Evolving Neural Networks to Play Checkers Without Relying on Expert Knowledge

Kumar Chellapilla and David B. Fogel, *Fellow, IEEE*

Abstract— **An experiment was conducted where neural networks compete for survival in an evolving population based on their ability to play checkers. More specifically, multilayer feedforward neural networks were used to evaluate alternative board positions and games were played using a minimax search strategy. At each generation, the extant neural networks were paired in competitions and selection was used to eliminate those that performed poorly relative to other networks. Offspring neural networks were created from the survivors using random variation of all weights and bias terms. After a series of 250 generations, the best-evolved neural network was played against human opponents in a series of 90 games on an internet website. The neural network was able to defeat two expert-level players and played to a draw against a master. The final rating of the neural network placed it in the "Class A" category using a standard rating system. Of particular importance in the design of the experiment was the fact that no features beyond the piece differential were given to the neural networks as *a priori* knowledge. The process of evolution was able to extract all of the additional information required to play at this level of competency. It accomplished this based almost solely on the feedback offered in the final aggregated outcome of each game played (i.e., win, lose, or draw). This procedure stands in marked contrast to the typical artifice of explicitly injecting expert knowledge into a game-playing program.**

Index Terms—**Alpha–beta search, checkers, evolutionary computation, feedforward neural networks, game playing.**

I. Introduction

THERE has been interest in designing computer algorithms to play common games since the early advent of the modern digital computer. Chess has received the most attention in this regard, with efforts to beat the human world champion finally being successful in 1997 (Deep Blue defeated Garry Kasparov). Other games have also been tackled, including othello, backgammon, and checkers. With a few exceptions (e.g., [13]), these efforts have relied on domain-specific information programmed into an algorithm in the form of weighted features that were believed to be important for assessing the relative worth of alternative positions in the game. That is, the programs relied on human expertise to defeat human expertise.

Although the accomplishment of defeating a human world champion in any significant game of strategy is a worthy goal, the majority of these "successes" did not incorporate any learning in their algorithms. Every "item" of knowledge was preprogrammed. In some cases, the programs were even tuned to defeat particular human opponents, indicating their brittle nature. The fact that they require human expertise *a priori* is testament to the limitation of this approach.

In contrast, Fogel [1] offered experiments where evolution was used to design neural networks that were capable of playing tic-tac-toe without incorporating features prescribed by experts. The neural networks competed against an expert system, but their overall quality of play was judged solely on the basis of their win, loss, and draw performance over a series of 32 games. No effort was made to assign credit to the evolving networks for any specific move or board feature. The results indicated that successful strategies for this simple game could be developed even without prescribing this information.

A more significant challenge lies in having an evolutionary algorithm learn competent strategies in a complex setting in the absence of a knowledgeable, hand-crafted (i.e., human-designed) opponent. To address this concern, consider the problem of designing an evolutionary algorithm that improves the strategy of play in the game of checkers (also known as draughts, pronounced "drafts") simply by playing successive games between candidate strategies in a population, selecting those that perform well relative to others in the population, making random variations to those strategies that are selected, and iterating this process. Following the previous experiments using the game of tic-tac-toe, strategies can be represented by neural networks. Before providing the algorithmic details, however, the game will be described here for completeness.

II. Background

A. Rules of the Game

Checkers is played traditionally on an eight by eight board with squares of alternative colors of red and black (see Fig. 1). There are two players, denoted as "red" and "white" (or "black" and "white," but here, for consistency with a commonly available website on the internet that allows for competitive play between players who log in, the notation will remain with red and white). Each side has 12 pieces (checkers) which begin in the 12 alternating squares of the same color that are closest to that player's side, with the rightmost square on the closest row to player being left open. The red player moves first and then play alternates between sides. Checkers are allowed to move forward diagonally one square at a time, or, when next to an opposing checker and there is a space available directly behind that opposing checker, by jumping diagonally over an opposing checker. In the latter case, the opposing checker is removed from play. If a jump

Manuscript received February 11, 1999; revised May 17, 1999.

K. Chellapilla is with the Department of Electrical and Computer Engineering, University of California at San Diego, La Jolla, CA 92093 USA.

D. B. Fogel is with Natural Selection, Inc., La Jolla, CA 92037 USA.

Publisher Item Identifier S 1045-9227(99)08827-X.

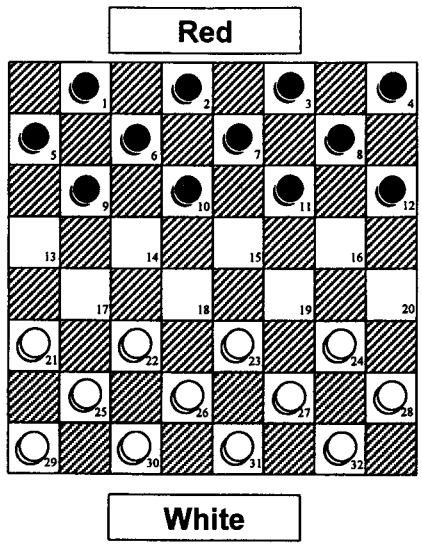

Fig. 1. The starting position for the game of checkers. Pieces move forward diagonally toward the opposing side of the board. Any piece that reaches the opposing far side becomes a "king" and can thereafter move forward or backward diagonally.

would in turn place the jumping checker in position for another jump, that jump must also be played, and so forth, until no further jumps are available for that piece. Whenever a jump is available, it must be played in preference to a move that does not jump; however, when multiple jump moves are available, the player has the choice of which jump to conduct, even when one jump offers the removal of more opponent's pieces (e.g., a double jump versus a single jump). When a checker advances to the last row of the board it becomes a king, and can thereafter move diagonally in any direction (i.e., forward or backward). The game ends when a player has no more available moves, which most often occurs by having their last piece removed from the board but can also occur when all existing pieces are trapped resulting in a loss for the player with no remaining moves and win for the opponent (the object of the game). The game can also end when one side offers a draw and the other accepts.[1]

Unlike tic-tac-toe and many other simpler games, there is no known "value" of the game of checkers. That is, it is not known if the player who moves first can force a win or a draw. The number of possible combinations of board positions is over 5×10^{20} [2, p. 43], and the game tree of possible sequences of moves remains too large to enumerate. Endgame positions with up to eight pieces remaining on the board have been enumerated and incorporated into some checkers-playing computer programs as look-up tables to determine exactly which moves are best (as well as the ultimate outcome) under these conditions (e.g., in the program Chinook [3]). The number of positions with up to eight pieces is about 440 billion. The number of positions increases rapidly with the number of pieces as a combinatorial function thereby making an exhaustive listing of longer endgame sequences impractical.

[1] The game can also end in other ways: 1) by resignation, 2) a draw may be declared when no advancement in position is made in 40 moves by a player who holds an advantage, subject to the discretion of an external third party, and if in match play 3) a player can be forced to resign if they run out of time, which is usually limited to 60 min for the first 30 moves, with an additional 60 min being allotted for the next 30 moves, and so forth.

B. Computer Algorithms for Checkers

There have been many attempts to design programs to play checkers since the late 1950's. These are documented in [2] and for the sake of space only two will be described here. The current computer world champion checkers program is Chinook, designed by Schaeffer *et al.* at University of Alberta. The program uses a linear handcrafted evaluation function that considers several features of the game board including: 1) piece count; 2) kings count; 3) trapped kings; 4) turn; 5) runaway checkers (unimpeded path to a king); and other minor factors [2, pp. 63–65]. In addition, the program has: 1) access to a library of opening moves from games played by grand masters; 2) the complete endgame database for all boards with eight or fewer pieces; and 3) a "fudge factor" that was chosen to favor boards with more pieces than fewer pieces. This last facet was included to present more complicated positions to human opponents in the hopes of eliciting a mistake. No machine learning methods have been employed successfully in the development of Chinook. All of its "knowledge" has been programmed by humans. Chinook played to a draw after six games in a 40-game match against the former human world champion Marion Tinsley, the best player to have lived.[2] Tinsley retired the match after these six games for health reasons and died shortly thereafter. Chinook is rated at 2814, well beyond the ratings of 2632 and 2625 held by the current best human players [2, p. 447] (see the results section below for a more detailed explanation of the ratings systems).

In contrast to Chinook, the most well-known effort in designing an algorithm to play checkers is owed to Samuel [4] which was one of the first apparently successful experiments in machine learning. The method relied in part on the use of a polynomial evaluation function comprising a subset of weighted features chosen from a larger list of possibilities. The polynomial was used to evaluate alternative board positions some number of moves into the future using a minimax strategy. The technique relied on an innovative self-learning procedure whereby one player competed against other. The loser was replaced with a deterministic variant of the winner by altering the weights on the features that were used, or in some cases replacing features that had very low weight with other features. Samuel's program, which also included rote learning of games played by masters, was played against and defeated R. W. Nealey in 1962. IBM Research News described Nealey as "a former Connecticut checkers champion, and one of the nation's foremost players."

The success of Samuel [4] was overstated, and continues to be overstated. The 1962 match against Nealey was, in retrospect, pocked with errors on both sides, as has been demonstrated by using Chinook to analyze the game [2, p. 93–97]. Moreover, Nealey was not a "former Connecticut champion" as advertised at the time of the match, although he did earn this title later. Nealey defeated Samuel's program

[2] From 1950 until his death in 1995, Tinsley won every tournament in which he played and lost only three games.

in a rematch the next year, and Samuel played four games with his program against both the world champion and challenger in 1966, losing all eight games. As Schaeffer [2, p. 97] wrote: "The promise of the 1962 Nealey game was an illusion."

The promise of machine learning methods for addressing complicated games, however, is not an illusion. Reinforcement and evolutionary learning methods have been employed with success in backgammon [5], [6], particularly when using neural networks to determine the appropriate moves. Of particular interest is the possibility of using neural networks to extract nonlinear structure inherent in a problem domain [7]. More specifically, a significant challenge is to devise a method for having neural networks learn how to play a game such as checkers without being given expert knowledge in the form of weighted features, prior games from masters, look-up tables of enumerated end game positions, or other similar information. This would appear to be a necessary precursor to any effort to generate machine intelligence that is capable of solving new problems in new ways. The measure of success is the level of play that can be attained against humans without preprogramming in the requisite knowledge to play well.

Early speculation on the potential success of just these sorts of efforts was entirely negative. Newell in [8] offered "It is extremely doubtful whether there is enough information in 'win, lose, or draw' when referred to the whole play of the game to permit any learning at all over available time scales." Minsky [8] noted being in "complete agreement" with Newell. To test the veracity of these conjectures, an experiment in evolutionary algorithms was designed whereby neural networks that represent strategies for playing checkers were competed against themselves starting from completely random initializations. Points were assigned for achieving a win, loss, or draw in each of a series of games. Only the total point score was used to represent the quality of a neural network's play (i.e., no credit assignment was employed, even to the level of identifying which games were won, lost, or tied). Those networks with the highest scores were maintained as parents for the next generation. Offspring networks were created by randomly varying the connection weights of their parents, and the process was iterated. After 250 generations, the best-evolved neural network was tested against a range of human players. In contrast to the comments by Newell and Minsky, the network not only rose to a reasonably high level of play, it was able to occasionally defeat expert players and played to a draw against a master.

III. Method

The following protocol was adopted for evolving strategies in the game of checkers. Each board was represented by a vector of length 32, with each component corresponding to an available position on the board. Components in the vector could take on elements from $\{-K, -1, 0, +1, +K\}$, where K was the value assigned for a king, "1" was the value for a regular checker, and "0" represented an empty square. The sign of the value indicated whether or not the piece in question belonged to the player (positive) or the opponent (negative). A player's move was determined by evaluating the

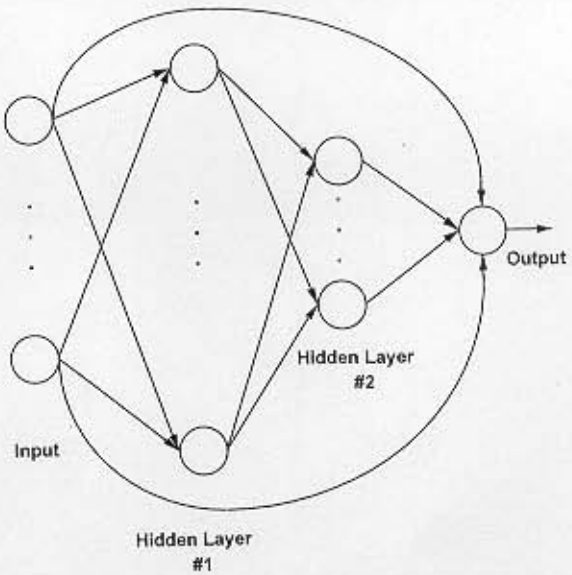

Fig. 2. The neural network topology chosen for the evolutionary checkers experiments. The networks have 32 input nodes corresponding to the 32 possible positions on the board. The two hidden layers comprise 40 and ten hidden nodes, respectively. All input nodes are connected directly to the output node with a weight of 1.0. Bias terms affect each hidden and output node, but are not shown.

presumed quality of potential future positions. This evaluation function was structured as a fully connected feedforward neural network with an input layer, two hidden layers, and an output node. The nonlinearity function at each hidden and output node was chosen to be the hyperbolic tangent (tanh, bounded by ± 1) with a variable bias term, although other sigmoidal functions could undoubtedly have been chosen. In addition, the sum of all entries in the input vector was supplied directly to the output node. Fig. 2 shows the general structure of the network. At each generation, a player was defined by their associated neural network in which all of the connection weights (and biases) were evolvable, as well as their evolvable king value. For all experiments offered here, each network comprised 40 nodes in the first hidden layer and ten nodes in the second layer.[3]

It is important to note that, with one exception, no attempt was made to offer useful features as inputs to a player's neural network. The common approach to designing superior game playing programs is to perform exactly this sort of intervention where a human expert delineates a series of board patterns or general features that are weighted in importance, positively or negatively [3], [9], [10]. In addition, as with Chinook, entire opening sequences from games played by grand masters and look-up tables of end game positions can also be stored in memory and retrieved when appropriate. This is exactly opposite of the approach taken here. The only feature that could be claimed to have been offered is a function of the piece differential between a player and its opponent, owing to the

[3]These values were chosen after initial experiments with ten and eight nodes in each hidden layer gave modestly encouraging results and no further tuning of the number of nodes was undertaken. No claim of optimality is offered for the design chosen, and indeed the result that reasonable levels of play can be achieved without tuning the neural structure is one of the main points to be made here.

sum of the inputs being supplied directly to the output node. The output essentially sums all the inputs which in turn offers the piece advantage or disadvantage. But this is not true in general, for when kings are present on the board, the value K or $-K$ is used in the summation, and as described below, this value was evolvable rather than prescribed by the programmers *a priori*. Thus the evolutionary algorithm had the potential to override the piece differential and invent a new feature in its place. Absolutely no other explicit or implicit features of the board beyond the location of each piece were implemented.

When a board was presented to a neural network for evaluation, the output node designated a scalar value that was interpreted as the worth of that board from the position of the player whose pieces were denoted by positive values. The closer the output value was to 1.0, the better the evaluation of the corresponding input board. Similarly, the closer the output was to -1.0, the worse the board. All positions that were wins for the player (e.g., no remaining opposing pieces) were assigned the value of exactly 1.0 and likewise all positions that were losses were assigned the value of exactly -1.0.

To begin the evolutionary algorithm, a population of 15 strategies (neural networks), P_i, $i = 1, \cdots, 15$, defined by the weights and biases for each neural network and the strategy's associated value of K, was created at random. Weights and biases were generated by sampling from a uniform distribution over $[-0.2, 0.2]$, with the value of K set initially to 2.0. Each strategy had an associated self-adaptive parameter vector σ_i, $i = 1, \cdots, 15$, where each component corresponded to a weight or bias and served to control the step size of the search for new mutated parameters of the neural network. To be consistent with the range of initialization, the self-adaptive parameters for weights and biases were set initially to 0.05.

Each parent generated an offspring strategy by varying all of the associated weights and biases, and possibly the value of K as well. Specifically, for each parent P_i, $i = 1, \cdots, 15$ an offspring P_i', $i = 1, \cdots, 15$, was created by

$$\sigma_i'(j) = \sigma_i(j)\exp(\tau N(0,1)), \qquad j = 1, \cdots, N_w$$
$$w_i'(j) = w_i(j) + \sigma_i'(j)N_j(0,1), \qquad j = 1, \cdots, N_w$$

where N_w is the number of weights and biases in the neural network (here this is 1741), $\tau = 1/\sqrt{2\sqrt{N_w}} = 0.1095$, and $N_j(0,1)$ is a standard Gaussian random variable resampled for every j. The offspring king value K' was obtained by

$$K_i' = K_i \exp((1/\sqrt{2})N(0,1)).$$

For convenience, the value of K_i' was constrained to lie in [1.0, 3.0] by resetting to the limit exceeded when applicable. Thus the offspring's king value had the possibility of taking on any real value in [1.0, 3.0].

All parents and their offspring competed for survival by playing games of checkers and receiving points for their resulting play. Each player in turn played one checkers game against each of five randomly selected opponents from the population (with replacement). In each of these five games, the player always played red, whereas the randomly selected opponent always played white. In each game, the player scored -2, 0, or +1 points depending on whether it lost, drew, or won the game, respectively (a draw was declared after 100 moves for each side). Similarly, each of the opponents also scored -2, 0, or +1 points depending on the outcome. These values were somewhat arbitrary, but reflected a generally reasonable protocol of having a loss be twice as costly as a win was beneficial.[4] In total, there were 150 games per generation, with each strategy participating in an average of ten games. After all games were complete, the 15 strategies that received the greatest total points were retained as parents for the next generation and the process was iterated.

Each game was played using a fail-soft alpha–beta search [11] of the associated game tree for each board position looking a selected number of moves into the future. The minimax move for a given ply was determined by selecting the available move that affords the opponent the opportunity to do the least damage as determined by the evaluation function on the resulting position. The depth of the search, d, was set at four to allow for reasonable execution times (100 generations on a 400-MHz Pentium II required about two days, although no serious attempt was made to optimize the run-time performance of the algorithm). In addition, when forced moves were involved, the search depth was extended (let f be the number of forced moves) because in these situations the player has no real decision to make. The ply depth was extended by steps of two, up to the smallest even number that was greater than or equal to the number of forced moves f that occurred along that branch. If the extended ply search produced more forced moves then the ply was once again increased in a similar fashion. Furthermore, if the final board position was left in an "active" state, where the player has a forced jump, the depth was once again incremented by two ply. Maintaining an even depth along each branch of the search tree ensured that the boards were evaluated after the opponent had an opportunity to respond to the player's move. The best move to make was chosen by iteratively minimizing or maximizing over the leaves of the game tree at each ply according to whether or not that ply corresponded to the opponent's move or the player's move. For more on the mechanics of alpha–beta search, see [11].

This evolutionary process, starting from completely randomly generated neural network strategies, was iterated for 250 generations. The best neural network (from generation 250) was then used to play against human opponents on an internet gaming site (www.zone.net). Each player logging on to this site is initially given a rating, R_0, of 1600 and a player's rating changes according to the following formula [which follows the rating system of the United States Chess Federation (USCF)]:

$$R_{\text{New}} = R_{\text{Old}} + C(\text{Outcome} - W) \tag{1}$$

where

$$W = \frac{1}{1 + 10^{\left(\frac{R_{\text{Opp}} - R_{\text{Old}}}{400}\right)}}$$

$$\text{Outcome} \in \{1 \text{ if Win, } 0.5 \text{ if Draw, } 0 \text{ if Loss}\}$$

[4]The choice reflected a desire to avoid networks that would tolerate losing. It would be interesting to observe any difference in performance that would result from setting the payoff to be zero sum.

R_{Opp} is the rating of the opponent, and $C = 32$ for ratings less than 2100.[5]

Over the course of two weeks, 90 games were played against opponents on this website. Games were played until: 1) a win was achieved by either side; 2) the human opponent resigned; or 3) a draw was offered by the opponent; and a) the piece differential of the game did not favor the neural network by more than one piece and b) there was no way for the neural network to achieve a win that was obvious to the authors, in which case the draw was accepted. There was a fourth condition which occurred infrequently in which the human opponents abandoned the game without resigning (by closing their graphical-user interface) thereby breaking off play without formally accepting defeat. When an opponent abandoned a game in competition with the neural network, a win was counted if the neural network had an obvious winning position (one where a win could be forced easily in the opinion of the authors) or if the neural network was ahead by two or more pieces; otherwise, the game was not recorded. In no cases were the opponents told that they were playing a computer program, and no opponent ever commented that they believed their opponent was a computer algorithm.

Opponents were chosen based primarily on their availability to play (i.e., they were not playing someone else at the time) and to ensure that the neural network competed against players with a wide variety of skill levels. In addition, there was an attempt to balance the number of games played as red or white. In all, 47 games were played as red. All moves were based on a ply depth of $d = 6$, 8, or in rare cases 10, depending on the perceived time required to return a move (less than a minute was desired). The majority of moves were based on $d = 6$.

IV. Results

Fig. 3 shows a histogram of the number of games played against players of various ratings along with the win–draw–loss record attained in each category. The evolved neural network dominated players rated 1800 and lower, and had almost as many losses as wins against opponents rated between 1800 and 1900. Fig. 4 shows the sequential rating of the neural network and the rating of the opponents played over all 90 games. Table I provides a listing of the class intervals and designations accepted by the USCF. The highest rating attained by the evolved neural network was 1975.8 on game 74. The final rating of the neural network was 1914.4. However, the order in which games are played has a significant impact on the final rating obtained.

From (1) the largest increase in rating occurs when a weak player defeats a strong player, while the largest decrease in rating occurs when a strong player loses to a weak player. Given that the 90 independent games played to evaluate the player could have been played in any order (since no

[5]More complicated transformations are applied for ratings that switch between designated classes above 2100 points, and the value for C changes as well. These situations were not relevant to the scores attained here. The formulas above pertain legitimately to players with established ratings based on 20 or more games, but the internet gaming zone appeared to use this formula consistently. The USCF uses a different rating formula for players with under 20 games. In essence, the internet gaming zone estimates the player's performance of their first 20 games to be 1600.

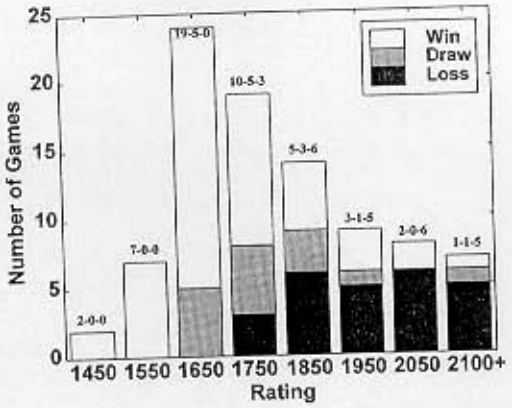

Fig. 3. The performance of the evolved neural network after 250 generations, played over 90 games against human opponents on an internet checkers site. The histogram indicates the rating of the opponent and the associated performance against opponents with that rating. Ratings are binned in intervals of 100 units (i.e., 1650 corresponds to opponents who were rated between 1600 and 1700). The numbers above each bar indicate the number of wins, draws, and losses, respectively. Note that the evolved network generally defeated opponents who were rated less than 1800, and played to about an equal number of wins and losses with those who were rated between 1800 and 1900.

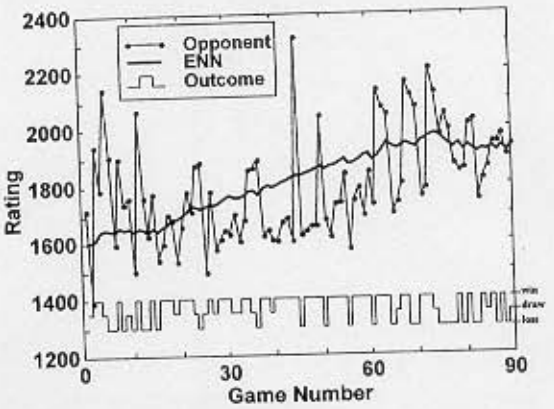

Fig. 4. The sequential rating of the evolved neural network (ENN) over the 90 games played against human opponents. The graph indicates both the network's rating and the corresponding rating of the opponent on each game, along with the result (win, draw, or loss). The highest rating for the ENN was 1975.8 on game 74. The final rating after game 90 was 1914.3, placing the ENN as a better than median Class A player.

TABLE I
The Relevant Categories of Player Indicated by the Corresponding Range of Rating Score

Class	Rating
Senior Master	2400+
Master	2200-2399
Expert	2000-2199
Class A	1800-1999
Class B	1600-1799
Class C	1400-1599
Class D	1200-1399
Class E	1000-1199
Class F	800-999
Class G	600-799
Class H	400-599
Class I	200-399
Class J	below 200

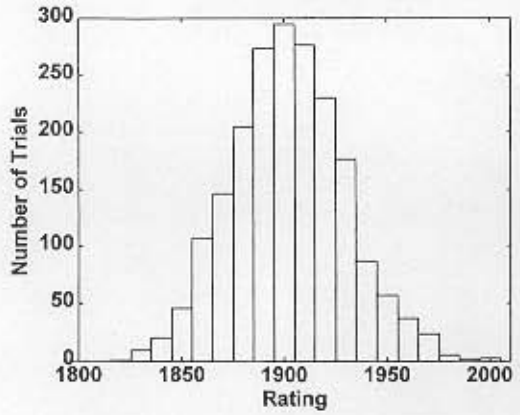

Fig. 5. The rating of the evolved neural network after 250 generations computed over 2000 random permutations of the 90 games against human opponents on the internet checkers site (www.zone.com). The mean rating was 1901.98, with a standard deviation of 27.57. The minimum and maximum ratings obtained were 1820.61 and 2006.39.

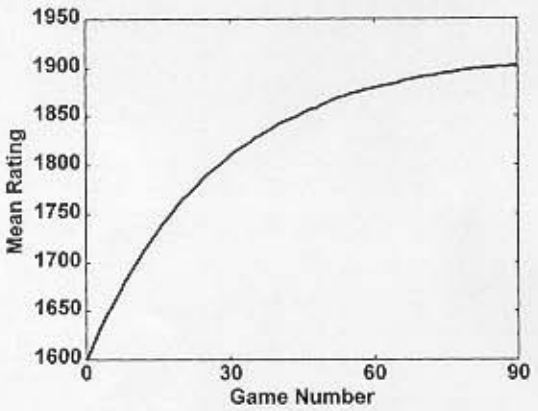

Fig. 6. The mean sequential rating of the evolved neural network (ENN) over 2000 random permutations of the 90 games played against human opponents. The mean rating starts at 1600 (the standard starting rating at the website) and steadily climbs to a rating of above 1860 by game 50. As the number of games reaches 90, the mean rating curve begins to saturate and reaches a value of 1901.98 which places the ENN subjectively as a Class A player.

learning was performed during the series of games played on the website), a better estimate of the network's true rating can be obtained by sampling from the population of all possible orderings of opponents. (Note that the total number of orderings is $90! \approx 1.48 \times 10^{138}$, which is too large to enumerate.) The network's rating was recalculated over 2000 random orderings drawn uniformly from the set of all possible orderings. Fig. 5 shows the histogram of the ratings that resulted from each permutation. The corresponding mean rating was 1901.98, with a standard deviation of 27.57. The minimum and maximum ratings obtained were 1820.61 and 2006.39. Fig. 6 shows the rating trajectory averaged over the 2000 permutations as a function of the number of games played. The mean rating starts at 1600 (the standard starting rating at the website) and steadily climbs to a rating of above 1860 by game 50. As the number of games reaches 90, the mean rating curve begins to saturate and reaches a value of 1901.98 which places it subjectively as a Class A player. For comparison, the top ten rated players registered at this internet site (as of December 13, 1998) had ratings of:

1) 2308
2) 2294
3) 2272
4) 2248
5) 2246
6) 2239
7) 2226
8) 2224
9) 2221
10) 2217

Thus the top ten players were all at the master level.

The best performance of the evolved network was likely recorded in a game against a player rated 2207 (master level), which ended in a draw. At the time, this opponent was ranked number 18 on the website out of more than 40 000 registered players. The sequence of moves is shown in the Appendix. Certain moves are annotated, but note that these annotations are not offered by an expert checkers player (instead being offered here by the authors). Undoubtedly, a more advanced player might have different comments to make at different stages in the game. Also shown is the sequence of moves of the evolved network in a win over an expert rated 2134, who was ranked 48th on the website.

V. Conclusion

Overall, the results indicate the ability for an evolutionary algorithm to start with essentially no preprogrammed information in the game of checkers (except the possibility for using piece differential as indicated above) and learn, over successive generations, how to play at a level that is challenging to many humans. Further, the best evolved neural network was sometimes able to defeat expert-level players, and even play to a draw against a master. No doubt, the limited ply depth that could be employed was a handicap to the performance observed. This was particularly evident in the end game, where it is not uncommon to find pieces separated by several open squares, and a search at $d = 6$, 8, or even 10, may not allow pieces to effectively "see" that there are other pieces within eventual striking distance. Moreover, the coordinated action of even two pieces moving to pin down a single piece can necessitate a long sequence of moves where it is difficult to ascribe advantage to one position over another until the final result is in view. Finally, it is well known that many end game sequences in checkers can require very high ply (e.g., [3, pp. 20–60]), and all of these cases were simply unavailable to the neural network to assess. With specially designed computer hardware, it would be possible to implement the best neural network directly on a chip and greatly increase the number of boards that could be evaluated per unit time, and thereby the ply that could be searched. Under the available computing environment, the speed was limited to evaluating approximately 3500 possible board positions per second. For comparison, Deep Blue was able to evaluate 200 million chess boards per second (Hoan, cited in [12]).

Another limitation of the procedure was in the use of minimax as a strategy for choosing the best move. Although this is a commonly accepted protocol, it is not always the best choice for maximizing the chances of obtaining a win against an opponent that may make a mistake. By assuming that the opponent will always make the move that is worst from the player's perspective, the player must play conservatively, minimizing that potential damage. This conservatism can work against the player, because when offered the choice between one move that engenders two possible opponent responses, each with values of say +0.05 and +0.2 points, respectively, and another move with two possible responses of 0.0 and +0.9 points, the minimax strategy will favor the first move because it can at worst still yield a gain of +0.05. But the qualitative difference between +0.05 and 0.0 is relatively small (both are effectively even positions), and if the second move had been favored there would have been the potential for the opponent to make an error, thereby leaving them in a nearly certain defeat (corresponding to the board evaluated at +0.9). The proper heuristic to use when evaluating the relative advantage of one move over another is not always clear.

To summarize, the information given to the evolving neural networks was essentially limited to:

1) a representation defining the location of each piece (and its type) on the board;
2) a variable coding value for a king;
3) a mechanism for computing all possible legal moves in any potential state of the game;
4) a means for searching ahead up to a prescribed ply depth;
5) a heuristic (minimax) for selecting which move to favor in light of the neural network evaluation function;
6) the potential to use piece differential as a feature.

None of these capabilities are much different from those that a novice human player brings to their first game. They are told the rules of how pieces move, thereby giving them the potential to make legal moves. They are told the object of the game, and the most direct manner to achieve that object is to remove the opponent's pieces, therefore having more pieces than your opponent is a clearly evident subgoal. They are told that kings have properties that extend beyond regular pieces, and they must choose some internal representation to separate these two types of pieces. And they are told that the game is played in turns, so it is again clearly evident that moves must be considered in light of what moves the opponent is likely to make in response. The novice human player also recognizes the spatial characteristics of the board, the nearness or distance between pieces, a series of empty squares in a row indicating the potential for moving unimpeded, and other nuances that carry over from recognizing patterns in everyday life. The neural network evolved here had no knowledge of the spatial nature of the game; its board was simply a 32-component vector rather than an eight by eight checker board. It would be of interest to assess the performance of neural networks that could evaluate board positions based upon such spatial features. Yet, even with this handicap, the evolutionary algorithm was able to learn how to play competent checkers based essentially from the information contained in "win, lose, or draw."

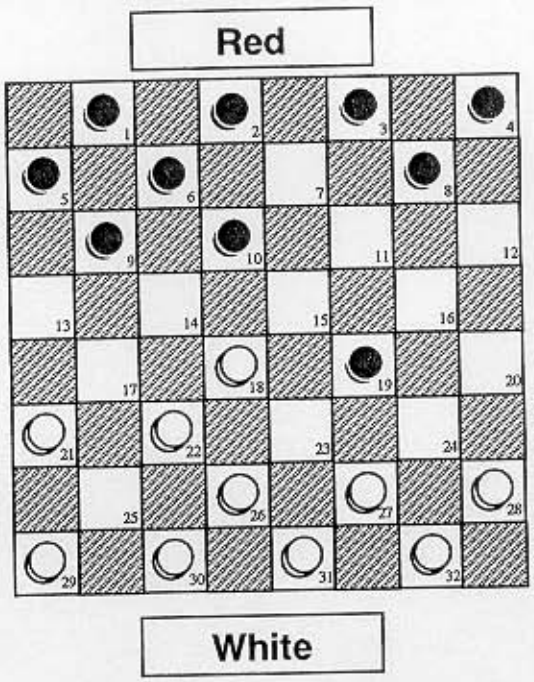

Fig. 7. White (human) to move. White chooses 21–17 which appears to be a mistake, as it results in white giving up a two for one exchange [R:9–14, W:18–9 (f); R:5–14–21(f)]. White never recovered from this deficit.

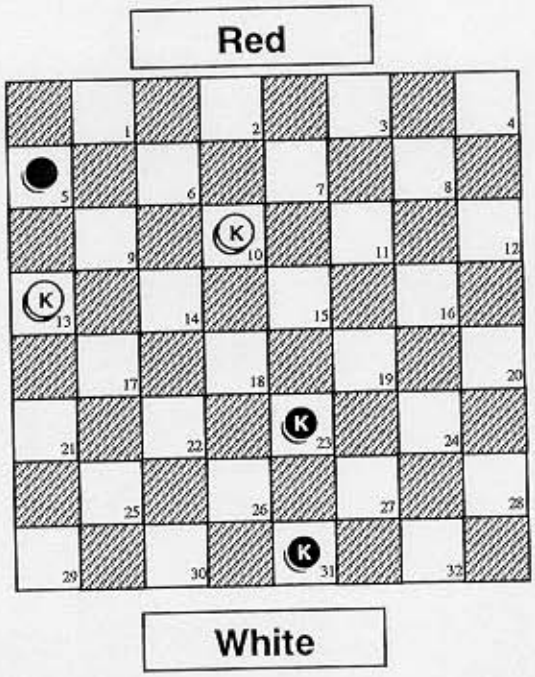

Fig. 8. The final board position, with White (human) to move. Both White (human) and Red (NN) toggle three times (R:27–23, W:13–17; and R:23–27, W:17–13) between the 45th and 50th moves. On move number 52, White (human) offered a draw which was accepted.

APPENDIX

This Appendix contains the complete sequence of moves from two selected games where the best-evolved neural network from generation 250 played to a draw against a human rated 2207 (master level) and defeated a human rated 2134 (expert level). The notation for each move is given in the form a–b, where a is the position of the checker that will move

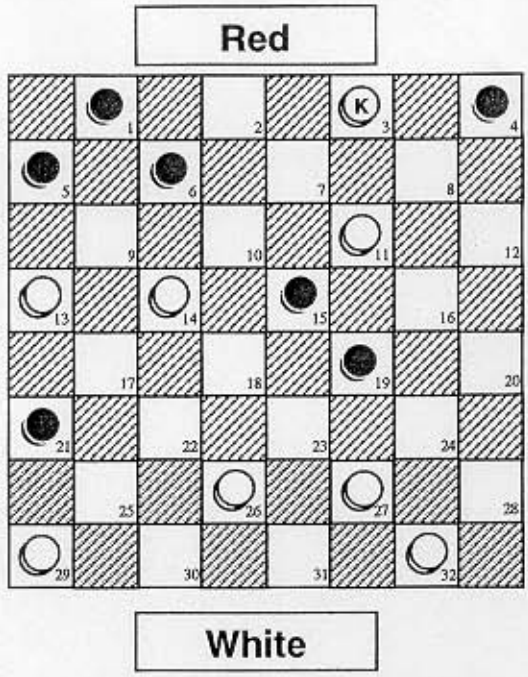

Fig. 9. The game reaches this position on move number 20 as Red (human) walks into a trap set by White (NN) by moving R:2–6. In the next three moves White gives up three pieces [20.R:2–6, 20.W:14–9; 21.R:5–14(f) 21.W:29–25; 22.R:21–30(f), 22.W:3–7; 23.R:30–23(f)] and comes back with a triple jump [23.W:27–18–9–2(f)], which also earns a king.

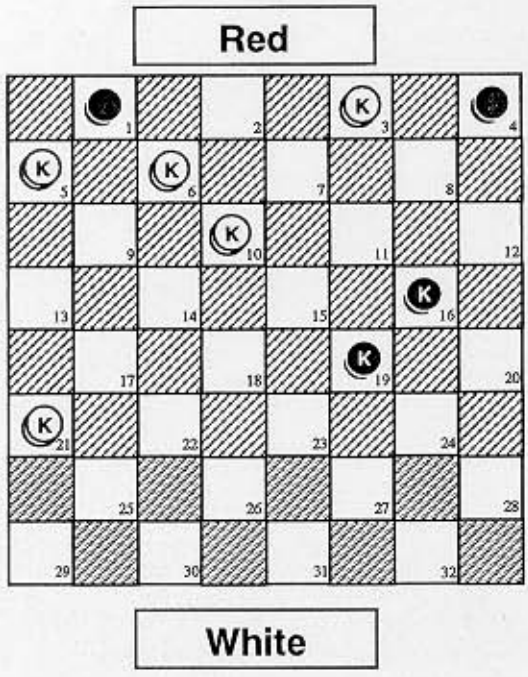

Fig. 10. An endgame position with Red (human) to move. Red chooses to move R:16–11. With six-ply alpha–beta search, the NN already sees a victory. It gives up three pieces [62:W:10–15; 63.R:11–18, 63.W:3–7; 64.R:1–10(f)] and comes back with a winning triple jump [64.W:7–14–23–16(f)]. The game continues, 65.R:4–8(f), 65.W:21–25, at which point White (uman) resigns. Note that White did not have to play R:11–18 but all other options also led inevitably to defeat.

and b is the destination. Forced moves (mandatory jumps or occasions where only one move is available) are indicated by (f). Accompanying the move sequences are two figures for each game indicating a pivotal position and the ending. The figures are referred to in the annotations.

Game against Human rated 2207 (master). NN plays red; human plays white. (f) denotes a forced move. Comments on moves are offered in brackets.

Computer	*Human*	*Comments*
1.R:11–16	1.W:22–18	
2.R:16–19	2.W:24–15	
3.R:10–19(f)	3.W:23–16(f)	
4.R:12–19(f)	4.W:25–22	
5.R:7–10	5.W:21–17	[Human gives up two for one exchange and never recovered]. (see Fig. 7)
6.R:9–14	6.W:18–9(f)	
7.R:5–14–21	7.W:22–18	
8.R:8–11	8.W:27–24	
9.R:11–16	9.W:24–15(f)	
10.R:10–19(f)	10.W:31–27	
11.R:3–7	11.W:18–15	[Attacking/defending piece on 19]
12.R:1–5	12.W:29–25	
13.R:4–8	13.W:27–24	
14.R:6–9	14.W:24–20	
15.R:8–12	15.W:20–11(f)	[NN willing to swap piece on 16 and give up king]
16.R:7–16(f)	16.W:15–11	[White goes for king]
17.R:19–23	17.W:26–19(f)	
18.R:16–23(f)	18.W:11–8	[NN pinning the back rank]
19.R:2–6	19.W:8–3	[Human gets king]
20.R:23–26	20.W:30–23(f)	[NN swaps out for king]
21.R:21–30(f)	21.W:3–8	[NN gets king with piece that made double jump on move 7]
22.R:30–26	22.W:23–19	
23.R:26–23	23.W:19–15	
24.R:12–16	24.W:8–11	
25.R:23–19	25.W:11–20(f)	[NN willing to swap]
26.R:19–10(f)	26.W:32–27	
27.R:9–13	27.W:20–16	
28.R:10–15	28.W:28–24	
29.R:15–18	29.W:16–19	
30.R:13–17	30.W:24–20	
31.R:17–22	31.W:20–16	[NN and Human heading for second kings]
32.R:22–26	32.W:19–23	[Attack]
33.R:26–31	33.W:23–14(f)	[NN parries by swapping]
34.R:31–24(f)	34.W:16–11	
35.R:24–27	35.W:11–7	
36.R:6–10	36.W:14–17	[NN—on the attack; Human-king backs off]
37.R:27–31	37.W:7–2	[Human gets king]
38.R:10–15	38.W:2–7	[NN heads for king]
39.R:15–18	39.W:7–11	
40.R:18–23	40.W:17–13	[Trapping piece on 5]
41.R:23–27	41.W:11–15	
42.R:27–32	42.W:15–19	[NN gets king]
43.R:32–27	43.W:19–15	
44.R:27–23	44.W:15–10	
45.R:23–27	45.W:10–14	[NN toggles position]
46.R:27–23	46.W:14–10	
47.R:23–27	47.W:10–15	
48.R:27–23	48.W:15–10	
49.R:23–27	49.W:10–15	
50.R:27–23	50.W:13–17	[6 ply search]
51.R:23–27	51.W:17–13	[8 ply search]
52.R:27–23	—	[Human offers draw accepted after 10 ply search] (see Fig. 8)

Game against Human rated 2134 (Expert). Human plays red, NN plays white. (f) denotes a forced move. Comments on moves are offered in brackets.

Human	*Computer*	*Comment*
1.R:11–15	1.W:22–18	[standard NN response]
2.R:15–22(f)	2.W:26–17	
3.R:8–11	3.W:17–13	
4.R:9–14	4.W:30–26	
5.R:11–15	5.W:25–22	
6.R:14–17	6.W:21–14(f)	[NN in danger of giving up a king after swap]
7.R:10–17(f)	7.W:24–19	
8.R:15–24(f)	8.W:28–19	[Swap]
9.R:17–21	9.W:22–17	[Good move—NN pins the piece on 21]
10.R:7–10	10.W:19–16	
11.R:12–19(f)	11.W:23–16(f)	
12.R:10–15	12.W:26–23	
13.R:15–19	13.W:31–26	[NN chooses not to swap]
14.R:3–7	14.W:16–12	
15.R:7–10	15.W:23–16(f)	[Human gives up a piece—no option?]
16.R:10–15	16.W:16–11	
17.R:15–19	17.W:12–8	[Human makes bad move? 15–18 better?]
18.R:6–10	18.W:8–3	[NN gets king]
19.R:10–15	19.W:17–14	
20.R:2–6	20.W:14–9	[NN sets the trap, see Fig. 9]
21.R:5–14(f)	21.W:29–25	[NN gives up king, goes down two pieces]
22.R:21–30(f)	22.W:3–7	
23.R:30–23(f)	23.W:27–18–9–2(f)	[NN springs trap, triple jumps to king]
24.R:19–23	24.W:7–10	
25.R:15–18	25.W:2–6	
26.R:18–22	26.W:6–9	
27.R:22–26	27.W:9–5	
28.R:26–31	28.W:11–7	[Human gets first king]
29.R:23–26	29.W:7–3	[NN gets king]
30.R:26–30	30.W:32–28	[Human gets king]
31.R:31–27	31.W:13–9	
32.R:30–26	32.W:9–6	[Going for king]
33.R:26–23	33.W:6–2	[NN gets fourth king]
34.R:23–19	34.W:3–7	
35.R:19–16	35.W:10–15	
36.R:27–32	36.W:15–11	
37.R:16–12	37.W:7–10	
38.R:32–27	38.W:11–7	
39.R:12–8	39.W:2–6	
40.R:8–12	40.W:7–3	
41.R:27–23	41.W:28–24	
42.R:12–16	42.W:24–20	
43.R:16–12	43.W:6–9	
44.R:23–18	44.W:9–13	
45.R:18–23	45.W:13–17	
46.R:23–19	46.W:17–13	[NN toggles after long exchange]
47.R:19–24	47.W:10–7	
48.R:24–19	48.W:7–10	
49.R:19–24	49.W:13–9	
50.R:24–19	50.W:20–16	
51.R:19–24	51.W:16–11	[NN going for fifth king]
52.R:12–16	52.W:11–7	
53.R:16–11	53.W:7–2	[NN gets fifth king]
54.R:11–16	54.W:2–7	
55.R:24–20	55.W:7–2	
56.R:20–24	56.W:2–7	
57.R:24–20	57.W:7–2	
58.R:16–19	58.W:9–14	[Human plays a different move]
59.R:20–16	59.W:14–17	
60.R:16–20	60.W:17–21	[NN move seems pointless]
61.R:20–16	61.W:2–6	
62.R:16–11	62.W:10–15	[NN gives up a piece having already seen victory, see Fig. 10]
63.R:11–18	63.W:3–7	
64.R:1–10(f)	64.W:7–14–23–16(f)	[NN moves in for the kill]
65.R:4–8(f)	65.W:21–25	[Human Resigns]

REFERENCES

[1] D. B. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. Piscataway, NJ: IEEE Press, 1995.

[2] J. Schaeffer, *One Jump Ahead: Challenging Human Supremacy in Checkers*. Berlin, Germany: Springer-Verlag, 1996.

[3] J. Schaeffer, R. Lake, P. Lu, and M. Bryant, "Chinook: The world man-machine checkers champion," *AI Mag.*, vol. 17.1, pp. 21–29, 1996.

[4] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM J. Res. Develop.*, vol. 3.3, pp. 210–219, 1959.

[5] G. Tesauro, "Practical issues in temporal difference learning, " *Machine Learning*, vol. 8, pp. 257–277, 1992.

[6] J. B. Pollack and A. D. Blair, "Coevolution in the successful learning of backgammon strategy," *Machine Learning*, vol. 32, pp. 225–240, 1998.

[7] G. Tesauro, "Comments on 'coevolution in the successful learning of backgammon strategy'," *Machine Learning*, vol. 32, pp. 241–243, 1998.

[8] M. Minsky, "Steps toward artificial intelligence," in *Proc. IRE*, Jan. 1961, vol. 49, no. 1, pp. 8–30.

[9] J. Schaeffer, J. Culberson, N. Treloar, B. Knight, P. Lu, and D. Szafron, "A world championship caliber checkers program," *Artificial Intell.*, pp. 273–290, vol. 53, no. 2–3, 1992.

[10] N. J. L. Griffith and M. Lynch, "Neurodraughts: The role of representation, search, training regime and architecture in a TD draughts player," Univ. Limmerick, Ireland, unpublished Tech. Rep., 1997.

[11] H. Kaindl, "Tree searching algorithms," in *Computers, Chess, and Cognition*, T. A. Marsland and J. Schaeffer, Eds. New York: Springer-Verlag, 1990, pp. 133–168.

[12] D. Clark, "Deep thoughts on deep blue," *IEEE Expert*, vol. 12.4, p. 31, 1997.

[13] D. E. Moriarty and R. Mikkulainen, "Discovering complex othello strategies through evolutionary neural networks," *Connectionist Sci.*, vol. 7, no. 3, pp. 195–209, 1995.

Kumar Chellapilla received the M.Sc. degree in electrical engineering in 1997 from Villanova University, Villanova, PA. He is currently pursuing the Ph.D. degree at the University of California, San Diego.

His current research interests include evolutionary computation and his dissertation research involves the design and optimization of intelligent systems using evolutionary computation. He has published more than 40 papers in evolutionary computation in journals and conferences.

Mr. Chellapilla was Cochair of the special session on evolutionary programming held at the Third Annual Genetic Programming Conference, Madison, WI, July, 1998. He is a member of several programs, committees, and international conferences and has been a reviewer for various journals including the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, the IEEE TRANSACTIONS OF NEURAL NETWORKS, and the PROCEEDINGS OF THE IEEE. He has also offered tutorials and invited talks in the field of evolutionary computation at several conferences. He is a member of many technical societies including the American Association of Artificial Intelligence, the Evolutionary Programming Society, the Institute of Electrical and Electronics Engineers, Phi Kappa Phi, and Sigma Xi.

David B. Fogel (M'89–SM'94–F'99) received the Ph.D. degree in engineering sciences (systems science) from the University of California, San Diego, in 1992.

He is Executive Vice President and Chief Scientist of Natural Selection, Inc., La Jolla, CA, a small business focused on solving difficult problems in industry, medicine, and defense using evolutionary computation, neural networks, fuzzy systems, and other methods of computational intelligence. He has more than 150 publications including 4 books, most recently *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, 2nd ed. (Piscataway, NJ: IEEE Press) and *Evolutionary Computation: The Fossil Record* (Piscataway, NJ: IEEE Press, 1998). He is coeditor-in-chief of the *Handbook of Evolutionary Computation* (Oxford, U.K.: Oxford Univ. Press, 1997) and coauthor of *How to Solve It: Modern Heuristics*, (New York, Springer-Verlag, to be published).

Dr. Fogel is the founding editor-in-chief of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION. He also serves as an Associate Editor or member of the editorial board for several other archive journals.