Artificial Intelligence*, 34:77–96.

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

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. In Rumelhart, D. E., and McClelland, J. L., editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*, 318–362. Cambridge, MA: MIT Press.

Shannon, C. E. (1950). Programming a computer for playing chess. *Philisophical Magazine*, 41:256–275.