Effect of Look-Ahead Depth in Evolutionary Checkers

Article *in* Journal of Computer Science and Technology · September 2012

DOI: 10.1007/s11390-012-1280-6

2 authors:

Belal Al-Khateeb
University of Anbar
73 PUBLICATIONS **652** CITATIONS

SEE PROFILE

Graham Kendall
University of Nottingham, Malaysia Campus
393 PUBLICATIONS **15,045** CITATIONS

SEE PROFILE

Al-Khateeb B, Kendall G. Effect of look-ahead depth in evolutionary checkers. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 27(5): 996–1006 Sept. 2012. DOI 10.1007/s11390-012-1280-6

Effect of Look-Ahead Depth in Evolutionary Checkers

Belal Al-Khateeb[1] and Graham Kendall[2], *Senior Member*, *IEEE*

[1] *College of Computer, Al-Anbar University, Ramadi, Iraq*

[2] *School of Computer Science, The University of Nottingham, Nottingham, U.K.*

E-mail: belal@computer-college.org; gxk@cs.nott.ac.uk

Received August 30, 2011; revised July 13, 2012.

Abstract It is intuitive that allowing a deeper search into a game tree will result in a superior player to one that is restricted in the depth of the search that it is allowed to make. Of course, searching deeper into the tree comes at increased computational cost and this is one of the trade-offs that has to be considered in developing a tree-based search algorithm. There has been some discussion as to whether the evaluation function, or the depth of the search, is the main contributory factor in the performance of an evolved checkers player. Some previous research has investigated this question (on Chess and Othello), with differing conclusions. This suggests that different games have different emphases, with respect to these two factors. This paper provides the evidence for evolutionary checkers, and shows that the look-ahead depth (like Chess, perhaps unsurprisingly) is important. This is the first time that such an intensive study has been carried out for evolutionary checkers and given the evidence provided for Chess and Othello this is an important study that provides the evidence for another game. We arrived at our conclusion by evolving various checkers players at different ply depths and by playing them against one another, again at different ply depths. This was combined with the two-move ballot (enabling more games against the evolved players to take place) which provides strong evidence that depth of the look-ahead is important for evolved checkers players.

Keywords evolutionary Checkers, look-ahead depth, neural network

1 Introduction

Designing automated computer game-playing programs can be traced back to the 1950s[1-2], with significant interest still being evident today. Deep Blue represents one of the landmark successes when, in May 1997, a computer program defeated, arguably, the best chess player (Garry Kasparov) that has ever lived[3-4]. Game playing crosses discipline boundaries, being of interest to areas such as artificial intelligence (AI), knowledge representation, automated search and machine learning. A knowledge-based approach is often used in traditional computer games programs, whereby encoding by hand is utilised to incorporate human knowledge about the game into the computer by means of an evaluation function and a database of opening and endgame sequences. Chinook, similar to Deep Blue, uses this methodology and, in 1994, it became the first machine to be crowned a world champion when it defeated the world checkers champion (Marion Tinsley) at that time[5].

Chinook and Deep Blue require considerable effort by the design teams as the evaluation function is designed and tuned by experts: both algorithmic experts and game experts (e.g., chess grand masters). This is not only time consuming but only produces an automated game player that is able to play a single game. If it were possible to automate (e.g., evolve) an evaluation function, then the same methodology could be used across many games, without the need for human experts.

Evolutionary algorithms are a class of algorithms that draw their inspiration from biological processes. Genetic algorithms[6], for example, are motivated by Darwin's principle of natural evolution (*Survival of the Fittest*). A population of candidate solutions compete against one another for survival, with the individuals that are best suited to their environment (e.g., produce the best solution to the problem at hand) surviving to the next generation, whilst the poorest performing individuals die off. Each individual is subject to random mutation, as well as being bred with other members of the population to produce children which are (hopefully) better than the two parents they were bred from.

Genetic programming[7] follows a similar paradigm. Whereas genetic algorithms typically manipulate vectors representing solutions, genetic programming manipulates syntactically correct computer programs.

Regular Paper

The algorithmic steps are similar in both versions of the algorithms (evaluation of individuals, selection of the best individuals, breeding and random mutation), but the underlying representations are very different.

Neural networks[8], model (albeit a very simplified version) the brain and the processes that it performs (a set of inter-connected neurons and whether neurons fire or not). A relatively recent innovation is to *evolve* neural networks, where a population of neural networks exist and compete for survival, coming under evolutionary pressure in the same that is apparent in genetic algorithms and genetic programming. Indeed, genetic algorithms are just one possible way to control the evolutionary process, in that the neural networks are encoded as a vector which enables manipulation by a genetic algorithm.

In this work, we evolve a population of neural networks[9] utilising evolution strategies[10-13], where the connection weights of the network are randomly changed (mutated) by a random value drawn from a normal distribution.

Evolutionary computation actually covers a number of areas, being one of the categories under the more general heading of AI. Other evolutionary computation methodologies include ant colony optimisation, particle swarm optimisation and evolution strategies. Excellent introductions to this area can be found in [10-13].

Blondie24[14] attempted to evolve a neural network that could play checkers at the level of a human expert, but without the need for human knowledge to be incorporated, which was not the case with Deep Blue and Chinook. We note that the ideas in Blondie24 were also later applied to Chess[15−17]. Blondie24 was not provided with any human expertise. Instead it evolves its own evaluation function through self-play and the networks are subject to evolutionary pressure. By using only the positions and type of pieces on the board, together with a piece difference (the difference between the number of pieces for each player), the evolutionary algorithm utilises feedforward artificial neural networks to evaluate alternative positions in the game. The architecture of Blondie24 is shown in Fig.1[14].

This feedforward neural network consists of two hidden layers, of 40 and 10 nodes respectively. The output layer has one node. This provides a real-valued output, which is used to measure the quality of the board position. That is, we are attempting to evolve the weights of the connections in the network such that higher values are provided by the output neuron when the input to the network is presented with a *better* board position. Of course, we cannot be sure whether a given position is better than another one, but we hope that evolutionary pressure will give us some insights without the need for human expertise. The input neurons reflect all the 32 board positions where checkers can reside. In addition, the network inputs also reflect the temporal nature of the board in an attempt to show that squares are connected diagonally, but not horizontally or vertically.

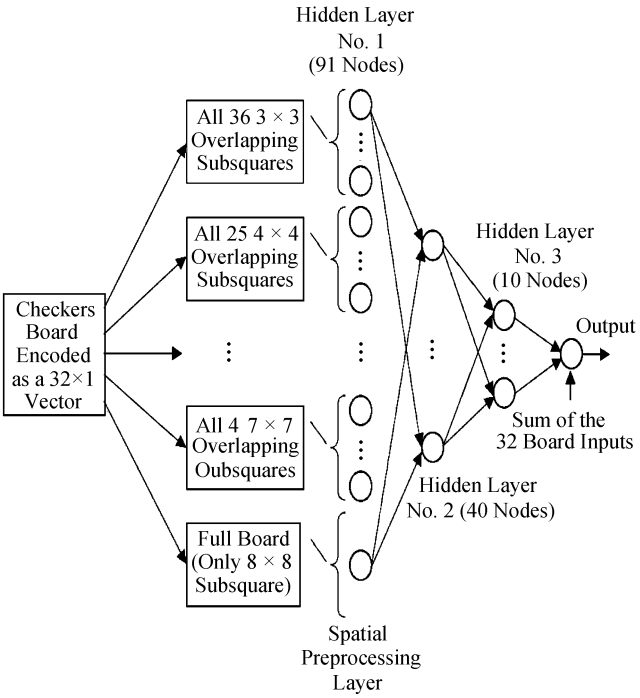

Fig.1. Blondie24 architecture[14].

All the layers are fully connected. The total number of weights and biases is 5 046 and these values are evolved by the algorithm presented in Section 5.

Fogel[14], stated that "*At four ply, there really is not any "deep" search beyond what a novice could do with a paper and pencil if he or she wanted to*". Whilst this might be true, it would be an immense task for a human player to derive a 4-ply search tree with paper and pencil, for every move they have to make, let alone make sense of the tree (e.g., manually apply minimax) to decide the best move to make. Of course, the search might be done at some subconscious level, especially if *intuitive* pruning were taking place, but this (as far as we are aware) has not been reported in the scientific literature.

Although, there has been a lot of discussion about the importance of the look-ahead depth used in Fogel's work[14], there is little work that has rigorously investigated its importance. In this paper, we extend our previous work[18] to investigate the importance of the look-ahead depth to an evolutionary checkers player. In [18] we showed that players with higher plies perform better than those with a lower ply, when playing at a

fixed ply of six. In this work we carry out two additional experiments, where players evolve/play at different plies, in order to provide additional evidence to support our hypothesis.

Our experiments demonstrate that the look-ahead depth is important for the learning process of computer checkers. This is achieved through evolving various checkers players with different ply depths and by using three different methods for playing them.

The original work by Fogel and Chellapilla[19-22] was an important milestone in artificial intelligence and evolutionary computation. It has motivated many researchers to carry out work in the area. The full list of papers are too numerous to cite here but the interested readers are pointed towards the proceedings of the IEEE Congress of Evolutionary Computation, IEEE Transactions on Evolutionary Computation and IEEE Transactions on Computational Intelligence and AI in Games. However, since the seminal work of Fogel and Chellapilla, researchers have tackled many games such as Cribbage[23], Go[24] as well as other areas such as finance[25-26]. It is interesting to note that Go is now considered to be the Drosophila of artificial intelligence (replacing Chess) due, in part, to its large branching factor. A recent survey paper[27] provides an excellent starting point for readers interested in this area, where Monte Carlo Tree Search is emerging as the algorithm of choice.

Given the wide breadth of research that has emanated from the original work in this area, there are still many areas left to explore, as well as new research directions to investigate. In this paper, we take a reflective look at the work of Fogel and investigate if look ahead is a contributory factor in its effectiveness. To do this, we use the original algorithm of Fogel, and explore the effect of adjusting various parameters.

The rest of the paper is organised as follows. Section 2 will discuss the game of Checkers, while the related work is presented in Section 3. The two-move ballot is discussed in Section 4. Section 5 shows our experimental setup for this work. Section 6 presents our results and we conclude the paper in Section 7.

2 Checkers

This section gives a brief description of Checkers, and the next section summarises previous approaches that have been used to automate playing the game.

Checkers, sometimes called draughts, is played on an 8×8 board between two players, black and white (black moves first). Each player has 12 pieces (also called checkers), which are placed on the 12 alternating squares of the same colour that are closest to that player's side as shown in Fig.2. Checkers can only move forward diagonally one square at a time. The aim of the game is to remove all the opponent's pieces by jumping them (when your piece is adjacent to an opponent's piece with an empty square on the other side). If a checker is able to jump another checker it must make that move; a so-called forced jump. Multiple jumps are similarly enforced. If multiple forced jumps are available the player may choose which one to make. When any pieces advance to the last row of the board (on the opponent's side), that piece becomes a King and it can move diagonally one square at a time either forward or backward. The game ends when one player removes all his/her opponent's pieces, or when a player has no more available moves. The game can also end when one player offers a draw and the other accepts.

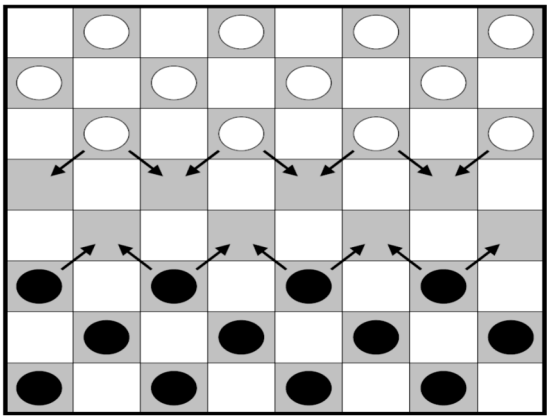

Fig.2. Checkers board (showing possible moves). Black moves first.

3 Background

Arthur Samuel, in 1954, attempted to illustrate that a computer program could play against itself in order to evolve a checkers player, using an early form of temporal difference learning. Samuel's program adjusted weights for 39 features[2,28]. These features were adjusted during the game using a method we now refer to as reinforcement learning, instead of tuning the weights manually[29-32]. Samuel found that piece difference was more important feature than the other 38 features (e.g., capacity for advancement, control of the centre of the board, threat of fork) taking on various levels of importance. Due to memory limitations, Samuel only used 16 of the 38 features in his evaluation function. To include the remaining 22 features he swapped between features using a procedure called term replacement[14,28]. Samuel used two evaluation functions (Alpha and Beta) to determine the weights for the features. The process gives an appropriate weight to each parameter and sums them together. This evaluation function evaluates each

leaf node in the game tree. This process is considered to be one of the first attempts to use heuristic search methods in the quest for the next best move in a game tree. Samuel used minimax with three ply and a procedure called *rote learning*[2] was also included in the program. For further reading about Alpha, Beta and minimax, the interested reader is referred to [33-34].

Starting in 1989, Jonathan Schaeffer and his colleagues at the University of Alberta, designed a checkers program called Chinook[5,35]. Solving the game was Schaeffer's initial motivation. However, this was a challenging goal as there are approximately 5×10^{20} different positions to evaluate[36]. A further motivation was to produce the world's best checkers player. This was done by using an evaluation function, which includes several features, all based on human expertise, including grand masters. Like Samuel's program, the main feature in Chinook's evaluation function is the piece count. The next most important features are the king and the *runaway* checker (a clear path for a checker to become a king, without any obstacles). The summation of each term provides an overall assessment of the board for a particular game state. Initially, Schaeffer gave initial values to the weight of each feature and then manually tuned them when he found an error or when a Chinook move led to a losing position.

Chinook also utilised opening and endgame databases to further enhance its ability. Chinook's opening game database contained more than 40 000 sequences. The endgame database contained all the possibilities that lead to a win, a draw or a loss, for a given number of pieces left on the board. The final version of Chinook's endgame database contained all 6-piece end sequences, allowing it to play perfectly from these positions.

In 1989 Chinook, with a 4-piece endgame database[37], won the computer Olympiad. Later, with its final 6-piece endgame database, together with its evaluation function modified by a *fudge* factor[35,38], it finished in second place to Marion Tinsley (recognized as the best checkers player who ever played the game) in the U.S. National Checkers Championship held in 1990. After a further sequence of matches in 1994 between Chinook and Tinsley, Chinook became the World Man-Machine Checkers Champion (after Tinsley's resignation due to health problems, who died in the following year)[35]. In 1996 Chinook retired with rating at 2 814.

The building of the opening and endgame databases ultimately led Schaeffer to achieve his initial motivation (solving the game of checkers)[39]. Perfect play by both sides leads to a draw.

Blondie24 attempted to evolve a computer checkers program, injecting as little expert knowledge as possible[14,19-22,40]. Evolutionary neural networks were used as a self-learning computer program. The neural network used for a particular player provided the evaluation function for a given board position. The networks acted randomly at first (as their weights were initialized randomly), and gradually improved over time. The final network was able to beat the majority ($> 99\%$) of human players who registered on www.zone.com at that time. Blondie24 is a significant achievement, particularly in machine learning[32] and AI although it does not play at the level of Chinook[5]. However this was not the objective of the research; it aimed to answer the challenge set by Samuel[1-2] and what Newell and Simon (two early AI pioneers) said that progress in this area would not be made without addressing the credit assignment problem.

Al-Khateeb and Kendall[41] enhanced Blondie24 by introducing a round robin tournament, instead of randomly choosing five opponents in each round. The results showed that this type of modification did evolve a superior checkers player, even catering for the fact that it evolved over less generations in order to have the same number of matches.

Many researchers have shown the importance of the look-ahead depth for computer games, but none of them was related to checkers. Most of the findings are related to chess[42-45], where it was shown that increasing the depth level will produce superior chess players. However, Runarsson and Jonsson[46] showed that this was not the case for Othello, as they found that better playing strategies are found when temporal difference (TD) learning with ε-greedy is applied with a lower look-ahead search depth during evolution and a deeper look-ahead search during game play.

Given that Chess appears to benefit from a deeper look-ahead while Othello does not, this paper will establish if the evolutionary method for playing checkers benefits from a deeper look-ahead.

Even though Blondie24 answered the challenge set by Samuel, it has still attracted comments about its design. One of them is concerned with the piece difference feature and how it affects the learning process of Blondie24. This was answered by Fogel[14,40], Evan Hughes[47], and Al-Khateeb and Kendall[48]. The results show that both the piece difference and the neural network architecture contribute to the learning. Another design issue questions the importance of the look-ahead depth. Al-Khateeb and Kendall[18] showed that the look-ahead depth is important for the evolutionary algorithm that is used to construct a checkers player. Preliminary experiments for this work are reported in [18] and we will extend it in this paper by adding two

further experiments. This provides further evidence and enables us to draw stronger conclusions. Therefore, this paper has three different experiments (see Section 4). The first experiment is the same as in [18] while the second and the third experiments are new to [18].

4 Two-Move Ballot and Standard Rating Formula in Checkers

When the world's best players play the game of checkers, it often ends in a draw. To overcome this and make the games more competitive, the Two-Move Ballot is used. This was introduced in the 1870s[5]. The first two moves (each side's first move) are randomly chosen. There are 49 possibilities to play in this way, but research has shown that six of these moves (openings) are unbalanced, as it will give an advantage to one side over the other. Therefore, only 43 of the 49 available moves are considered. At the start of the game a card is randomly chosen indicating which of the 43 openings is to be played. The original game, with no forced opening moves, is called go-as-you-please (GAYP).

Checkers players are rated according to a standard system[14] (following the tradition of the United States Chess Federation) where the initial rating for a player is $R_0 = 1\,600$ and the player's score is adjusted based on the outcome of a match (*outcome*) and the rating of the opponent:

$$R_{\text{new}} = R_{\text{old}} + C(\mathit{Outcome}\text{-}W),$$

where

- R_{new} is the new rating for the player and R_{old} is the current player's rating,
- $W = 1/(1 + 10^{((R_{\text{opp}} - R_{\text{old}})/400)})$,
- *Outcome* is 1 for win, 0.5 for draw, or 0 for loss,
- R_{opp} is the opponent's rating,
- $C = 32$ for ratings less than 2 100, $C = 24$ for ratings between 2 100 and 2 399, and $C = 16$ for ratings at or above 2 400.

We use the standard rating formula in this work to compare different players. For the purpose of showing how the players perform, relative to one another, we use 5 000 different (random) orderings for the 86 (each player plays 43 games as black and 43 games as white) games and then compute the mean and the standard deviation for the standard rating formulas. We use 5 000 orderings for the wins, losses and draws in order to derive a distribution of ratings for a given set of matches, in order to capture the fact that the initial ordering of matches is purely arbitrary. We say that a player is better than his/her opponent if his/her mean standard rating formula puts him/her in a level that is higher than his/her opponent. The determination of the player's level is determined according to Table 1. We note that the purpose of this paper is to compare the performance of the two players and not to measure their actual ratings, which could only realistically be done by playing against a number of different players.

Table 1. Relevant Categories of Player Indicated by the Corresponding Range of Rating Score[14]

Class	Rating
Senior Master	2 400+
Master	2 200∼2 399
Expert	2 000∼2 199
Class A	1 800∼1 999
Class B	1 600∼1 799
Class C	1 400∼1 599
Class D	1 200∼1 399
Class E	1 000∼1 199
Class F	800∼999
Class G	600∼799
Class H	400∼599
Class I	200∼399
Class J	Below 200

5 Experimental Setup

For the purpose of investigating our hypothesis (i.e., showing the importance (or not) of a look-ahead to the game of checkers), an evolutionary method for playing checkers, based on the same algorithm that was used to construct Blondie24, was implemented in order to provide a platform for our research. The following algorithm is used to construct the checkers player.

1) Initialize a random population of 30 neural networks (strategies), P_i, $i = 1, \ldots, 30$, sampled uniformly $[-0.2, 0.2]$ for the weights and biases.

2) Each strategy has an associated self-adaptive parameter vector, s_i, $i = 1, \ldots, 30$, initialized to 0.05.

3) Each neural network plays against five other neural networks selected randomly from the population.

4) For each game, each competing player is allowed to search for 4-ply and receives a score of +1 for a win, 0 for draw and −2 for a loss.

5) Games are played until either one side wins, or until one hundred moves are made by both sides, in which case a draw is declared.

6) After completing all games, the 15 strategies that have the highest scores are selected as parents and retained for the next generation. Those parents are then mutated to create another 15 offspring using the following equations:

For each parent P_i, $i = 1, \ldots, 15$, an offspring P_i' was created by:

$$s_i(j) = s_i(j) \exp(tN_j(0,1)), \quad j = 1, \ldots, N_w,$$
$$w_i(j) = w_i(j) + s_i(j)N_j(0,1), \quad j = 1, \ldots, N_w,$$

where $w_i(j)$ is the weight to be updated, N_w is the number of weights and biases in the neural network (here this is 5 046), $t = \frac{1}{\sqrt{2\times\sqrt{Nw}}} = 0.083\,9$, and $N_j(0,1)$ is a standard Gaussian random variable resample for every j.

7) Repeat steps $3 \sim 6$ for 840 generations.

We note that the values we used for the various parameters (e.g., number of neural networks, number of generations) were chosen as they were used by Fogel in his original work[20-22].

Our implementation has the same structure and architecture that Fogel utilised in Blondie24. Four players were evolved.

1) C_1 was evolved using 1-ply depth.

2) C_2 was evolved using 2-ply depth.

3) C_3 was evolved using 3-ply depth.

4) C_4 was evolved using 4-ply depth.

Three different experiments were carried out in order to provide an extensive review for the importance of the look-ahead depth to an evolutionary method for playing checkers. These experiments were:

1) Each player played against the rest of players but was now allowed to search to a depth of 6-ply.

2) Each player played against the rest of players using the ply they were trained at.

3) Each player played against itself at a different ply.

6 Results

All the experiments were run using the same computer (1.86 GHz Intel Core2 processor and 2 GB RAM). All the experiments to evolve the players were run for the same period, 19 days. The algorithms were run for this amount of time as, when running the original algorithm, it took 19 days to execute the 840 generations that were originally used in Fogel's algorithm. In order to provide fair comparisons all the other algorithms were also run for this length of time. This means that the algorithms differed in the number of generations that were executed, but ran for the same time (on the same computer), which seems to be the fairest way to compare the various algorithms.

6.1 Results of Playing C_1, C_2, C_3 and C_4 Using 6-Ply

To measure the effect of increasing the ply depth, each player trained at a given ply was matched with all of the other players trained with a different ply. A league was held between C_1, C_2, C_3 and C_4; each match in the league was played using the idea of a two-move ballot (see Section 4). For each match we played all of the 43 possible games, both as black and white. This gives a total of 86 games. The total number of games played was 258. Each game was played using 6-ply. The games were played until either one side won or a draw was declared after 100 moves for each player. The results are shown in Tables 2∼4 and Fig.3.

Table 2. Number of Wins (for the Row Player) out of 258 Games

	C_1	C_2	C_3	C_4	Σ Wins
C_1	-	28	17	13	58
C_2	33	-	24	19	76
C_3	45	31	-	25	101
C_4	59	40	35	-	134

Table 3. Number of Draws (for the Row Player) out of 258 Games

	C_1	C_2	C_3	C_4	Σ Draws
C_1	-	25	24	14	63
C_2	25	-	31	27	83
C_3	24	31	-	26	81
C_4	14	27	26	-	67

Table 4. Number of Losses (for the Row Player) out of 258 Games

	C_1	C_2	C_3	C_4	Σ Losses
C_1	-	33	45	59	137
C_2	28	-	31	40	99
C_3	17	24	-	35	76
C_4	13	19	25	-	57

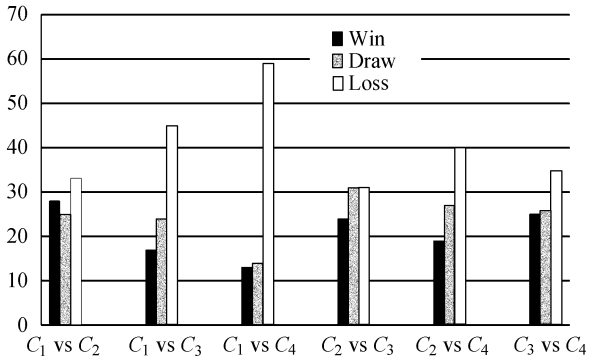

Fig.3. Results of playing a league between C_1, C_2, C_3 and C_4 using 6-ply.

It is clear from Tables 2 and 4 that the total number of wins increases and the total number of losses decreases when the evolved ply depth increases. Therefore, increasing the ply depth leads to a superior player. Table 5 shows the mean and the standard deviation (SD) of the players' ratings after 5 000 different orderings for the 86 games, while Table 6 summarises the results when playing the league between players using

a starting position where all pieces are in their original positions (i.e., no two-move ballot).

Table 5. Standard Rating for All Players after 5 000 Different Orderings of the 86 Games Played

	Mean	SD	Class
C_1	1188.94	28.94	E
C_2	1206.24	27.62	D
C_1	1146.58	27.40	E
C_3	1266.18	26.14	D
C_1	1264.11	27.21	D
C_4	1474.99	26.14	C
C_2	1179.47	26.85	E
C_3	1205.10	25.60	D
C_2	1114.61	27.17	E
C_4	1200.21	25.88	D
C_3	1176.02	28.26	E
C_4	1205.26	26.98	D

Table 6. Wins/Losses for C_1, C_2, C_3 and C_4 When Not Using Two-Move Ballot Using 6-Ply

		C_2	C_3	C_4
C_1	Black	Lost	Lost	Lost
	White	Drawn	Lost	Lost
C_2	Black	-	Lost	Lost
	White	-	Drawn	Lost
C_3	Black		-	Lost
	White		-	Lost

The results in Table 5, obtained using 5 000 different orderings for the 86 games (obtained using the two-move ballot) show that increasing the ply depth by one increases the performance of the checkers player as C_2 is better (using our definition given earlier with respect to players having a different rating class) than C_1, C_3 is better than C_2 and C_4 is better than C_3, and by using the average value for the standard rating formula the results (when playing C_2 against C_1) put C_2 in class D (rating = 1 206) and put C_1 in Class E (rating = 1 189). Also the results (when playing C_3 against C_2) in Table 5 put C_3 in class D (rating = 1 205) and put C_2 in class E (rating = 1 179) and finally (when playing C_4 against C_3) put C_4 in class D (rating = 1 205) and put C_3 in class E (rating = 1 176). Table 6 shows that C_2 won as black and drew as white when playing against C_1 using a starting position where all pieces are in their original positions, also C_3 won as black and drew as white when playing against C_2, and C_4 won as black and as white when playing against C_3.

The results shown in Table 5 also show that increasing the ply depth by two increases the performance of the checkers player as C_3 and C_4 perform better than C_1 and C_2 respectively, and by using the average value for the standard rating formula, the results (when playing C_3 against C_1) put C_3 in class D (rating = 1 266) and C_1 in Class E (rating = 1 147), while (when playing C_4 against C_2), C_4 is in Class D (rating = 1 200) and C_2 is in class E (rating = 1 115). Also C_3 won as black and as white when playing against C_1 and C_4 won as black and as white when playing against C_2 using a starting position where all pieces are in their original positions as shown in Table 6.

Finally the results in Table 5 show that C_4 is performing better than C_1, and by using the average value for the standard rating formula, the results (when playing C_4 against C_1) puts C_4 in class C (rating = 1 475) and C_1 in class D (rating = 1 264), while Table 6 shows that C_4 won as black and as white when playing against C_1 using a starting position where all pieces are in their original positions.

6.2 Results of Playing C_1, C_2, C_3 and C_4 Using the Ply They Were Trained at

In this experiment, a league was held between C_1, C_2, C_3 and C_4, and each match in the league was played using the idea of a two-move ballot (see Section 4). For each match we played all of the 43 possible games, both as black and white. This gives a total of 86 games. The total number of games played was 258. Each game was played using the ply they were trained at. The games were played until either one side won or a draw was declared after 100 moves for each player. The results are shown in Tables 7 ∼ 9 and Fig.4.

Table 7. Number of Wins (for the Row Player) out of 258 Games

	C_1	C_2	C_3	C_4	Σ Wins
C_1	-	25	13	9	47
C_2	36	-	21	15	72
C_3	51	37	-	25	113
C_4	64	49	36	-	149

Table 8. Number of Draws (for the Row Player) out of 258 Games

	C_1	C_2	C_3	C_4	Σ Draws
C_1	-	25	22	13	61
C_2	25	-	28	22	75
C_3	22	28	-	25	75
C_4	13	22	25	-	61

Table 9. Number of Losses (for the Row Player) out of 258 Games

	C_1	C_2	C_3	C_4	Σ Losses
C_1	-	36	51	64	151
C_2	25	-	37	49	111
C_3	13	21	-	36	70
C_4	9	15	25	-	49

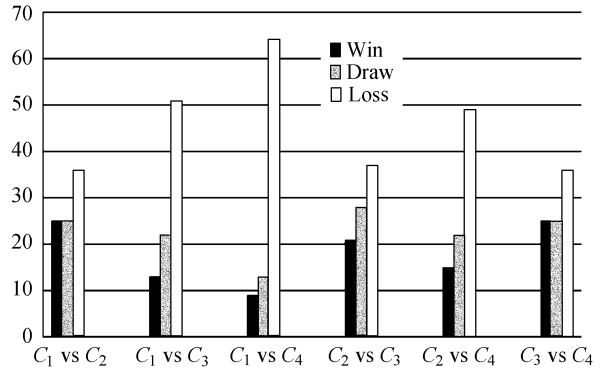

Fig.4. Results of playing a league between C_1, C_2, C_3 and C_4 using the ply they were trained at.

The results in Tables 7∼9 show that the number of wins decreases as the number of ply of the opponents increases. This result is as one may have been expected. The total numbers of wins are summed up in the last column. Those cases players trained and playing at higher ply exhibit the best performance. Table 10 shows the mean and the standard deviation (SD) of the players' ratings after 5 000 different orderings for the 86 played games, while Table 11 summarises the results when playing the league between players using a starting position where all pieces are in their original positions (i.e., no two-move ballot).

Table 10. Standard Rating Formula for All Players after 5 000 Different Orderings of the 86 Games Played

	Mean	SD	Class
C_1	1167.96	27.14	E
C_2	1215.12	28.30	D
C_1	1158.80	25.86	E
C_3	1327.26	26.58	D
C_1	1257.51	24.66	D
C_4	1525.32	25.10	C
C_2	1177.67	26.11	E
C_3	1202.98	27.15	D
C_2	1167.74	26.20	E
C_4	1317.52	27.00	D
C_3	1167.07	27.19	E
C_4	1216.00	28.35	D

Table 11. Wins/Losses for C_1, C_2, C_3 and C_4 When not Using Two-Move Ballot Using the Ply They Were Trained at

		C_2	C_3	C_4
C_1	Black	Lost	Lost	Lost
	White	Drawn	Lost	Lost
C_2	Black	-	Lost	Lost
	White	-	Drawn	Lost
C_3	Black		-	Lost
	White		-	Lost

The results in Table 10, obtained using 5 000 different orderings for the 86 games (obtained using the two-move ballot) show that increasing the ply depth by one increases the performance of the checkers player as C_2 is better (using our definition given earlier with respect to players having a different rating class) than C_1, C_3 is better than C_2, C_4 is better than C_3. And by using the average value for the standard rating formula the results (when playing C_2 against C_1) put C_2 in class D (rating = 1 215) and put C_1 in Class E (rating = 1 167). Also the results (when playing C_3 against C_2) in Table 10 put C_3 in class D (rating = 1 202), put C_2 in class E (rating = 1 177), and finally (when playing C_4 against C_3) put C_4 in class D (rating = 1 216) and put C_3 in class E (rating = 1 167). Table 11 shows that C_2 won as black and drew as white when playing against C_1 using a starting position where all pieces are in their original positions, also C_3 won as black and drew as white when playing against C_2, and C_4 won as black and as white when playing against C_3.

The results shown in Table 10 also show that increasing the ply depth by two increases the performance of the checkers player as C_3 and C_4 perform better than C_1 and C_2 respectively. And by using the average value for the standard rating formula, the results (when playing C_3 against C_1) put C_3 in class D (rating = 1 327) and C_1 in Class E (rating = 1 158), while (when playing C_4 against C_2), C_4 is in Class D (rating = 1 317) and C_2 is in class E (rating = 1 167). Also C_3 won as black and as white when playing against C_1 and C_4 won as black and as white when playing against C_2 using a starting position where all pieces are in their original positions as shown in Table 11.

Finally the results in Table 10 show that C_4 performs better than C_1, and by using the average value for the standard rating formula, the results (when playing C_4 against C_1) puts C_4 in class C (rating = 1 525) and C_1 in class D (rating = 1 257). Table 11 shows that C_4 won as black and as white when playing against C_1 using a starting position where all pieces are in their original positions.

6.3 Results of Playing C_1, C_2, C_3 and C_4 amongst Themselves

In this experiment, each player played against itself using different plies. Each match was played using the idea of a two-move ballot (see Section 4). For each match we played all of the 43 possible games, both as black and white. This gives a total of 86 games. The total number of games played was 258. The games were played until either one side won or a draw was declared after 100 moves for each player. The results are shown in Tables 12∼14.

Table 12. Number of Wins (for the Row Player) out of 258 Games

	1-Ply	2-Ply	3-Ply	4-Ply	Σ Wins
C_1	-	20	15	8	43
C_2	36	-	26	24	86
C_3	49	41	-	22	112
C_4	65	52	40	-	157

Table 13. Number of Draws (for the Row Player) out of 258 Games

	1-Ply	2-Ply	3-Ply	4-Ply	Σ Draws
C_1	-	30	22	13	65
C_2	30	-	29	20	79
C_3	22	29	-	24	75
C_4	13	20	24	-	57

Table 14. Number of Losses (for the Row Player) out of 258 Games

	1-Ply	2-Ply	3-Ply	4-Ply	Σ Losses
C_1	-	36	49	65	150
C_2	20	-	41	52	113
C_3	15	26	-	40	81
C_4	8	24	22	-	54

The results in Tables 12∼14 clearly show that playing with deeper look-ahead search will result in larger number of wins, which confirms that playing at higher ply will increase player performance and hence give superior performance.

7 Conclusions

In this paper we have evolved different checkers players using an evolutionary algorithm that is based on Fogel's work to evolve Blondie24.

The experiments produced many checkers players, using different depths of ply during learning. Three different experiments were done in which our expectations were that better value functions would be learned when training with deeper look-ahead search. This was found to be the case. The main results are that, during training and game playing, better decisions are made when deeper look-ahead is used.

The first experiment showed that players with higher ply perform better than those with a lower ply, when playing at a fixed ply of six. The second experiment also showed that players trained with higher plies also performed better than the players trained with a lower ply, when playing at the ply that they were trained at. The third experiment showed that players trained on a higher ply perform significantly worse when playing at a lower ply. This suggests that there is no point playing at a level that is lower than the trained level.

An interesting point to note from the results is that increasing the depth level by one will give a different performance depending on the level. For example, the results in Table 2 indicates that increasing the level number from two to three gives a better performance than the performance gained when increasing the level number from one to two. The same occurs when increasing the depth from three to four, which is better than increasing the depth from one to two and from two to three. This suggests that the performance will increase when training at a level of five, six and so on. It is normally the case that increasing the ply depth will increase the computational cost of evolving the player. This is not the case in our experiments. As we mentioned before all the experiments were run for the same amount of time (19 days).

The results suggest that starting with a depth of four ply is the best value function to start with during learning phase for checkers. That is, train at four ply and then play at the highest possible ply.

There are many possible research directions. Directly related to this work, it would be interesting to investigate the findings of this work to other perfect information games such as Chess. As we said in the introduction, there has been much work carried out in this area and it would benefit from concentrated effort into certain games to see if any theoretical insights can be established which are lacked at the moment. Looking at imperfect, or stochastic games would also be welcomed. For example, bridge, backgammon, poker and (particularly) Go.

Other extensions would also be welcomed. Hybridisations with methodologies such as Monte Carlo Tree Search[27] have the potential to make significant improvements by further increasing the depth of the search that can be explored.

Moreover, there is much scope to extend the methodologies outlined in the paper to other areas such as the stock market prediction.

In summary, the look-ahead depth is an important design feature in evolutionary checkers and is worthy of further attention for other games.

References

[1] Turing A M. Computing machinery and intelligence. *Mind*, 1950, 59: 433-460.

[2] Samuel A L. Some studies in machine learning using the game of checkers. *IBM Journal on Research and Development*, 1959, 3(3): 210-229.

[3] Newborn M. Kasparov versus Deep Blue. Computer Chess Comes of Age. New York: Springer-Verlag, 1996.

[4] Campbell M, Hoane Jr. A J, Hsu F H. Deep blue. *Artificial Intelligence*, 2002, 134(1-2): 57-83.

[5] Schaeffer J. One Jump Ahead: Computer Perfection at Checkers (2nd edition). New York: Springer, 2009.

[6] Sastry K, Goldberg D, Kendall G. Chapter 4: Genetic algorithms. In *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, Burke E K, Kendall G (eds.), Springer, 2005, pp.97-125.

[7] Koza J, Poli R. Chapter 5: Genetic programming. In *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, Burke E K, Kendall G (eds.), Springer, 2005, pp.127-164.

[8] Fausett L V. Fundamentals of Neural Networks: Architectures, Algorithms and Applications, Prentice Hall, 1993.

[9] Yao X. Evolving artificial neural networks. *Proc. the IEEE*, 1999, 87(9): 1423-1447.

[10] Jong K A D. Evolutionary Computation: A Unified Approach. Cambridge, USA: MIT Press, 2006.

[11] Eiben A E, Smith J E. Introduction to Evolutionary Computing (1st edition). Springer, 2003.

[12] Fogel D B. Evolutionary Computation: Toward a New Philosophy of Machine Intelligence (3rd edition). Piscataway, USA: IEEE Press, 1995.

[13] Michalewicz Z, Fogel D. How to Solve It: Modern Heuristics. Springer-Verlag, 2000.

[14] Fogel D B. Blondie24: Playing at the Edge of AI. San Francisco, USA: Morgan Kaufmann Publishers, 2002.

[15] Fogel D B, Hays T J, Hahn S L, Quon J. A self-learning evolutionary chess program. *Proceeding of the IEEE*, 2004, 92(12): 1947-1954.

[16] Fogel D B, Hays T J, Hahn S L, Quon J. Further evolution of a self-learning chess program. In *Proc. the IEEE 2005 Symposium on Computational Intelligence & Games*, April 2005, pp.73-77.

[17] Fogel D B, Hays T J, Hahn S L, Quon J. The Blondie25 chess program competes against Fritz 8.0 and a human chess master. In *Proc. the IEEE 2006 Symposium on Computational Intelligence and Games*, May 2006, pp.230-235.

[18] Al-Khateeb B, Kendall G. The importance of look-ahead depth in evolutionary checkers. In *Proc. the 2011 IEEE Congress on Evolutionary Computation*, June 2011, pp.2252-2258.

[19] Chellapilla K, Fogel D B. Anaconda defeats Hoyle 6-0: A case study competing an evolved checkers program against commercially available software. In *Proc. the 2000 Congress on Evolutionary Computation*, July 2000, 2: 857-863.

[20] Chellapilla K, Fogel D B. Evolution, neural networks, games, and intelligence. *Proceedings of the IEEE*, 1999, 87(9): 1471-1496.

[21] Chellapilla K, Fogel D B. Evolving an expert checkers playing program without using human expertise. *IEEE Transactions on Evolutionary Computation*, 2001, 5(4): 422-428.

[22] Chellapilla K, Fogel D B. Evolving neural networks to play checkers without relying on expert knowledge. *IEEE Transactions on Neural Networks*, 1999, 10(6): 1382-1391.

[23] Kendal G, Shaw S. Investigation of an adaptive cribbage player. In *Proc. the 3rd International Conference on Computers and Games*, July 2002, pp.29-41.

[24] Cai X, Venayagamoorthy G K, Wunsch D C II. Evolutionary swarm neural network game engine for capture go. *Neural Networks*, 2010, 23(2): 295-305.

[25] Binner J M, Gazely A M, Kendall G. An evaluation of UK risky money: An artificial intelligence approach. *Global Business and Economics Review*, 2009, 11(1): 1-18.

[26] Kendall G, Su Y. Imperfect evolutionary systems. *IEEE Transactions on Evolutionary computation*, 2007, 11 (3): 294-307.

[27] Cameron B B, Powley E, Whitehouse D, Lucas S M, Cowling P I, Rohlfshagen P, Tavener S, Perez D, Samothrakis S, Colton S. A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 2012, 4(1): 1-43.

[28] Samuel A L. Some studies in machine learning using the game of checkers II: Recent progress. *IBM Journal on Research and Development*, 1967, 11(6): 601-617.

[29] Kaelbling L P, Littman M L, Moore A W. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 1996, 4: 237-285.

[30] Sutton R S, Barto A G. Reinforcement Learning: An Introduction. Cambridge, USA: MIT Press, 1998.

[31] Vrakas D, Vlahavas I P L. Artificial Intelligence for Advanced Problem Solving Techniques. Hershey, USA: Information Science Reference, 2008.

[32] Mitchell T M. Machine Learning. McGraw-Hill, 1997.

[33] Von Neumann J. Zur Theorie der Gesellschaftsspiele. *Math. Annalen*, 1928, 100(1): 295-320.

[34] Edwards D J, Hart T P. The Alpha-Beta Heuristic (AIM-030). Massachusetts Institute of Technology, 1963, http://dspace.mit.edu/handle/1721.1/6098.

[35] Schaeffer J, Lake R, Lu P, Bryant M. Chinook: The world man-machine checkers champion. *AI Magazine*, 1996, 17(1): 21-30.

[36] Russell S, Norvig P. Artificial Intelligence: A Modern Approach (3rd edition). Prentice Hall, 2009.

[37] Schaeffer J, Culberson J C, Treloar N, Knight B, Lu P, Szafron D. A world championship caliber checkers program. *Artificial Intelligence*, 1992, 53(2/3): 273-289.

[38] Fogel D B. Evolutionary Computation: Toward a New Philosophy of Machine Intelligence (2nd edition). USA: Wiley-IEEE Press, 2000.

[39] Schaeffer J, Burch N, Björnsson Y, Kishimoto A, Müller M, Lake R, Lu P, Sutphen S. Checkers is solved. *Science Express*, 2007, 317(5844): 1518-1522.

[40] Fogel D B, Chellapilla K. Verifying anaconda's expert rating by competing against Chinook: Experiments in co-evolving a neural checkers player. *Neurocomputing*, 2002, 42(1-4): 69-86.

[41] Al-Khateeb B, Kendall G. Introducing a round robin tournament into Blondie24. In *Proc. the IEEE 2009 Symposium on Computational Intelligence and Games*, Sept. 2009, pp.112-116.

[42] Levene M, Fenner T I. The effect of mobility on minimaxing of game trees with random leaf values. *Artificial Intelligence*, 2001, 130(1): 1-26.

[43] Nau D S, Luštrek M, Parker A, Bratko I, Gams M. When is it better not to look ahead?. *Artificial Intelligence*, 2010, 174(16-17): 1323-1338.

[44] Smet P, Calbert G, Scholz J, Gossink D, Kwok H-W, Webb M. The effects of material, tempo and search depth on win-loss ratios in chess. In *Proc. the 16th Australian Conf. Artificial Intelligence*, Dec. 2003, pp.501-510.

[45] Bettadapur P, Marsland T A. Accuracy and savings in depth-limited capture search. *International Journal of Man-Machine Studies*, 1988, 29(5): 497-502.

[46] Runarsson T P, Jonsson E O. Effect of look-ahead search depth in learning position evaluation functions for Othello using ε-greedy exploration. In *Proc. the IEEE 2007 Symposium on Computational Intelligence and Games*, April 2007, pp.210-215.

[47] Hughes E J. Piece difference: Simple to evolve. In *Proc. the 2003 Congress on Evolutionary Computation*, Dec. 2003, pp.2470-2473.

[48] Al-Khateeb B, Kendall G. The importance of a piece difference feature to Blondie24. In *Proc. the 10th Annual Workshop on Computational Intelligence*, Sept. 2010, pp.1-6.

Belal Al-Khateeb received the B.Sc. (honors) (first class) and M.Sc. degrees in computer science from Al-Nahrain University, Iraq, in 2000 and 2003 respectively, and the Ph.D. degree from the School of Computer Science, University of Nottingham, UK, in 2011. He is currently a lecturer at the College of Computer, Al-Anbar University. He has published over 14 refereed journal and conference papers. His current research interests include evolutionary and adaptive learning particularly in computer games, expert systems, and heuristics and meta/hyper-heuristics. He has a particular interest in computer games programming. Dr. Al-Khateeb is a reviewer of two international journals (including one IEEE Transaction) and four conferences.

Graham Kendall received the B.S. (honors) (first class) degree in computation from the University of Manchester Institute of Science and Technology (UMIST), UK, in 1997, and the Ph.D. degree from the School of Computer Science, University of Nottingham, UK, in 2000. Before entering academia, he spent almost 20 years in the IT industry, working for various UK companies, taking on roles such as computer operator, technical support manager, and service manager. He is currently the Dunford Professor of computer science with the Automated Scheduling, Optimization, and Planning Research Group, School of Computer Science, University of Nottingham. He also holds the position of Vice-Provost (Research and Knowledge Transfer) at Nottingham's Malaysia Campus. He has edited and authored 9 books and published over 175 refereed journal and conference papers. His current research interests include scheduling, optimization, evolutionary and adaptive learning, and heuristics and meta/hyper-heuristics. He has a particular interest in solving real-world problems. Prof. Kendall is a fellow of the Operational Research Society and an Associate Editor of seven international journals (including two IEEE Transactions). He was one of the co-founders of the IEEE Symposium of Computational Intelligence in Games and he also chairs (since 2003) the Steering Committee of the Multidisciplinary International Conference on Scheduling: Theory and Applications.