Introducing a Round Robin Tournament into Evolutionary Individual and Social Learning Checkers

Belal Al-Khateeb
School of Computer Science
Al-Anbar University
Ramadi, Iraq
belal@computer-college.org

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

***Abstract*—** In recent years, much research attention has been paid to evolving self-learning game players. Fogel's Blondie24 is a demonstration of a real success in this field; inspiring many other scientists. In this paper, artificial neural networks are used as function evaluators in order to evolve game playing strategies for the game of checkers. We introduce a league structure into the learning phase of an individual and learning system based on the Blondie24 architecture. We show that this helps eliminate some of the randomness in the evolution. The best player we evolve is tested against an implementation of an evolutionary checkers program, and also against a player, which utilises the proposed round robin tournament and finally against an individual and social learning checkers program. The results are promising, suggesting many other research directions.

I. INTRODUCTION

Designing automated computer game playing programs has been of interest since the 1950s [1,2], and is still of interest today, being further motivated by successes such as Deep Blue in 1997 [3,4], which defeated Garry Kasparov, considered the best chess player that ever lived. What is not so well known is that Chinook became the first machine to be a world champion when it defeated the previous checkers world champion (Marion Tinsley) [5]. In our view Chinook and Deep Blue are significant achievements, both requiring considerable effort by the teams behind them.

Many important aspects of artificial intelligence are encompassed by automated game playing. These include knowledge representation, search and machine learning. Incorporating human knowledge into the algorithm is often used by the way of an evaluation function and a database of opening and end game sequences.

Blondie24 [6] is an evolutionary algorithm, presented by David Fogel and Kumar Chellapilla, which is able to play the game of checkers at the expert level. One of the main aims of their work was to inject as little expert (human) knowledge as possible. By only using the number, type and positions of pieces on the checkers board, the algorithm utilises feedforward artificial neural networks to evaluate board positions. One of the core features in the design of Blondie24 was to have the program learn, through self play. This is an alternative approach to using human knowledge as an input to the system and utilising a human designed evaluation function to judge the quality of a given board position. The architecture of Blondie24 is shown in Figure1 [6].

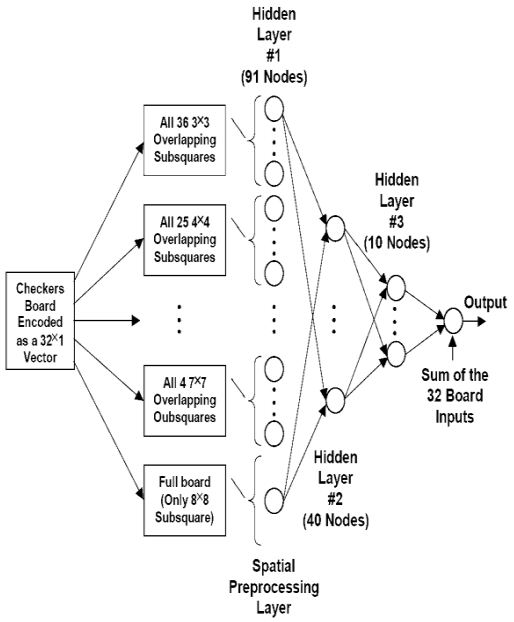

Figure 1: Blondie24 Architecture [6]

The success of Blondie24 inspired the motivation of this work, where the experiments described in this paper are based on an evolutionary algorithm that was used as the basis for Blondie24 and also based on an individual and social learning algorithm that was used by Al-Khateeb and Kendall [7]. We

978-0-7695-4593-6/11 $26.00 © 2011 IEEE
DOI 10.1109/DeSE.2011.10

hypothesise that the introduction of a round robin tournament to an evolutionary individual and social learning checkers will evolve a superior player. It was not our intention to produce a world checkers champion. Rather, we want to study the effectiveness of such introduction as a machine learning approach to game playing.

Our experiments demonstrate that the introduction of the round robin tournament to the evolutionary individual and social learning checkers produced a stronger checkers player. This is achieved through playing against a set of players.

The rest of the paper is organised as follows; in section II the nature of the game of checkers together with related work is presented. Two-Move ballot and *t*-test are 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. CHECKERS

This section gives a brief description of Checkers, and then summarises previous approaches to learning to play it. Checkers, sometimes called draughts, is played on an 8x8 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 figure 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 of the players advance to the last row of the board (on the opponent 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 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.

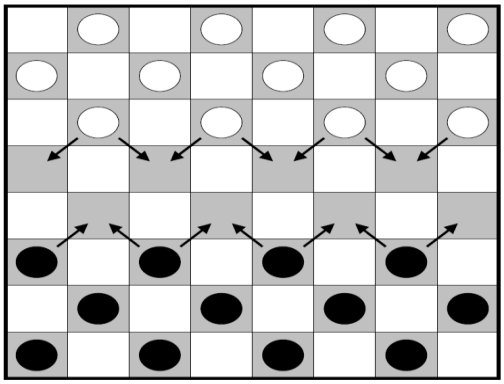

Figure 2: Checkers board (showing possible moves). Black moves first.

Arthur Samuel, in 1954, attempted to illustrate that a computer program could improve by playing against itself by starting work on evolving a checkers player, using an early form of temporal difference learning. Samuel's program adjusted weights for 39 features [2,8]. These features were adjusted during the game using a method we now refer to as reinforcement learning, instead of tuning the weights manually [9-12]. Samuel found that piece difference 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. Due to memory limitations, Samuel only used 16, out of the 38 features, in his evaluation function. 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 the same 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 [8].

Starting in 1989, Jonathan Schaeffer, and colleagues, at the University of Alberta, designed a checkers program called Chinook [10,11]. 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 [12].

A further motivation was to produce the world's best checkers player. This was done by using an evaluation function, which includes several features and all were based on human expertise, including grand masters. Like Samuel's program, the main feature in Chinook's evaluation function is the piece count, where each piece on the board takes 100 points. The next most important feature is the king, which takes a value that is greater than a regular checker by 30 percent, except when the king is trapped (a trapped king cannot move because it will be taken by the opponent), when it takes the same value as a regular checker. Another feature that is important to Chinook's evaluation function is the *runaway* checker (a clear path for a checker to become a king, without any obstacles), which takes a value of 50 points in addition to its previous value, and subtracts three points for each move required to advance the checker to be a king. Other additional features in the evaluation function included the "*turn*", "*mobile kings*" and the "*dog hole*" (a checker that is trapped by its

opponent and cannot be moved). Each one of those features was assigned a different weight indicating its importance.

The summation of each term provided an overall assessment of the board for a particular game state. Initially, Schaeffer gave initial values to the weights and then hand tuned them when he found an error (e.g. an obviously incorrect move being made) or when a Chinook move led to a losing position.

Chinook also utilised opening and end game databases to further enhance its ability. Initially Chinook's opening game database contained 4,000 sequences. Later it contained more than 40,000. The end game 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 end game database contained all six piece end sequences, allowing it to play perfectly from these positions.

In 1989 Chinook, with a four piece end game database [12], won the computer Olympiad. Later, with its final six piece end game database, together with its evaluation function modified by a *fudge* factor [11,13], 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, he died the following year) [11]. In 1996 Chinook retired with rating at 2,814.

The building of the open/end game databases ultimately led Schaeffer to achieve his initial motivation (solving the game of checkers) [14]. Perfect play by both sides leads to a draw.

Blondie24 represents a landmark in evolutionary learning by attempting to design a computer checkers program, injecting as little expert knowledge as possible [6,20-24]. 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 initially (as their weights were initialized randomly), and gradually improved 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 [20,21]. 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.

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 [6,23], Evan Hughes [26], and Al-Khateeb and Kendall [27]. The results showed 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 [28] show that the look-ahead depth in important to the evolution of a checkers player.

Al-Khateeb and Kendall [25] enhanced Blondie24 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).

Al-Khateeb and Kendall [7] enhanced Blondie24 by introducing an individual and social learning. We also utilise this work in this paper (see Section V).

III. TWO-MOVE BALLOT and *T*-TEST

When human experts play checkers the game often ends in a draw. To overcome this, the Two-Move Ballot (introduced in the 1870s [5]) is used to make the games more competitive.

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 use 5000 different orderings for the 86 two-ballot move (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 the mean value of the standard rating formula puts him in a level that is higher than his opponent. The determination of the players' level is done 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.

Also a student *t*-test with the following settings: unequal variances, $\alpha = 0.05$, and one-tail test, will be used in order to test the hypothesis, determine whether two players are statistically the same, where the data source is the 5000 different orderings for the 86 two-ballot move games. It is worth mentioning that any two players are statistically the same if the value (called *p* value) obtained from the *t*-test is less than alpha.

Table 1: Standard Rating System Categories [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

IV. EXPERIMENTAL SETUP

For the purpose of investigating our hypothesis (will using a round robin tournament together with individual and social learning produce a strong evolutionary checkers player?), an evolutionary checkers player, based on the same algorithm that was used to construct Blondie24, was firstly implemented in order to provide a platform for our research. Our implementation has the same structure and architecture that Fogel utilised in Blondie24. We call this player C_0. Then we implemented an individual and learning player with a round robin tournament using the following algorithm;

1- Initialise a random population of 30 neural networks (players) sampled uniformly [-0.2,0.2] for the weights.
2- Each player has its associated self-adaptive parameter, initialised to 0.05.
3- Initialise *M* (frequency of individual learning) and *N* (frequency of social learning).
4- Play each player in the current population against all other players using round robin tournament.
5- For each game, the player receives a score of +1 for a win, 0 for draw and -2 for a loss.
6- Games are played until either side wins, or until one hundred moves are made by both sides, in which case a draw is declared.
7- If the generation number is exactly divisible by *M* and not by *N* then
   - Select the best player(s) with the highest score (if two or more players have equal scores, we will select all those players) and copy them to the social pool.
   - Select the best 15 players and mutate them to get 15 offspring using equations (1) and (2).
8- If the generation number is exactly divisible by *N* then for all players, *i*, do:
   - Normalize the individual scores (values between 0 and 1) for all the players using the following equation:

$$V_i = \frac{(X_i - Min)}{(Max - Min)} \qquad (3)$$

   where V_i is the normalized value for player *i*, *Min* and *Max* is the lowest and highest score in the current population among all players, X_i is the score of player *i* before being normalized.
   - If the normalised value is 1 and the player is not using a strategy drawn from the pool, then publish the strategy into the pool.
   - If the normalised value is 1 and the player is using a strategy drawn from the pool then do not publish the strategy into the pool but update the strategy's score in the pool.
   - For the rest of the players, there are two cases:-
     1. If the normalised value is between 1 and 0.9, then the player is satisfied with his current strategy and retains it.
     - If the normalised value is less than 0.9, then the player is not satisfied with his current strategy. The player has three options:-
       a- With 1/3 probability, replace the current strategy by copying a new strategy from the pool. Roulette wheel selection is used to select the new strategy from the pool.
       b- With 1/3 probability, replace the current strategy by creating a new random strategy.
       c- With 1/3 probability, retain the current strategy.
9- If the generation number is not exactly divisible by *M* or *N* then

- Select the 15 best players and mutate them to get 15 offspring using equations (1) and (2).

10- Repeat steps 4-9 for *K* generations or for specified time.

The resultant player from the above algorithm is called C_{10}-RR, which is obtained over 140 generations. Our previous efforts to introduce individual and social learning into evolutionary checkers [7] showed that the value of *M*=5 and *N*=10 are the best values for determining when individual and social learning should occur. Therefore we decided to use the same values in the above algorithm.

Our previous efforts to enhance Blondie24 introduced a round robin tournament [25]. We also use this player (Blondie24-RR) to investigate our hypothesis. We also decided to use C_{10}, which is a result of our previous efforts to introduce individual and social learning to an evolutionary checkers [7], to investigate our hypothesis.

V. RESULTS

In order to test the outcome of introducing round robin tournament into the evolutionary phase of C_{10}, C_{10}-RR is set to play against C_0, Blondie24-RR and C_{10}, all using the two-move ballot. The results are shown in tables 2 through 4 and figure3.

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).

Table 2: Results when Playing C_{10}-RR against C_0 using the Two-Move Ballot.

	Opponent:C_0		
	Win	Draw	Lose
C_{10}-RR	49	20	17

Table 3: Results when Playing C_{10}-RR against Blondie24-RR using the Two-Move Ballot.

	Opponent:Blondie24-RR		
	Win	Draw	Lose
C_{10}-RR	41	22	23

Table 4: Results when Playing C_{10}-RR against C_{10} using the Two-Move Ballot.

	Opponent:C_{10}		
	Win	Draw	Lose
C_{10}-RR	35	23	28

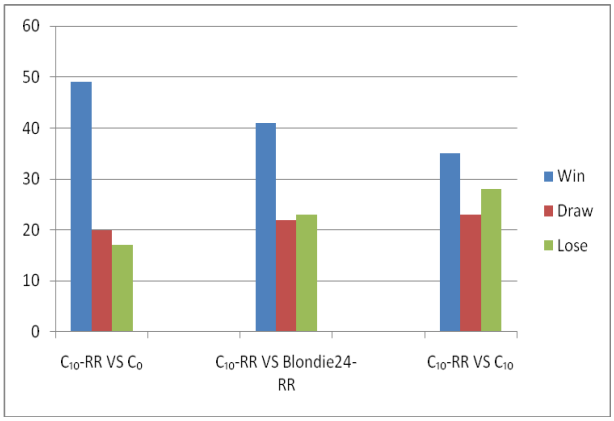

Figure 1: C_{10}-RR against C_0, Blondie24-RR and C_{10}.

Table 5 summarises the results when playing against C_0, Blondie24-RR and against C_{10} using a starting position where all pieces are in their original positions (i.e. no two-move ballot), while table 6 shows the mean and the standard deviation of the players' ratings after 5000 different ordering for the 86 played games.

Table 5: Summary of Wins/Loses When not Using Two-Move Ballot.

		C_0	**Blondie24-RR**	**C_{10}**
C_{10}-RR	**Red**	Won	Won	Won
	White	Won	Won	Won

Table 6: Standard rating formula for playing C_{10}-RR against C_0, Blondie24-RR and against C_{10} after 5000 orderings.

	Mean	**SD**	**Class**
C_{10}-RR	1405.51	27.54	C
C_0	1264.71	26.66	D
C_{10}-RR	1250.44	28.71	D
Blondie24-RR	1171.91	27.61	E
C_{10}-RR	1229.99	29.08	D
C_{10}	1188.00	27.86	E

The results in tables 2 and 6 show that C_{10}-RR is statistically better than C_0 as the results (when playing C_{10}-RR against C_0) put C_{10}-RR in class C (rating = 1405) and put C_0 in class D (rating = 1264), and by using student *t*-test the results show that C_{10}-RR and C_0 are statistically different as the *p* value for the one tail *t*-test is less than alpha.

The results in tables 3 and 6 show that C_{10}-RR is statistically better than Blondie24-RR as the results (when playing C_{10}-RR against Blondie24-RR) put C_{10}-RR in class D

(rating = 1250) and put Blondie24-RR in class E (rating = 1171), and by using student t-test the results show that C_{10}-RR and C_0 are statistically different as the *p* value for the one tail *t*-test is less than alpha.

Finally the results in tables 4 and 6 show that C_{10}-RR is statistically better than C_{10} as the results (when playing C_{10}-RR against C_{10}) put C_{10}-RR in class D (rating = 1229) and put C_{10} in class E (rating = 1188), and by using student *t*-test the results show that C_{10}-RR and C_0 are statistically different as the *p* value for the one tail *t*-test is less than alpha.

As C_{10}-RR is better than C_0 and Blondie24-RR and most importantly is better than C_{10}, then it seems quite appropriate to use the individual and social learning together with a round robin tournament in order to enhance the process of evolutionary checkers.

VI. CONCLUSIONS

This paper has introduced a round robin tournament into an evolutionary individual and social learning checkers program that is based on the Blondie24 architecture. The proposed algorithm shows promising results when tested against an implementation of an evolutionary checkers program and also against two players obtained as a result of the authors' previous efforts to introduce a round robin tournament into Blondie24 and also to introduce individual and social learning.

Based on the results in tables 2 and 6, C_{10}-RR is found to be superior to C_0. Also the result in tables 4 and 6 showed that C_{10}-RR is better than Blondie24-RR. Finally the results in table 5 and 6 showed that C_{10}-RR is better than C_{10}.

Based on the results it would seem appropriate to use a round robin tournament, together with individual and social learning to enhance the evolutionary checkers algorithm.

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] Al-Khateeb B. and Kendall G., Introducing Individual and Social Learning into Evolutionary Checkers, Transactions on Computational Intelligence and AI in Games (TCIAIG), (Under Review), 2011.

[8] 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.

[9] Kaelbling, L. P., Littman M. L. and Moore A.W., Reinforcement Learning: A Survey, Journal of Artificial Intelligence Research, Vol. 4, 1996, 237-285.

[10] Sutton, R. S. and Barto, A. G., Reinforcement learning, MA: MIT Press, 1998.

[11] Vrakas D., Vlahavas I. PL., Artificial intelligence for advanced problem solving techniques, Hershey, New York, 2008.

[12] Mitchell, Tom M., Machine learning, McGraw-Hill, 1997.

[13] Fogel, D. B., Evolutionary Computation: Toward a new philosophy of machine intelligence (Second edition). NJ: IEEE Press, 2000.

[14] Kendall G. and Su Y., Imperfect Evolutionary Systems, IEEE Transactions on Evolutionary Computation, 2007, Vol. 11, 294-307.

[15] 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.

[16] 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.

[17] 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" A1 2003: Advances in artificial intelligence, Lecture Notes in Computer Science, Vol. 2903 , 2003, 501-510.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] Chellapilla K. and Fogel D.B., Evolution, Neural Networks, Games, and Intelligence," Proceedings of the IEEE, Vol. 87, 1999, 1471-1496.

[23] 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.

[24] 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.

[25] 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.

[26] Hughes E., Piece Difference: Simple to Evolve, The 2003 Congress on Evolutionary Computation (CEC 2003), Vol. 4, 2003, 2470 – 2473.

[27] 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.

[28] Al-Khateeb B. and Kendall G., The Importance of look ahead Depth in Evolutionary Checkers, In Proceeding of the 2011 IEEE Congress on Evolutionary Computation (CEC 2011), New Orleans, USA, 2011.