The Importance of Look-Ahead Depth in Evolutionary Checkers

Belal Al-Khateeb
School of Computer Science
The University of Nottingham
Nottingham, UK
bxk@cs.nott.ac.uk

Graham Kendall, *Senior Member, IEEE*
School of Computer Science
The University of Nottingham
Nottingham, UK
gxk@cs.nott.ac.uk

Abstract— **Intuitively it would seem to be the case that any learning algorithm would perform better if it was allowed to search deeper in the game tree. However, 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 the player. There has been some evidence suggesting that look-ahead (i.e. depth of search) is particularly important. In this work we provide a rigorous set of experiments, which support this view. We believe this is the first time such an intensive study has been carried out for evolutionary checkers. Our experiments show that increasing the depth of a look-ahead has significant improvements to the performance of the checkers program and has a significant effect on its learning abilities.**

I. INTRODUCTION

Samuel's checkers player [1,2] was considered as one of the early attempts to design an automated computer game playing program. Interest today has not diminished and the introduction of Deep Blue in 1997 (after a series of modifications to a previous version from 1996) represents one of the landmark successes in this area. In 1997 Deep Blue defeated Garry Kasparov, arguably the best chess player that has ever lived [3,4]. Many important aspects of interest to artificial intelligence are encompassed by game playing, such as knowledge representation, search and machine learning. A knowledge-based approach is often used in traditional computer games programs, by incorporating human knowledge about the game into the computer by means of an evaluation function and a database of opening and end game sequences.

Chinook became the first machine to be crowned a world champion when it defeated the current checkers world champion (Marion Tinsley) [5]. In our view Chinook and Deep Blue are both significant achievements, requiring considerable effort by the teams behind them.

Blondie24 [6] is an evolutionary algorithm, capable of playing the game of checkers. One of its objectives was not to provide the algorithm with human expertise, which is often done with other game playing architectures. By using only the positions and type of pieces on the board, together with a piece difference, the evolutionary algorithm utilises feedforward artificial neural networks to evaluate alternative positions in the game. This is a direct contradiction of the alternative, which is to preload the algorithm with all the information about how to make good moves and avoid bad ones. The architecture of Blondie24 is shown in Figure 1 [6].
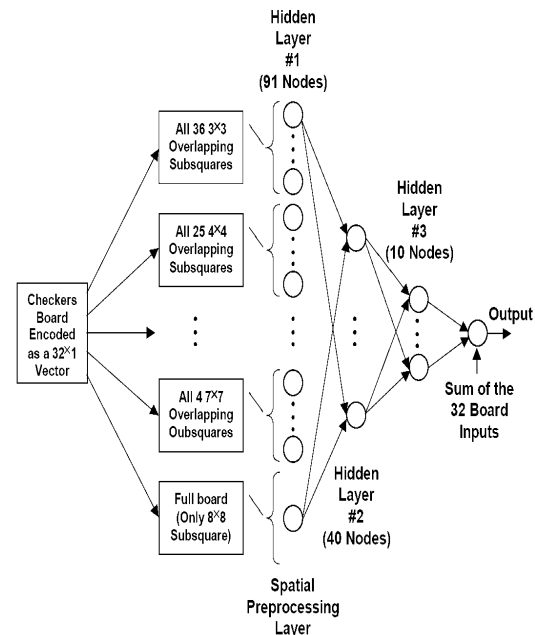

Figure 1: Blondie24 Architecture [6]

Although, there has been a lot of discussion about the importance of the look-ahead depth level used in Fogel's work [6], there is little work that has rigorously investigated its importance. Fogel, in his work on evolving Blondie24 [6], showed the importance of using a four ply search in Blondie24 by stating that "*At four ply, there really isn't any "deep" search beyond what a novice could do with a paper and pencil if he or she wanted to*". In fact we don't believe that this is the case as generating all the possible moves from a four ply

search is not an easy task for novices, and would also be time consuming. Of course, it might be done at some subconscious level, where pruning is taking place, but this (as far as we are aware) has not been reported in the scientific literature. In this paper, a study that investigates the importance of a look-ahead depth is conducted to investigate how well a checkers program performs by using various depth levels in order to provide evidence as to whether this is important or not.

Our experiments demonstrate that the look-ahead depth is important for the learning process of computer checkers. This is achieved through playing two leagues between two sets of players.

The rest of the paper is organised as follows; in section II related work is presented. Blondie24 is discussed in section III. Section IV shows our experimental setup for this work. Section V presents our results and we conclude in Section VI.

II. BACKGROUND

In an attempt to illustrate that a computer program could improve by playing against itself, Arthur Samuel, in 1954, started work on evolving a checkers player, using an early form of temporal difference learning. Samuel's program adjusted weights for 39 features [2,7]. These features were adjusted during the game using a method we now refer to as reinforcement learning, instead of tuning the weights manually [8-11]. Piece difference was found to be the most important feature with the other 38 features (e.g. capacity for advancement, control of the centre of the board, threat of fork, etc.) taking on various levels of importance. In his evaluation function, out of the 38 features, Samuel only used 16. This was because of memory limitations. To include the remaining 22 features he swapped between features using a procedure called term replacement [6]. Samuel used two evaluation functions (Alpha and Beta) to determine the weights for the features. At the start, Alpha and Beta have identical weights for every feature. While Beta weights remain unchanged, Alpha weights were modified during the course of the algorithm. The process gave an appropriate weight to each parameter and summed them together. This evaluation function was applied to evaluate 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 included in the program. This procedure is responsible for storing the evaluation of different board positions in a look-up table for fast retrieval (look-ahead and memorization). Samuel also incorporated alpha-beta pruning that included a supervised learning technique to allow the program to learn how to select the best parameters to calculate the evaluation function [7].

Traditional knowledge-based approaches for developing game-playing machine intelligence are sometimes criticised for the human expertise that is incorporated into the algorithm, and also for the inability of the programs to learn [6,12,13]. One of the criticisms of traditional knowledge-based approaches for developing game-playing machine intelligence

is the large amount of pre-injected human expertise that is required for the computer program, together with the lack of learning capabilities of these programs [6,12]. Domain experts provide the evaluation function, along with opening and end game databases. This means that the *intelligence* of a computer game is achieved from a pre-designed evaluation function and a look up of database moves. Moreover, this intelligence, unlike human intelligence, is not adaptive. Humans collect experience and knowledge from reading books and watching the play of other people before playing themselves. Humans also further their skill through trial-and-error. Novice players, rather than grand masters, could discover new features and strategies for playing a game. Old features could also be discarded and the strategies abandoned. Humans also adapt their strategies when they meet different types of players, under different conditions, in order to accommodate their special characteristics. We do not see such adaptations and characteristics in the knowledge based computer game programs. Fogel commented on this phenomenon in computer game-playing [6]:

"… *To date, artificial intelligence has focused mainly on creating machines that emulate us. We capture what we already know and inscribe that knowledge in a computer program. We program computers to do things – and they do those things, such as play chess, but they only do what they are programmed to do. They are inherently "brittle". … We'll need computer programs that can teach themselves how to solve problems, perhaps without our help. …*"

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 [14-17], where it was shown that increasing the depth level will produce superior chess players. However, Runarsson and Jonsson [18] showed that this was not the case for Othello, as they found that better playing strategies are found when TD learning when ε–greedy is applied with a lower look-ahead search depth and a deeper look-ahead search during game play.

Given that chess appears to benefit from a deeper look-ahead, but this is not true for Othello, this paper will establish if checkers benefits from a deeper look-ahead.

When humans play checkers at the expert level the game often ends in a draw. To overcome this, and make the games more competitive, the Two-Move Ballot is used.

The Two-Move Ballot 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 showed that six of these 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 moves is to be played. It is worth mentioning that the original game, with no forced opening moves is called go-as-you-please (GAYP).

Checkers players are rated according to a standard system [6] (following the tradition of the United States Chess Federation) where the initial rating for a player is $R_0 = 1600$

and the player's score is adjusted based on the outcome of a match and the rating of the opponent:

$$R_{new} = R_{old} + C(Outcome - W)$$

Where

- $W = 1/(1 + 10^{((Ropp - Rold)/400)})$
- *Outcome value 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 2100, $C = 24$ for ratings between 2100 and 2399, and $C = 16$ for ratings at or above 2400.

For the purpose of providing some form of statistical test, we will use 5000 different orderings for the 86 (each player plays 43 games as red and 43 games as white) games and then compute the mean and the standard deviation for the standard rating formulas. We say that a player is statistically better than his opponent if his mean value of the standard rating formula puts him in a level that is higher than his opponent. The determination of the player level is 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.

Blondie24 [19-23], was designed to address Samuel's challenge; to build a machine that could teach itself *how* to play rather than *being* told, and to recognize the important features rather than having the pre-programmed.

Al-Khateeb and Kendall [24] enhanced Blondie24 (which we present in the next section) by introducing a round robin tournament, instead of randomly choosing the opponents. The results are reported in [24], and we utilise this work in this paper (see Section V).

Table 1: The relevant categories of player indicated by the standard rating system [6].

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

III. BLONDIE24

Blondie24 represents a landmark in evolutionary learning by attempting to design a computer checkers program, injecting as little expert knowledge as possible [6,19-23]. 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. Evolutionary pressure motivated these networks, which acted randomly initially (as their weights were initialized randomly), to gradually improve over time. The final network was able to beat the majority (>99%) of human players registered on www.zone.com at that time. Blondie24 represents a significant achievement, particularly in machine learning and artificial intelligence although Blondie24 does not play at the level of Chinook [5]. However this was not the objective of the research; but rather it aimed to answer the challenge set by Samuel [1,2] and which Newell and Simon (two early AI pioneers) said that progress in this area would not be made without addressing the credit assignment problem. The major difference between Blondie24 and other traditional game-playing programs is in the employment of the evaluation function [19,20]. In traditional game-playing programs, the evaluation function usually consists of important features drawn from human experts. The weighting of these features are altered using hand tuning. Whereas, in Blondie24, the evaluation function is an artificial neural network that only knows the number of pieces on the board, the type of each piece and their positions. The neural network is not pre-injected with any other knowledge that experienced players would have.

The following algorithm represents Blondie24 [19-20]:

1- Initialise a random population of 30 neural networks (strategies), P_i=1,…,30, sampled uniformly [-0.2,0.2] for the weights and biases.

2- Each strategy has an associated self-adaptive parameter vector, s_i=1,…,30 initialised 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 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 was 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:

$$s_i(j) = s_i(j)exp(tN_j (0,1)), j = 1, ..., N_w$$
$$w_i(j) = w_i(j) + s_i(j)N_j(0,1), j = 1, ..., N_w$$

where N_w is the number of weights and biases in the neural network (here this is 5046), $t = \dfrac{1}{\sqrt{2 \times \sqrt{N_w}}} = $ 0.0839, and $N_j(0,1)$ is a standard Gaussian random variable calculated for every j.

7- Repeat steps 3 to 6 for 840 generations (this number was an arbitrary choice in the implementation of Blondie24).

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 [6][22], Evan Hughes [25], and Al-Khateeb and Kendall [26]. The obtained results showed that both the piece difference and the neural network architecture contribute to the learning.

IV. 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 checkers player, based on the same algorithm that was used to construct Blondie24, was implemented in order to provide a platform for our research. Our implementation has the same structure and architecture that Fogel utilised in Blondie24. Four players were evolved.

1- C_1 is evolved using one ply depth.
2- C_2 is evolved using two ply depth.
3- C_3 is evolved using three ply depth.
4- C_4 is evolved using four ply depth.

Each player played against all other players but was now allowed to search to a depth of 6-ply.

Our previous efforts to enhance Blondie24 introduced a round robin tournament [24]. We also use this player (Blondie24-RR) to investigate the importance of the look-ahead depth. This is done by implementing three other players, which are the same as Blondie24-RR, but, trained on different ply depths, those players are as follows.

1- Blondie24-RR1Ply is evolved using one ply depth.
2- Blondie24-RR2Ply is evolved using two ply depth.
3- Blondie24-RR3Ply is evolved using three ply depth.

It is worth mentioning that Blondie24-RR (reported in [24]) is constructed using a four ply depth. Each player was set to play against all the other three players but now using 6-ply.

V. RESULTS

All the experiments were run using the same computer (1.86 GHz Intel core2 processor and 2GB Ram). All the experiments to evolve the players were run for the same period (19 days, which reflects the same period used by Fogel but taking into account improved computational power).

A Results for C_1, C_2, C_3 and C_4

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 is 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 II). For each match we play all of the 43 possible games, both as red and white. This gives a total of 86 games. The total number of games played was 258. Each game is played using 6-ply. The games were played until either one side wins or a draw is declared after 100 moves for each player. The results are shown in tables 2 thru 4 and figure 2.

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	-	27	103
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	91
C_4	14	27	26	-	67

Table 4: Number of losses (for the row player) out of 258 games.

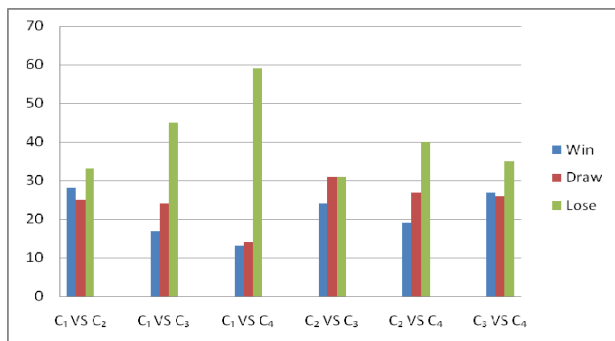|	C_1	C_2	C_3	C_4	Σ losses
C_1	-	33	45	59	137
C_2	28	-	31	40	99
C_3	17	24	-	33	74
C_4	13	19	27	-	59

Figure 2: Results of playing a league between C_1, C_2, C_3 and C_4.

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 of the players' ratings after 5000 different orderings for the 86 played 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 formula for all players after 5000 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/Loses for C_1, C_2, C_3 and C_4 when not using two-move ballot.

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

The results in table 5, obtained using 5000 different orderings for the 86 games (obtained using the two-move ballot) show that increasing 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 = 1206) and put C_1 in Class E (rating = 1189). Also the results (when playing C_3 against C_2) in table 5 put C_3 in class D (rating = 1205) and put C_2 in class E (rating = 1179) and finally (when playing C_4 against C_3) put C_4 in class D (rating = 1205) and put C_3 in class E (rating = 1176). Table 6 shows that C_2 won as red 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 red and drew as white when playing against C_2 and C_4 won as red and as white when playing against C_3.

The results shown in table 5 also show that increasing ply depth by two increases the performance of the checkers player as C_3 and C_4 are significantly better than the 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 = 1266) and C_1 in Class E (rating = 1147), while (when playing C_4 against C_2), C_4 is in Class D (rating = 1200) and C_2 is in class E (rating = 1115). Also C_3 won as red and as white when playing against C_1 and C_4 won as red 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 significantly better than the 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 = 1475) and C_1 in class D (rating = 1264), while table 6 shows that C_4 won as red and as white when playing against C_1 using a starting position where all pieces are in their original positions.

B. Results Using Round Robin Players

The same procedure was also used to play a league between Blondie24-RR, Blondie24-RR1Ply, Blondie24-RR2Ply and Blondie24-RR3Ply. The results are shown in tables 7 thru 9 and figure 3, where the 1ply represents Blondie24-RR1Ply, while 2ply represents Blondie24-RR2Ply, 3ply represents Blondie24-RR3Ply and 4ply represents Blondie24-RR.

Table 7: Number of wins (for the row player) out of 258 games for the round robin players.

	1ply	2ply	3ply	4ply	Σ wins
1ply	-	28	20	14	62
2ply	32	-	29	21	82
3ply	42	34	-	27	103
4ply	57	46	39	-	142

Table 8: Number of draws (for the row player) out of 258 games for the round robin players.

	1ply	2ply	3ply	4ply	Σ draws
1ply	-	26	24	15	65
2ply	26	-	23	19	68
3ply	24	23	-	20	67
4ply	15	19	20	-	54

Table 9: Number of loses (for the row player) out of 258 games for the round robin players.

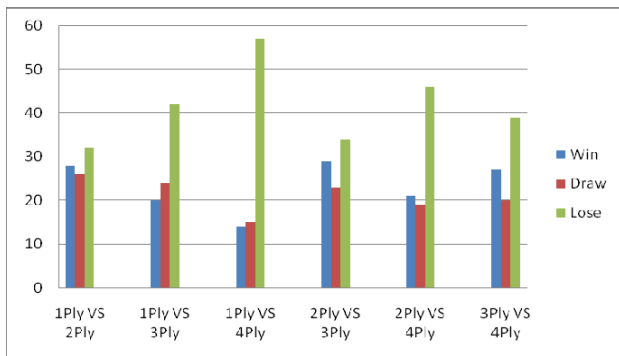|	1ply	2ply	3ply	4ply	Σ losses
1ply	-	32	42	57	131
2ply	28	-	34	46	108
3ply	20	29	-	39	88
4ply	14	21	27	-	62

Figure 3: Results of Playing a League between 1Ply, 2Ply, 3Ply and 4Ply.

It is clear from tables 7 and 9 that the total number of wins increases and the total number of losses decreases when the ply depth increases. Therefore, increasing the ply depth leads to a superior player. Table 10 shows the mean and the standard deviation of the players' ratings after 5000 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 the players after 5000 orderings.

	Mean	SD	Class
1Ply	1187.79	28.86	E
2Ply	1200.74	27.55	D
1Ply	1160.17	28.15	E
3Ply	1252.67	26.84	D
1Ply	1256.00	27.71	D
4Ply	1450.51	26.58	C
2Ply	1194.62	29.30	E
3Ply	1212.04	27.98	D
2Ply	1335.38	28.72	D
4Ply	1440.84	27.43	C
3Ply	1348.31	29.24	D
4Ply	1495.93	27.91	C

Table 11: Summary of Wins/Loses for 1Ply, 2Ply, 3Ply and 4Ply when not using two-move ballot.

		2Ply	3Ply	4Ply
1Ply	Red	Lost	Lost	Lost
	White	Lost	Lost	Lost
2Ply	Red	-	Lost	Lost
	White	-	Lost	Lost
3Ply	Red		-	Lost
	White		-	Lost

The results in table 10, obtained using 5000 different orderings for the 86 games (obtained using the two-move ballot) show that increasing depth by one increases the performance of the checkers player as Blondie24-RR2Ply is better than the Blondie24-RR1Ply, Blondie24-RR3Ply is better than Blondie24-RR2Ply and Blondie24-RR is better than Blondie24-RR3Ply. By using the average value for the standard rating formula the results (when playing Blondie24-RR2Ply against Blondie24-RR1Ply) put Blondie24-RR2Ply in class D (rating = 1201) and Blondie24-RR1Ply in Class E (rating = 1188). Playing Blondie24-RR3Ply against Blondie24-RR2Ply puts Blondie24-RR3Ply in class D (rating = 1212) and Blondie24-RR2Ply in class E (rating = 1195). Finally, when playing Blondie24-RR against Blondie24-RR3Ply, puts Blondie24-RR in class C (rating = 1496) and Blondie24-RR3Ply in class D (rating = 1348).

Table11 shows that Blondie24-RR2Ply won as red and drew as white when playing against Blondie24-RR1Ply using a starting position where all pieces are in their original positions. Also Blondie24-RR3Ply won as red and as white when playing against Blondie24-RR2Ply and Blondie24-RR won as red and as white when playing against Blondie24-RR3Ply.

The results shown in table 10 also show that increasing depth by two increases the performance of the checkers player as Blondie24-RR3Ply and Blondie24-RR are significantly

better than the Blondie24-RR1Ply and Blondie24-RR2Ply respectively, and by using the average value for the standard rating formula, the results (when playing Blondie24-RR3Ply against Blondie24-RR1Ply) put Blondie24-RR3Ply in class D (rating = 1253) and Blondie24-RR1Ply in class E (rating = 1160), while (when playing Blondie24-RR against Blondie24-RR2Ply) puts Blondie24-RR in Class C (rating = 1441) and Blondie24-RR2Ply in class D (rating = 1335). Also Blondie24-RR3Ply won as red and as white when playing against Blondie24-RR1Ply and Blondie24-RR won as red and as white when playing against Blondie24-RR2Ply using a starting position where all pieces are in their original positions.

VI. CONCLUSIONS

The experiments we have carried out produced many evolutionary checkers players, using different depths of ply during learning. 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.

An interesting point to note from the results is that increasing the depth level by one will give different performances depending on the level number. For example, the results in tables 2 and 7 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 level from three to four, which is better than increasing the depth from one to two and from two to three. One can say that the increasing of the ply depth will increase the computational cost of evolving evolutionary checkers, 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 ply possible.

VII. REFERENCES

[1] Turing, A. M., Computing machinery and intelligence, Mind, Vol.59, 1950, 433-460.
[2] Samuel, A. L., Some studies in machine learning using the game of checkers, IBM Journal on Research and Development, 1959, 210-229. Reprinted in: E. A. Feigenbaum and J. Feldman, eds., Computer and Thought, NY: McGraw-Hill, 1963. Reprinted in: IBM Journal on Research and Development, 2000, 207-226.
[3] Newborn M., Kasparov vs. Deep Blue, Computer Chess Comes of Age. New York: Springer-Verlag, 1997.
[4] Campbell M., Hoane A.J., and Hsu F.H., "Deep Blue," Artificial Intelligence, Vol. 134, 2002, 57-83.
[5] Schaeffer, J., One jump ahead: Computer Perfection at Checkers. New York: Springer, 2009.
[6] Fogel D. B., Blondie24 Playing at the Edge of AI, United States of America: Academic Press, 2002.
[7] Samuel, A. L., Some studies in machine learning using the game of checkers II – recent progress, IBM Journal on Research and Development, 1967, 601-617. Reprinted in: D. L. Levy, ed., Computer games, NY: Springer-Verlag, 1988, 366-400.
[8] Kaelbling, L. P., Littman M. L. and Moore A.W., Reinforcement Learning: A Survey, Journal of Artificial Intelligence Research, Vol. 4, 1996, 237-285.
[9] Sutton, R. S. and Barto, A. G., Reinforcement learning, MA: MIT Press, 1998.
[10] Vrakas D., Vlahavas I. PL., Artificial intelligence for advanced problem solving techniques, Hershey, New York, 2008.
[11] Mitchell, Tom M., Machine learning, McGraw-Hill, 1997.
[12] Fogel, D. B., Evolutionary Computation: Toward a new philosophy of machine intelligence (Second edition). NJ: IEEE Press, 2000.
[13] Kendall G. and Su Y., Imperfect Evolutionary Systems, IEEE Transactions on Evolutionary Computation, 2007, Vol. 11, 294-307.
[14] Levene M., and Fenner T. I., "The effect of mobility on minimaxing of game trees with random leaf values," Artificial Intelligence, Vol. 130, 2001, 1-26.
[15] Nau D. S., Lustrek M., Parker A., Bratko I. and Gams M., "When Is It Better Not To Look Ahead?," Artificial Intelligence, Vol. 174, 2001, 1323-1338.
[16] Smet P., Calbert G., Scholz J., Gossink D., Kwok H-W, and Webb M., "The Effects of Material, Tempo and Search Depth on Win-Loss Ratios in Chess" AI 2003: Advances in artificial intelligence, Lecture Notes in Computer Science, Vol. 2903 , 2003, 501-510.
[17] Bettadapur P., and Marsland T.A., "Accuracy and savings in depth-limited capture search,", International Journal of Man-Machine Studies, Vol. 29, 1988, 497 – 502.
[18] Runarsson, T.P. and Jonsson, E.O, Effect of look-ahead search depth in learning position evaluation functions for Othello using ε–greedy exploration, In Proceedings of the IEEE 2007 Symposium on Computational Intelligence and Games (CIG'07), Honolulu, Hawaii, 2007, 210 - 215.
[19] Chellapilla K. and Fogel, D. B., Anaconda defeats hoyle 6-0: A case study competing an evolved checkers program against commercially available software. Congress on Evolutionary Computation, La Jolla Marriot Hotel, La Jolla, California, USA, 2000, 857-863.
[20] Fogel D. B. and Chellapilla K., Verifying anaconda's expert rating by competing against Chinook: experiments in co-evolving a neural checkers player. Neurocomputing, Vol. 42, 2002, 69-86.
[21] Chellapilla K. and Fogel D.B., Evolution, Neural Networks, Games, and Intelligence," Proceedings of the IEEE, Vol. 87, 1999, 1471-1496.
[22] Chellapilla K. and Fogel D. B., Evolving an expert checkers playing program without using human expertise, IEEE Transactions on Evolutionary Computation, Vol. 5, 2001, 422-428.
[23] Chellapilla K. and Fogel D. B., Evolving neural networks to play checkers without relying on expert knowledge. IEEE Transactions on Neural Networks, Vol. 10, 1999, 1382-1391.
[24] Al-Khateeb B. and Kendall G., Introducing a Round Robin Tournament into Blondie24, In Proceedings of the IEEE 2009 Symposium on Computational Intelligence and Games (CIG'09), Milan, Italy, 2009, 112-116.
[25] Hughes E., Piece Difference: Simple to Evolve, The 2003 Congress on Evolutionary Computation (CEC 2003), Vol. 4, 2003, 2470 – 2473.
[26] Al-Khateeb B. and Kendall G., "The Importance of a Piece Difference Feature to Blondie24", In Proceedings of the the 10th Annual Workshop on Computational Intelligence (UK2010), Colchester, UK, 2010, 1-6.