The Importance of a Piece Difference Feature to Blondie24

Belal Al-Khateeb and Graham Kendall

Abstract— **In recent years, significant research attention has been paid to evolving self-learning checkers players. Fogel's Blondie24 has been very successful in this field and has inspired other researchers to further develop this area. In this paper we address the question of whether piece difference is an important factor in the Blondie24 architecture. Although this issue has been addressed before, this work provides a different experimental setup to previous work, but arrives at the same conclusion. Our experiments show that piece difference has a significant effect on learning abilities.**

I. INTRODUCTION

Interest in designing automated computer game playing programs dates back to at least the 1950's [1,2]. Interest today has not diminished and the introduction of Deep Blue in 1997 (after a series of modifications to a previous version of Deep Blue in 1996) represents one of the landmark successes in which a computer program defeated Garry Kasparov, arguably the best chess player ever [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, whereby encoding by hand is utilised in incorporating human knowledge about the game into the computer by means of an evaluation function and a database of opening and end game sequences.

Deep Blue beat Kasparov in 1997 but it is not so well known, that 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 significant achievements, both requiring considerable effort by the teams behind them.

Blondie24 [6] is an evolutionary algorithm that was presented by David B. Fogel and was capable of playing the game of checkers without using human knowledge to introduce special features to the game. By solely using the positions and type of pieces on the checkers board together with a piece difference, the evolutionary program utilises feedforward artificial neural networks to evaluate alternative positions in the game. The core feature in the design of Blondie24 is to make the program learn, through self play, how to play checkers.

Belal Al-Khateeb and Graham Kendall are with The School of Computer Science, University of Nottingham, UK (T:+44 (0) 115 951 4251, F:+44 (0) 115 951 4254; email: bxk@cs.nott.ac.uk, gxk@cs.nott.ac.uk).

This is direct contradiction of the alternative which is to preload it with all the information about how to make good moves and avoid bad ones. The architecture of Blondie24 is shown in Figure 1. The piece difference feature is directly connected to the output node.

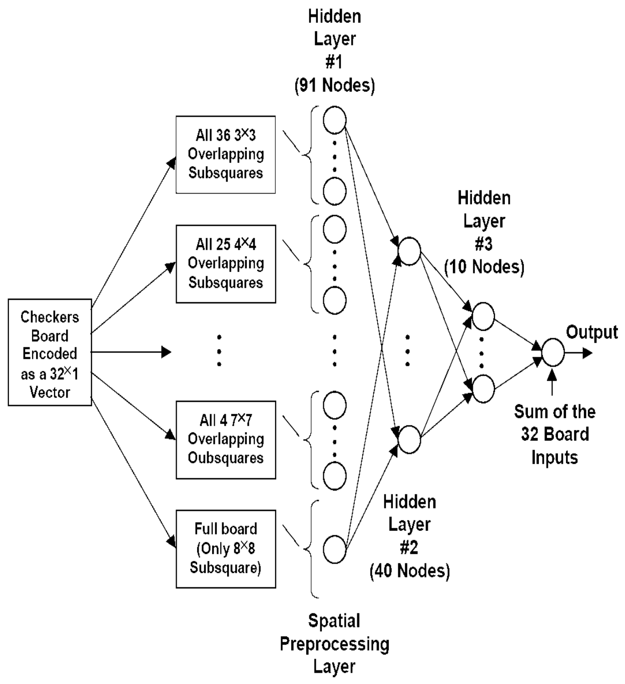
Hidden Layer #1 (91 Nodes)
All 36 3×3 Overlapping Subsquares
All 25 4×4 Overlapping Subsquares
Checkers Board Encoded as a 32×1 Vector
All 4 7×7 Overlapping Oubsquares
Full board (Only 8×8 Subsquare)
Spatial Preprocessing Layer
Hidden Layer #2 (40 Nodes)
Hidden Layer #3 (10 Nodes)
Output
Sum of the 32 Board Inputs

Figure 1: Blondie24 Architecture

Although, there has been a lot of discussion about the piece difference feature used in Fogel's work [6], there has been limited work which has investigated its importance. Fogel in his work on evolving Blondie24 [6] showed that a piece difference feature is important to the Blondie24 design but the neural network provides additional, and useful learning, to Blondie24. Hughes [7] reported some results in a tutorial given at the first IEEE Symposium on Computational Intelligence and Games (CIG2005). Further details are discussed in section IV.

The experiments of Fogel and Hughes about the piece different feature showed that piece difference is important, but the aim of their design was to show that the neural network architecture provides additional learning to the evolved player.

In this paper, a study that investigates the importance of the piece difference feature is conducted in a way that differs from the two studies that were carried out by Fogel and by Hughes. Our study aims to investigate how well the neural network performs without the piece difference.

The designed experiment shows that the piece difference feature is important for the learning process of Blondie24 and this is achieved through playing many games between two players, one with piece difference and the other without it.

The rest of the paper is organised as follows; in section II related work is presented. Blondie24 is discussed in section III. Section IV presents Brunette24. Section V shows our experimental setup for this work. Section VI presents our results and we conclude in Section VII.

II. BACKGROUND

In an attempt to illustrate that a computer program could improve by playing against itself, Arthur Samuel, in 1954 started, working on evolving a checkers player, using an early form of temporal difference learning. Samuel's program evolved weights for 39 features [2,8]. These features were evolved during the game using a reinforcement learning method instead of tuning them manually [9-12]. 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 stayed 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].

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 [6,13,14]. 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 [6,13]. 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. New 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. …*"

When world caliber humans play the game of checkers, it often ends in a draw. To overcome this, and make the games more competitive, Two-Move Ballot is used.

The Two-Move Ballot was introduced in 1870s [15]. 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 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 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).

Blondie24 [16-20], was designed to address Samuel's challenge; to build a machine that could tech itself how to play rather than be told, and to have recognize the important features rather than having to be told. The next section discusses Blondie24 in more details.

Al-Khateeb and Kendall [21] enhanced Blondie24 by introducing a round robin tournament, instead of the randomly choosing the opponents. The results are reported in [21], and we utilise this work in this paper (see Section V).

III. BLONDIE24

Blondie24 represents a landmark in evolutionary learning attempting to design a computer checkers program, injecting as little expert knowledge as possible [6,16-20]. 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 made 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], but this was not the objective of the research; but rather to answer the challenges set by Samuel and Newell (that progress in this area would not be made in anybody's lifetime without using credit assignment problem), which it successfully did. The major difference between Blondie24 and other traditional game-playing programs is in the employment of evaluation functions [16,17]. In traditional game-playing programs, the evaluation functions usually consist 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), $t = \frac{1}{\sqrt{2\times\sqrt{N_w}}}$ = 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).

Blondie24 represents a landmark in evolutionary learning. Even so, 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 playing a series of fourteen matches (seven as red and seven as white) between Blondie24 and a piece-count player [6,19]. The experiment showed that the piece-count player played a weak endgame, because it is unable to see far enough ahead to capture a piece. The games played out until either the game was completed (with one side winning, or a draw being declared due to the number of repeated positions). In the case of a draw an assessment of the outcome was made by examining the piece advantage that one player had over the other, and also by playing out the game using a strong computer program (Blitz98), which played out the remainder of the game and declared a winner.

Of the fourteen games played, two were played to completion, with Blondie24 winning both. For the remaining twelve games, Blondie24 held an advantage in ten games, while the piece-count player held the advantage in two games (see Table 1). By using Blitz98 to play out the twelve incomplete games, Blondie24 got wins in eight games; the piece-count player won one game, while the remaining three games ended in a draw (Table 2).

Table1: Results of Playing 14 Games between Blondie24 and Piece-count Using Material Advantage to Break Tie.

	Piece-count		
	Win	Draw	Lose
Blondie24	12	0	2

Table2: Results of Playing 14 Games between Blondie24 and Piece-count Using Blitz98 to Break Tie.

	Piece-count		
	Win	Draw	Lose
Blondie24	10	3	1

It is clear from Table 1 and 2 that Blondie24 is significantly better than a piece-count player, and by using a standard rating formula, the results suggest that Blondie24 is about 311 to 400 points better than the piece-count player based on material advantage or the final outcome using Blitz98 [6,19].

The results demonstrate that a piece difference feature is important to Blondie24 but the neural network has additional information that is important to learning within Blondie24.

IV. BRUNETTE24

Brunette24 was designed by Evan Hughes as a re-implementation of Blondie24 (this was reported in a tutorial given by Evan Hughes at the first IEEE Symposium on Computational Intelligence and Games (CIG2005). Hughes used the same structure that is used for Blondie24 and evolved Brunette24 by using the same algorithm that is described in section III, the Blondie24 algorithm.

Hughes wanted to investigate the importance of a piece difference feature to the design of Brunette24 by evolving a piece difference heuristic using co-evolution [7].

Hughes used the same experiment as Fogel to show the importance of a piece difference. This was done by playing 1000 games against a simple piece difference player. The evolved piece difference player managed to win 68% of the games, drew 30% and lost 2% (Table 3).

Also, to measure the success of the evolved piece difference player, Hughes played 1000 games against xcheckers, which is free software that can be downloaded from http://arton.cunst.net/xcheckers. The evolved piece difference player won 22% of the games, drew 66% and lost 12% (Table 4).

Table3: Results of Playing 1000 Games between the Evolved Piece Count player and Piece-count player.

	Piece-count		
	Win	Draw	Lose
Evolved Piece Count	680	300	20

Table4: Results of Playing 1000 Games between the Evolved Piece Count player and xcheckers.

	Xcheckers		
	Win	Draw	Lose
Evolved Piece Count	220	660	120

The results in Table 3 show that the evolved piece count player is significantly better than the piece count player and by using a standard rating formula, the results suggest that the evolved piece difference player is about 528 points better than piece difference player. While applying the standard rating formula to the results in Table 4 shows that the evolved piece difference player is about 80 points better than xcheckers (this was reported in a tutorial given by Evan Hughes at the first IEEE Symposium on Computational Intelligence and Games (CIG2005).

These results, like Fogel's, also show that a piece difference feature is important.

V. EXPERIMENTAL SETUP

For the purpose of investigating our hypothesis, Blondie24 was first re-implemented in order to provide a platform for our research. Our re-implementation has the same structure and architecture that Fogel utilised in Blondie24.

Two implementations were done, one with a piece difference feature, which is called Blondie24-RPD, while the other is without a piece difference feature and is called Blondie24-R.

Our previous efforts to enhance Blondie24 introduced a round robin tournament [21]. We decided to use the resultant player (Blondie24-RR) to show the importance of the piece difference feature. This is done by implementing a player which is the same as Blondie24-RR, but, does not include a piece difference feature. This player is called Blondie24-RRNPD.

VI. RESULTS

To measure the effect of a piece difference feature in Blondie24, Blondie24-RPD was played against Blondie24-R by using the idea of a two-move ballot (see Section II). We play all of the 43 possible games, both as red and white. This gives a total of 86 games. The games were played until either one side wins or a draw is declared after 100 moves for each player. Table 5 and Figure 2 show the results.

The same procedure was also used to play Blondie24-RR against Blondie24-RRNPD. The results are shown in Table 6 and Figure 3.

Table 5: Results when Playing Blondie24-RPD against Blondie24-R using the Two-Move Ballot

	Opponent:Blondie24-R		
	Win	Draw	Lose
Blondie24-RPD	59	14	13

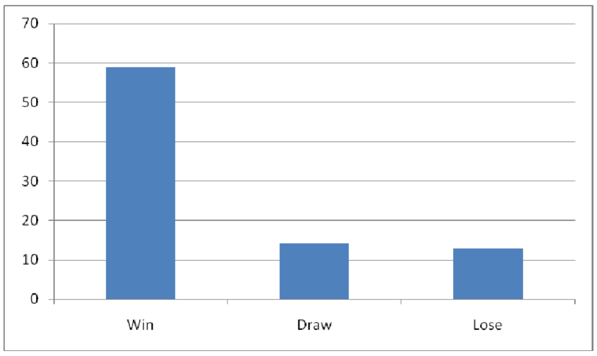

Figure2: Blondie24-RPD against Blondie24-R.

Table 6: Results when Playing Blondie24-RR against Blondie24-RRNPD using the Two-Move Ballot

	Opponent: Blondie24-RRNPD		
	Win	Draw	Lose
Blondie24-RR	61	16	9

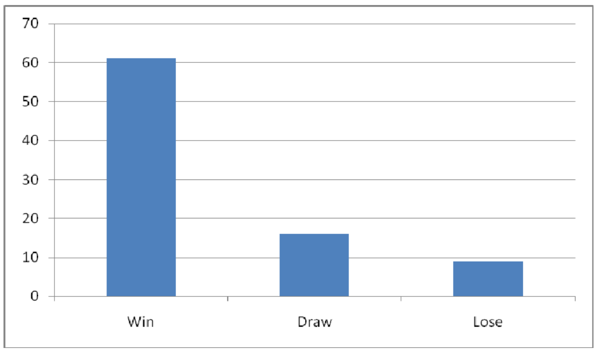

Figure3: Blondie24-RR against Blondie24-RNPD

VII. CONCLUSIONS AND FUTURE WORKS

The experiments we have carried out differ from those of Fogel and Hughes, yet our conclusion is the same.

The results shown in Table 5, obtained using the two-move ballot show that a piece difference feature is important to the design of Blondie24 as Blondie-RPD got 59 wins (from 86 games) over Blondie24-R, while Blondie24-R got only 13 wins. There were 14 draws. It is clear that Blondie24-RPD is significantly better than the Blondie24-R, and by using a standard rating formula, the results suggest that Blondie24-RPD is about 428 points better than Blondie24-R.

The results shown in Table 6 also shows that a piece difference feature is important to the design of Blondie24 as Blondie-RR got 61 wins, 16 draws and 9 losses. It is clear that Blondie24-RR is significantly better than the Blondie24-RRNPD, and by using a standard rating formula, the results suggest that Blondie24-RR is about 475 points better than Blondie24-RRNPD.

Using the results from our experiments and those of Fogel's and Hughes', we can conclude that a piece difference feature is important to the design of Blondie24. Of course, the neural network is also an important element of the whole design but the results presented here demonstrate a simple feature is able to significantly improve the overall playing strength.

Now that the importance of piece difference has been shown in the design of a Blondie24 framework, our future work is planning to investigate if the depth of the search is also an important element. We suspect that it is, but we would like to investigate this aspect of the framework.

VIII. 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: challenging human supremacy in checkers. New York: Springer-Verlag, 1997.

[6] Fogel D. B., Blondie24 Playing at the Edge of AI, United States of America: Academic Press, 2002.

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

[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] Schaeffer, J., One jump ahead: Computer Perfection at Checkers. New York: Springer, 2009.

[16] 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.

[17] 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.

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

[19] 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.

[20] 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.

[21] 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.