Machine Learning of Othello Heuristics

William A. Greene
Computer Science Department
University of New Orleans
New Orleans, Louisiana 70148
e-mail wagcs@uno.edu

Abstract: The machine learning algorithm of [3] is applied to the problem of learning which heuristics to apply when playing the board game Othello. The problem is large, for there are 46,875 heuristics considered. The results are respectable; the Learner is able to beat a practiced human player approximately fifty percent of the time. Suggestions for improvement are included.

Introduction

The experimental research reported in this paper takes as its starting point the machine learning algorithm of [3]. In that paper was described an algorithm that learns which are the best paths to follow through a state-space to reach some desirable goal-state. Below we summarize that algorithm. In this paper we apply the algorithm to the complex task of learning which are the best heuristics for playing the board game Othello. The skill level achieved by our Othello-playing program is quite respectable, namely, able to beat a skilled player (in the person of the author) approximately fifty percent of the time. We suggest reasons why the program's achieved skill level does not grow higher than it does.

The "heuristics" our program follows in playing Othello are not really the ones of a rule-based expert system (so, for instance, we do no pure logic programming), but our system is similar to such a system. Moves are chosen based on how well they place the board into a favorable configuration, as measured along featural dimensions such as corner stability and edge stability (described later).

The world of game-playing is an idealized one. It is a world much more completely and clearly defined, and much less messy, than the real world we live in. Game-playing has proved a fruitful field for artificial intelligence research, and the board game Othello has been the testbed for other AI researches. Rosenbloom's work [5] was the first large-scale artificial intelligence experimentation with Othello. More recently Othello has figured into the work of for instance Abramson and Korf [1], DeJong and Schultz [2], and Lee and Mahajan [4].

Othello

The board game Othello is often categorized as being in the GO family of games. Othello is played on an 8-by-8 board of (sub)squares, like chess, but in Othello the squares are not of different colors. There are two players, usually called Black and White, who alternate turns. On their turns the players place disk-shaped playing pieces, colored black or white as appropriate, onto board squares. Initially the board is empty except for the four central squares, of which Black and White each possess two, at opposing corners. On his turn a player places a playing piece of his color onto some square of the board in such a way as to capture one or more squares, or equivalently disks, of his opponent. The captured disks get replaced by those of the capturing player's color. (Usually the playing disks are black on one side and white on the other so they can be flipped over after capture.)

A played piece is invariably placed in an unoccupied square which is immediately adjacent (in one of the eight directions: up, down, left, right, or one of the four diagonal directions) to some square currently occupied by one's opponent. Moreover, to capture along one of the eight directions, there must in that direction be a contiguous sequence of one or more opponent squares, followed immediately by a square of one's own color. Then the squares captured in that direction are the opponent squares so bounded by the newly played piece and the first piece of one's own color.

If a player can capture by playing on a certain square, then he captures opponent squares wherever possible in all eight directions away from that square. If there are several unoccupied squares on which a player can play so as to capture, then he can choose to play on any one of them. A player who cannot play so as to capture forfeits his turn, but remains in the game. A player may not elect to forfeit if it is possible for him to capture.

The game is over when the board is filled or when neither player can capture. The winner is the player with the most occupied squares on the board at game end; tied games are possible.

The Learning Algorithm of [3]

Now we outline the learning algorithm of [3]; that reference should be consulted for complete details. Imag-

ine there is a state-space, some of whose states are goal-states, representing for instance end-of-game. We begin at some state and wend a path towards a goal-state. Some goal-states are more desirable than others and hence some paths are more desirable than others. At any particular step (or state) in the process of building a path, there are several alternative arcs out of the state that we might next follow. Which are the better or best alternative arcs to follow? It is exactly this question's answers that are learned by the algorithm of [3].

There are many learning trials, that is, path wendings. When building the current path we select among competing alternative arcs based on how well the arcs have participated in constructing desirable paths in the past. Each arc has a goodness measure, an integer which at the beginning of the learning trials has the value zero. Having completed the path of the current trial, that path is scored with an integer indicative of the desirability of the goal-state reached. In [3], the algorithm was applied to the simple matchstick game NIM and the score assigned a path was plus or minus one according as the game was won or lost. The path's score is then added to the goodness measure of every arc participating in the path. Thus an arc with a comparatively high goodness measure is one that, compared to competing arcs, has led more frequently to desirable goal-states, or led to more desirable goal-states, or both.

Finally, when building the current path, the algorithm (employing a random number generator) chooses among competing arcs in frequencies which are (more or less) proportional to the ratios of the arcs' associated goodness measures.

In [3] a probabilistic proof was given to show that this algorithm (using the ± 1 path-scoring scheme cited above, anyway) always eventually comes to overwhelming prefer the best arcs.

The "eventually" mentioned in the preceding sentence sometimes amounts to a very long time, even for the simple game of NIM. On the other hand, the experiments of [3] showed empirically that when taught by playing against a semi-skilled NIM-Teacher, or when self-taught by an initially NIM-ignorant Learner playing against a carbon-copy of itself, the NIM-Learner could achieve a very high level of proficiency in a matter of several minutes. Part of the motivation of the current research was to see how well the learning algorithm of [3] scaled up to the complex problem of learning which to prefer among a very large number (46,875) of heuristics for playing Othello.

Othello Strategies

Next we give some strategy truisms about playing Othello. Given the initial board configuration and the rules of the game, the general trend is for the occupied portion of the board to expand outward towards the edges and corners. The most important squares for me to occupy are the four corner squares, for my opponent cannot possibly capture them from me. The corners act as anchors for stabilizing my squares near the corners; for instance, contiguous edge squares I occupy that end in a corner are stable for me (uncapturable). Also stable for me is an isosceles right triangle, anchored at a corner, of squares all of my color. My playing onto an immediate neighbor of an unoccupied corner is risky, since it may give my opponent the chance to take the important corner square.

After the corners, generally the next most important squares are the edge squares. An edge square I possess provides a sort of partial stabilizing anchor for the row or column it ends. Playing onto a square on a file (that is, row or column) adjacent to an edge file provides some opportunity for my opponent to gain edge squares. Playing onto such a file is less desirable than playing onto a file two away from an edge (that is, onto row 3 or 6 or column 3 or 6), for the latter play may eventually obligate my opponent to enter the file nearer the edge.

As the occupied portion of the board expands outward, in general it is good for me to play on or capture squares two away from the corners (for the (1,1) corner, we have in the mind the three squares (1,3), (3,3), and (3,1)).

Some Othello-playing programs exploit the above types of considerations by assigning weights to squares according to their position and choosing on each turn to play on a square of best weight. Othello-playing programs that use a pure weighted-squares strategy are easily beaten by experienced human players.

Architecture of Othello Learning

The play of our Othello Learner can be described in overview as follows. Of the 64 board squares, 4 are initially occupied, so a game may take up to 60 turns before ending. Our program divides the game into five phases. On phase one the Learner plays somewhat by rote, hugging the center of the board. At the start of the last phase, which comprises the last, say, 8 or 10 turns, a game tree to end-of-game is constructed and then it is traversed by the Learner for finishing out the game. It is during the intermediate three phases that heuristics are followed in choosing moves. Thus, heuristical play during mid-game leads, it is to be hoped, to an advantageous board configuration from which the Learner can fashion a win by looking ahead during the final phase.

The first phase, phase one, covers the first 12 turns (of the two players together). During this phase the Learner uses a weighted-squares approach that has it attempt to confine itself to the 12 squares neighboring the four initially occupied ones, and in the process to attempt to occupy the "corners" (3,3), (3,6), (6,3), and (6,6), but this phase's strategy will also have it take the true corner squares (1,1), (1,8), (8,1), and (8,8) and edge squares if that becomes possible.

As said earlier, at the start of the last phase, phase five, the Learner builds a game tree to end-of-game. Tree construction is a variant on the mini-max procedure with

pruning. At Learner tree nodes, maximizing is done; at opponent nodes, minimizing. At end-of-game nodes, the static evaluation function is, of course, the number of squares occupied by the Learner minus those of its opponent. In the tree the Learner retains only its optimal moves, but retains all responses the opponent can make to those optimal moves. The rationale is obvious: the Learner will always be choosing moves it knows to be optimal for itself, but the Learner cannot know which response, optimal or otherwise, the opponent will counter with. Tree construction has the opponent do pruning: when a tree node corresponding to an opponent's turn achieves an extremum (minimum) less than its parent's extremum (maximum), the opponent abandons examination of remaining moves available to it. Thus, tree construction has the Learner practice space conservation, and the opponent, time conservation.

During the remainder of phase five the Learner plays by tracking down the tree. If on its turn there are several equally good moves for the Learner (in that they all can result in the same square difference at game's end), then one among them is chosen at random (this to introduce an element of unpredictability when the opponent is a human playing an interactive game against the Learner).

The Othello Heuristics

Now we describe the heuristics employed in the middle three phases of game play. Together these phases account for the approximately 40 middle turns taken by the two players. The phases simply divide this group of turns into thirds, the notion being that (for example) edge stability and corner stability may have different relative importances in the different phases.

A move available to the Learner results in a new board configuration, whose advantageousness to the Learner is measured along six parameters, and they are: (1) corner strength and potential, (2) corner stability, (3) edge stability, (4) interior stability, (5) mobility, and (6) square advantage. These measures are fully described below. Some of these notions for measures we borrowed from Rosenbloom [5], and some are of our own invention.

Calculation of all six measures follows the same general pattern: edge stability (for example) is given an integer rating both before and after the move, then the ratings are subtracted; this difference is then an integer rating of the change (improvement or decline) in edge stability. To keep space costs tractable, this integer difference is converted to a value in the small linear type <VeryLow, Low, Middling, High, VeryHigh>. Each move is thus associated with a 7-tuple, (1) game phase, in the range 2..4, (2) corner strength and potential, in the range VeryLow..VeryHigh, ..., and (7) square advantage, in the range VeryLow..VeryHigh. The 3 * 5**6 = 46,875 such 7-tuples are, in fact, our program's different "heuristics". (Note that two moves available to the Learner may get associated with the same heuristic.) Each heuristic has an associated "WinMeasure", which is an integer that tallies how well moves of such heuristical measurement have in the past participated in games of favorable conclusion for the Learner.

In the NIM-playing program of [3], (analogous) WinMeasure's were changed by adding plus or minus one according as the NIM game was won or lost. For the Othello program of this research, each heuristic employed in the chain of Learner turns on the current game gets its reward or punishment by adding to its WinMeasure the amount D = the difference in squares possessed by the Learner and opponent at end-of-game. (To win by many squares is a better win. In fact, in Othello competition among humans, handicapping is employed, such as: player A must beat player B by at least 10 squares.)

Of the several moves available to the Learner on a particular turn, the Learner chooses a move whose associated heuristic has a high WinMeasure. Actually, a heuristic with a high WinMeasure is first chosen, and then one of the moves associated with it is chosen. Specifically, as in [3], if the heuristics at hand are $\{h_i\}$ and have respective WinMeasures $\{WM_i\}$ then the k-th heuristic h_k is chosen with a likelihood of $WM_k / (\Sigma_i WM_i)$. (If need be we first translate the set of WinMeasures into a range of positive integers.) All the moves associated with the chosen heuristic might outwardly be taken to be equally good, but we in fact do not choose one at random. Instead, our algorithm, during phase 2 for instance, chooses the associated move with the highest integer change in corner strength and potential. The reason for choosing corner strength and potential as the phase 2 "tie breaker" was: it was this feature that most tended to exhibit VeryHigh values among phase 2 heuristics with large positive WinMeasures and exhibit VeryLow values among phase 2 heuristics with large negative WinMeasures. Similarly, on phases 3 and 4, it is the move (among those associated with the chosen heuristic) with the highest integer change in, respectively, edge stability and corner stability that is the one the Learner chooses.

Our use of the term heuristic is not inappropriate. If one of our heuristics has achieved a high WinMeasure then a move M associated with it has a good chance of being chosen, and consequently play proceeds as though the Learner had reasoned

IF it is phase 2
  AND move M results in a VeryHigh change in edge stability
  AND move M results in a Middling change in mobility
  AND ....
THEN give considerable likelihood to choosing move M

The 6 Board Ratings

Now we describe the six board features. Actually what we shall describe are the six integer-valued ratings given to a board. (Recall, the board will be given an integer rating before a potential Learner move, the board resulting

from the move will be re-rated, ultimately the ratings will be subtracted and the integer difference converted to a value in the range VeryLow..VeryHigh.)

Corner strength and potential is a sum of strength-and-potential assessments over each of the four corners. It is intended to be helpful for steering play during earlier phases when the corner regions are thinly filled. What is measured is the degree to which a corner region is already advantageous or offers positive opportunities or poses risks to the Learner. The assessment adds and subtracts weights devised from the author's own experience with Othello, and is somewhat ad hoc.

Corner stability is a sum of stability ratings over the four corner regions. The algorithm starts at a corner and considers the sequence of increasingly longer diagonal lines which are "perpendicular to" the corner and which progress towards the board's center. For a diagonal line, the algorithm counts the number of contiguous Learner squares which start at an edge and extend inwards, possibly all the way to the next edge. If the Learner squares do extend to the next edge, then the algorithm proceeds on to the next diagonal, else it stops its examination of this corner region. Essentially what is counted is the number of squares in the largest isosceles right triangle anchored at the corner which is entirely occupied by Learner squares. All the squares counted are stable for the Learner.

To describe edge stability and interior stability, we first must describe line stability. Imagine a line of eight board squares, each of which is either black or white or unoccupied. A typical such line is a board edge, though as will be seen we let for example an arbitrary board row also be abstracted as such a line. We solely consider future moves onto unoccupied squares of this line. An occupied black square is categorized as being either unstable-for-black (can be captured by some very next move onto this line) or semistable-for-black (not capturable on the very next play but is capturable under some combination of future moves, some of which may be Black's) or stable-for-black (cannot be captured by any combination of future moves onto this line). Occupied white squares are similarly categorized. The term line stability is a bit of a misnomer; it is not the stability of the line but rather that of the occupied squares on it that interests us.

Line stabilities are not computed on the fly; instead, all lines' stabilities are computed at program start-up time and saved in an array (with 3**8 = 6561 components). The algorithm for doing so uses dynamic programming: assuming we have already computed the stability of every line having N unoccupied squares, then we can easily compute the stability of a line with N+1 unoccupied squares.

The computation for the edge stability rating is now easily described, and is the sum of measurements for the four edges. We use our line stability table. For each edge the measure is 10*A + 3*B + C, where A is the number of squares stable for the Learner minus the number stable for the opponent, and B and C are the analogous differences for semistable and unstable. The coefficients 10, 3, and 1 are somewhat ad hoc.

We note in passing that corner possession is rewarded by each of the three measurements described so far.

The procedures for computing corner and edge stability carry an extra parameter, a "stability board" whose squares get marked as stable for the Learner when they are found to be so. The procedure for computing interior stability continues to mark this board on its interior squares. The interior squares are traversed in an inward spiral: first row 2's positions 2 to 6, then column 7's positions 2 to 6, then row 7's positions 7 to 3, then column 2's position's 7 to 3, then row 3's positions 3 to 5, then column 6's positions 3 to 5, and so on into the center of the board. For each interior square in the spiral, our reasoning, only heuristic, is as follows. If in each of the four bi-directions (horizontal, vertical, and the two diagonal ones) an interior square S possessed by the Learner has an immediate neighbor which is stable for the Learner, then we assert that S is stable for the Learner by so marking the stability board. Barring satisfaction of the antecedent of the preceding sentence, we exploit the line stability table as follows. If the square S is stable (versus semistable or unstable) for the Learner when viewed as an element of the four lines that are the row, the column, and the two (stretched) diagonals containing S, then we assert that S is stable for the Learner by so marking the stability board. With respect to the preceding sentence, we are stymied (unable to assert that S is stable for the Learner) if a diagonal is not stretchable to a full line of 8 squares. Stretchable by definition means that one of the diagonal's ending edge squares is occupied (by either player) and then stretching consists of padding with the color of that ending edge square.

Finally, the interior stability rating is the number of interior squares marked stable for the Learner.

By a player's immediate mobility we mean the number of squares he can now play upon, that is, capture from. By his future mobility we give the meaning the number of unoccupied squares which are not now playable upon but at least do neighbor one of his opponent's squares. By his mobility we give the meaning 2 * immediate-mobility + future-mobility. The coefficients 2 and 1 are somewhat ad hoc.

Let us note that in taking the differences in integer board ratings across a potential Learner move, for the mobility feature we make an exception and use the Learner's mobility before the move but the opponent's mobility after the move. The difference is then the exchange in the players' mobilities.

The square advantage rating given a board is simply the number of squares possessed by the Learner minus that by the opponent. (The difference in ratings across a potential Learner move is easily seen to equal 1 + 2*C where C is the number of squares captured on the move.)

More about Architecture

Within our computer program for playing Othello there are three agents who play the game, namely, the Learner, the Teacher, and the Human. The algorithmic behavior of the Learner has already been described. The Teacher provides practical instruction in Othello-playing by playing games against the Learner in so-called "learning bursts", consisting perhaps of 500 or 5000 consecutive games. Presumably the Teacher provides play which is to some degree informed; the Teacher might for example play by a weighted squares strategy. Alternatively, in the current version of our program the Teacher is a carbon copy of the Learner and plays by accessing (reading and updating) the same database of 46,875 heuristics and their WinMeasures as is accessed by the Learner. We call this self-taught learning, for in effect the Learner is playing against himself in order to learn by experience which are the better moves.

The Human participates interactively. Typically the human dictates that a learning burst take place, then plays several interactive games against the Learner, then dictates a further learning burst, and so on.

For the database of the 46,875 heuristics and their WinMeasures there are several options available. The WinMeasures can be initialized to zero. Or they can be preweighted so that for instance a large WinMeasure is initially attributed to any heuristic containing many VeryHigh's in its 7-tuple of values. A third alternative is that the WinMeasures can be read in from an external file; at the end of a session they can also be written to a file. Using external files it is practical to let learning experiments stretch over days.

Experimental Results

Next we describe the results of Othello learning experiments. Our work went through a number of versions and variants. Our initial approach was to have the Teacher play by a weighted-squares strategy, but the game skill thus acquired by the Learner was noticeably weak, and so we abandoned such a Teacher in favor of one that was a carbon copy of the Learner (self-taught learning). With respect to initial knowledge, we tried three approaches: all WinMeasures start at initial value zero, WinMeasures are initially pre-weighted, and thirdly, the author playing interactively starts off the learning with 25 games from a practiced player.

Our computer program was written in Ada, reached a length of 5000 lines, and was run on a VAX-11/8600. For the various approaches, many games -- up to 50,000 and beyond -- were played. In general, CPU time on the order of 12 hours was required to play 10,000 learning-burst games.

In our best results, described later, the Learner was able to beat the author in better than 60% of the games in a small sample of games.

For the various versions, examination of WinMeasures and the moves associated with them consistently showed evidence that learning had taken place. Recall that a heuristic is a 7-tuple whose first component (game phase) is in the range 2..4, and whose other six (board features) are in the range VeryLow..VeryHigh. Consistently, 3 or 4 heuristics containing all or nearly all VeryHigh's achieved very large WinMeasures. (Later we suggest a reason why more such "good" heuristics did not rise like cream to the comparative top.) Also, risky moves onto corner-endangering squares like (1,2) and (2,2) were observed being associated with heuristics having very negative WinMeasures. Additionally, it was observed that the Learner played towards a "cross endgame" whereby the four squares around each of the four corners are vacant, those regions are entirely bordered by Learner squares, and the rest of the board is filled. There are 16 vacant squares altogether on such a board (so the lookahead has not yet taken place), and necessarily the Learner will take at least one corner and probably several.

Curiously, for each of our three choices for initial knowledge (WinMeasures are initially zero, are preweighted, or thirdly result from 25 games against the author), it was seen that even after many games only relatively few heuristics actually got used, as demonstrated by the fact that their WinMeasures had changed from their initial values. Typically, only about 700 (resp., 2000, 4500) of the heuristics for phase 2 (resp., phases 3 and 4) had actually been used even after 50,000 games.

Our Learner will never play perfect Othello, because of its subalgorithm for selecting heuristics and moves; note that every heuristic retains some likelihood of being chosen. This is a mixed blessing. On the positive side, a good heuristic cannot be permanently blackballed due to some applications of it during early losing games. On the negative side, poor heuristics always have some likelihood, even if small, of being chosen.

Invariably our Learner never learned adequate aggressiveness in corner capturing; perhaps this is because only four corner captures can occur in a game. Similarly, the Learner too often made plays that sacrificed corners to the author. We attempted to solve these problems with an extra procedure that warped ratings so as to prejudice the selection of heuristics when corner play was involved, but were unsuccessful. We believe a distinct improvement would be a front-end to move selection that first tries to take corners and avoid jeopardizing corners, and barring that to make move selection as originally described.

To chart the course of learning we scheduled a learning burst of 10,000 games between the Learner and Teacher, and then the author played ten or so games against the Learner. This was followed by a further learning burst to 20,000 games, then another ten or so games against the author, and so on. Playing ten or so games against the author provides only a small sample of Learner achievement, but no other alternative was available to the author.

Our best results, in which the Learner beat the author on 7 of 11 games, arose in two different ways.

One of the two "best results" arose by pre-weighting the WinMeasures, followed by only 10,000 learning burst games. Pre-weighting was achieved by adding -500, -100, 0, 100, 500, resp., for each occurrence of VeryLow, Low, Middling, High, VeryHigh, resp., in the heuristic at hand (the maximum pre-weight is 3000). Under further learning beyond the 10,000 game mark, the Learner's success against the author declined to winning only 40 to 50 percent of the ten or so games. We hypothesize the following explanation. There are good moves which arise only relatively rarely, such as taking corners, or moves that achieve very high improvements in interior stability. After only 10,000 games the pre-weights are still large enough, comparatively, to enable selection of such moves when they are available. After many more learning games the WinMeasures of comparatively weaker but more frequently arising heuristics have grown very large and the better moves get swamped.

We believe this same analysis explains why, for most of the other variants and approaches we explored, even after 50,000 or 60,000 games the Learner's success rate against the author was stuck in the 40 to 50 percent range.

The second of our best results (Learner wins 7 of 11 games against the author) was achieved by deepening the lookahead on those 11 games. For all our versions, for time reasons the Learner's lookahead during learning bursts was not made until only 8 vacant squares remained. We achieved the second of our best results under the following scenario. WinMeasures were initialized to zero, and 60,000 learning burst games were played. Then for playing against the author, the Learner's lookahead was made when 10 vacant squares remained. For timed competitive play (timed in the sense that like competitive chess both adversaries have a fixed quantity of time for choosing the totality of their moves in one game), probably the lookahead could be inched up another ply or two.

Further Remarks and Observations

About our experiments we make the following general remarks and observations. Initializing WinMeasures via 25 games against an experienced player seems to provide too little information to make this approach have noticeably different results from the other approaches.

As opposed to the difference in board goodness before and after *one* potential Learner move, we also investigated a 2-ply approach in which an attempt was made to incorporate some consideration for the opponent's most aggressive response, but this proved much too time-expensive.

Phase 4 "good" heuristics developed large WinMeasures much more rapidly than those of phases 2 and 3. This seems to illustrate the propagation of skill backwards from game-end towards game-start that is mentioned in [3]. Moveover, the highly favored phase 2 and 3 heuristics were in fact not so "good". Perhaps an improvement is, after a certain amount of learning, to re-initialize WinMeasures for phases 2 and 3 and re-start the learning for those phases.

Curiously, playing against the carbon-copy Teacher, the Learner consistently won noticeably more often by playing second than by playing first. The reason for this is unexplained. Perhaps playing second let the Learner more often look ahead 8 moves as opposed to 7. Perhaps in Othello there is an inherent advantage to playing second.

Inspection of heuristics and their WinMeasures suggests that the aspect of mobility does not have the great significance to mid-game play that Rosenbloom [5] attributes to it.

Concluding Remarks

We cannot claim our Learner was extravagantly successful in acquiring skill at Othello, but we do feel the degree of learning was quite respectable. The Learner did not invent its own heuristics, rather it weighed among those we had provided. The quality of the provided heuristics would seem to place an inevitable upper bound on the game skill acquired -- if the heuristics are suboptimal, would not even their judicious application still result in suboptimal play?

We have marked our Learner's successes and weaknesses, and suggested explanations and corrections for some of the latter. Ours was a large problem -- 46,875 is a large number of heuristics. The heuristics in a production expert system typically number only in the hundreds. We retain the belief that the learning algorithm of [3] can be successfully used to reveal which are the better heuristics to line up for steering a path to success in, say, an expert system for managing a crisis in a nuclear power plant, as suggested in [3].

References

1. Abramson, Bruce, and Korf, Richard E., "A model of two-player evaluation functions", *Proceedings of the Sixth National Conference on Artificial Intelligence*, Seattle, WA, 1987, pp. 90-94.

2. DeJong, Kenneth, and Schultz, Alan, "Using experience-based learning in game playing", *Proceedings of the Fifth International Conference on Machine Learning*, Ann Arbor, MI, 1988, pp. 284-290.

3. Greene, William A., "A learning algorithm that always learns best alternatives", *Proceedings of IEEE SouthEastCon-90*, April 1990, New Orleans, LA, pp. 194-198.

4. Lee, Kai-Fu, and Mahajan, Sanjoy, "The development of a world class Othello program", *Artificial Intelligence* 43 (1990), pp. 21-36.

5. Rosenbloom, Paul S., "A world-championship-level Othello program", *Artificial Intelligence* 19 (1982), pp. 279-320.