AN INTEGRATED OF INDIVIDUAL AND SOCIAL LEARNING IN A CO-EVOLUTIONARY APPROACH TO THE GAME OF CONNECT-FOUR

RAZALI YAAKOB
HAIRULNIZAM MAHDIN

An Integration of Individual and Social Learning in a Co-evolutionary Approach to the Game of Connect-Four

Razali Yaakob
Department of Computer Science,
Faculty of Computer Science and IT,
Universiti Putra Malaysia,
43400 UPM Serdang,
Selangor, Malaysia.
Tel.: 603-89466523
Email: razaliy@fsktm.upm.edu.my

Hairulnizam Mahdin
Faculty of Information Technology and Multimedia,
Universiti Tun Hussein Onn Malaysia,
86400 Parit Raja, Batu Pahat,
Johor, Malaysia.
Tel.: 607-4538044
Email: hairuln@uthm.edu.my

ABSTRACT

In this paper, we investigate an integration of individual and social learning, utilising co-evolutionary neural networks. Individual learning takes place by playing copies of a player against itself. Social learning allows poor performing players to learn from those players, which are playing at a higher level. The networks are evolved via evolutionary strategies with the network output being used as input to a minimax search tree. Our experiments show that learning is taking place at the 99% confidence level. In terms of performance, the co-evolutionary neural network player has the ability to block two adjacent stones of an opponent.

1. INTRODUCTION

Game playing, as a testbed for investigating artificial intelligence techniques has a long history, and some notable results have been achieved, e.g. Deep Blue in chess [1, 2, 3], Chinook in checkers [4, 5], Victoria (Go-moku) [6], Logistello (Othello) [7, 8], TD-Gammon [9] and Neurogammon [10] (Backgammon), and Connect-Four [11, 12, 13]. Whilst some of these are, arguably, not artificial intelligence (e.g. is Deep Blue intelligent?). The results of these achievement are regularly reported in artificial intelligence publications (e.g. [1, 2].

Deep Blue achieved world champion status when, in 1997, it beat Garry Kasparov [1]. Deep Blue executed sophisticated search algorithms to analyse up to 200 million positions per second by utilising custom-built hardware [14].

Chinook, developed by Jonathan Schaeffer's team at The University of Alberta, won the world checkers title in 1994 [4, 5]. Chinook used an opening and endgame database together with an extensive checkers knowledge base.

In contrast, Fogel and Chellapilla developed a checkers program, which did not rely on human expert knowledge [15,16]. In [15, 16], the program learned checkers strategy using a co-evolutionary approach without utilising any pre-programmed knowledge. The neural networks play against themselves for a number of generations. At each generation, the players only receive points based on whether they have won, lost, or drawn. Without any expert knowledge, Fogel and Chellapilla have demonstrated that a program can learn to play a game and reach the level of a human expert.

A learning methodology that does not rely on human expertise is the aim of the work we present here. The objective of this research is to investigate an integration of individual and social learning in co-evolutionary neural networks for the game of Connect-Four. A feed forward neural network player is played against itself, and an evolutionary strategy is used to evolve these networks. We call this individual learning. After a period of individual learning we allow the players to "learn" from one another. We call this social learning.

The overall aim is to evolve a neural network, which is able to play a competitive game of Connect-Four. The output of the neural network is used as an evaluation of the current game position and is used at the leaf nodes of a minimax tree.

In real-life, humans have a variety of techniques in order to develop strategies to defeat other humans. Humans can improve their strategy by themselves or through the experience of competing against other humans. Humans can also copy strategies from a better player and develop their own strategy based on this copy.

Fogel and Chellapilla showed in [15, 16] that an automated player can learn to play checkers by competing against other players. However, none of these players copied their strategy from other, superior, players. In this work, we give an opportunity for a player to learn to play the game through its own experience and via the experience of others.

According to Vriend [17] and Tesfatsion [18], in the context of *agent-based computational economics,* in individual learning, the agents learn exclusively from its own experience, and in social learning, the agents learn from the experience of other agents.

The techniques we present in this paper are motivated from [19] with some minor modifications. In [19], a simulated stock market uses co-evolving neural networks, which are integrated with individual and social learning. The results show that the artificial stock traders are better than a baseline buy-and-hold strategy.

In this paper, the individual learning is similar to a (1+1)ES [20], where a single parent will mutate and produce a single offspring, and they will compete against each other for survival to the next generation. Meanwhile, we create a central pool to capture all the best players at the end of each period of individual learning (in our

experiment this is at the end of every 1000th generation). This pool can be used by poor players to provide them with a better strategy from which to start learning again.

Figure 1 shows the general structure of individual and social learning in this work. The details of each phase will be discussed in the next section. Based on Figure 1, we start by instantiating a population of random neural networks and then enter an individual learning phase. The players will play in pairs for a period of time before entering social learning. In social learning players can compare their strategy with the players in the central pool and either continue with their current strategy or copy a strategy from the central pool, or generate a new random strategy.

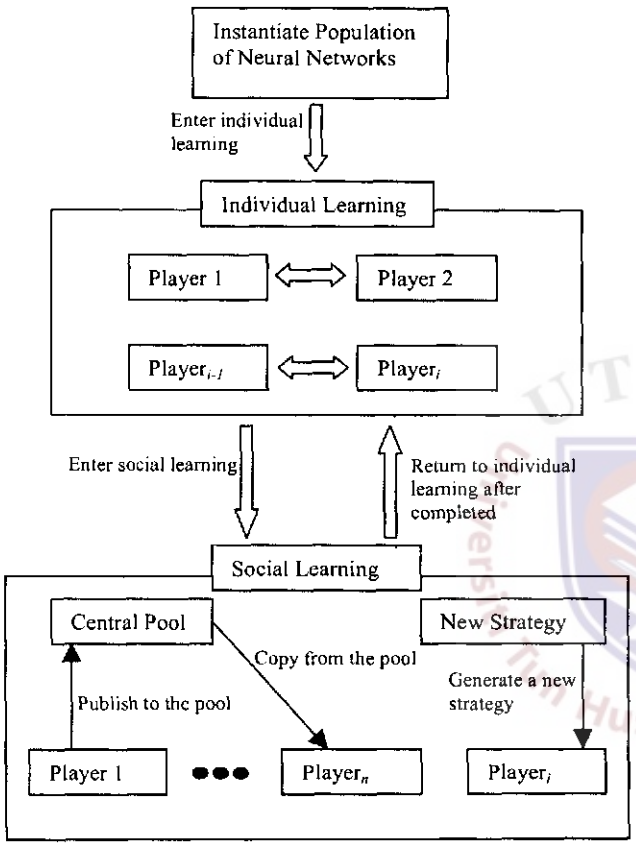

Figure 1: General structure of individual and social learning

Connect-Four was introduced by Milton Bradley in 1974 and known as "The Captain's Mistress" [21]. Allen [11] and Allis [12, 13] solved this game in 1989.

Allen uses brute-force, depth-first search with alpha-beta pruning, a database of prior-positions, and a heuristic of killer-move prediction to play the game. Allen also assigned "tweaking" to the program using expert knowledge.

The VICTOR, Connect-Four program [12, 13], uses a combination of strategic rules, depth-first search and conspiracy-number search [22, 23] to solve the program. At the start of development VICTOR had no search techniques. Initially, nine strategic rules were used simultaneously, but combinations were also allowed depending on the situation, and these are used to resolve threats from the opponent. Without any searching, i.e. it depends solely on knowledge, VICTOR easily solved a small board game, 4x4, 6x4, 4x5, 6x5, 4x6 and 6x6. After a combination of depth-first and conspiracy-number search was added to the program, VICTOR easily wins when playing as a white and drawing as a black on a 7x6 board.

Further research on Connect-Four can be found in [24, 25]. In [24], Neural Connect 4, an automated Connect-Four player, was trained to play the game from four different strategies, i.e. Naïve 4, Constructor/Destroyer (CD), Constructor/Destructor+ (CD+), and Search 4, which had been saved in a database and all these four strategies are used to test the performance of Neural Connect 4. The backpropagation algorithm in Neural Connect 4 beat all four strategies.

A simulated tournament is presented in [25]. It uses minimax to play the game and the results show that the program can easily win when playing as the first player on Connect-Three[1]. However, the performance was not so good when a combination of minimax search and heuristic-based (hyper-graph, weighting strategy) search were used.

2. RULES OF CONNECT-FOUR

Connect-Four is a board game (although often presented as a vertical board so that players can view both sides) with the standard size being seven columns and six rows. The aim of the game is to get four stones connected in a row (vertically, horizontally or diagonally).

This game is played between two players, normally called white and black. White always starts and alternately places a stone on any available column. A player will choose any available column and their token will be placed at the lowest square of a chosen column (the vertical board, due to the effects of gravity, ensure this).

Once a stone has been placed on the board, it will never be removed (or moved) until the end of the game. The game ends when four stones with the same colour form a line (horizontally, vertically or diagonally). Figure 2 shows a board where white has won the game. The game is considered drawn if neither player can make a move that can lead to a win.

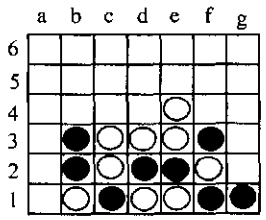

Figure 2: White wins the game as four of its stones are connected diagonally.

Connect-Four is a very simple game, but contains a lot of tactical elements. Players can win the game if they can arrive at one of two situations, i.e. two open lines or a forced open line. Two open lines are where the opponent cannot block the opponent as there are two positions that can be played which lead to a win. Figure 3 shows an example of two open lines. The black player cannot block both b1 and f1, and the white player is guaranteed to win. To avoid this situation, the opponent must block any two connected stones.

[1] Similar to Connect-Four, but only three stones in a row are needed to win the game.

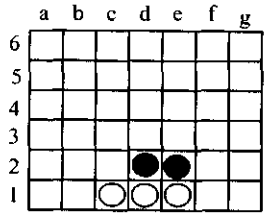

Figure 3: An example of two open lines.

Forced open lines are where the opponent does not have any choices but to block the current three stones in row, but that same move will give the opponent a chance to make another connection. Figure 4 shows an example of forced open lines.

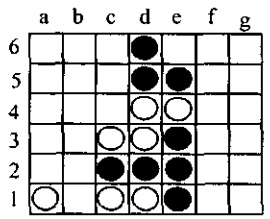

Figure 4: An example of forced open lines (black to play).

In order to block the white player from making four stones connected in row, the black player must play at b1. However, this move will give white a move that gives a line of four stones from a1 to d4.

3. EXPERIMENTAL SETUP

The 42 squares on the board are used as input to a neural network. The inputs are –1, 0 or +1, which indicates a black stone, an empty square or a white stone respectively. A hyperbolic tangent (tanh, bounded by ±1) function is chosen as a non-linear function for each hidden and output node.

The neural network consists of 42 nodes in the input layer, a single hidden layer with 14 nodes and an output node (in total 602 weights). The number of hidden nodes (14) was chosen to be about one third of the number of input nodes. This appears to be a good rule of thumb in the absence of any other data.

All weights in the neural network are mutated at each generation using an evolutionary strategy. Each weight has an associated self-adaptive parameter, σ, which serves to control the step size of the search for the mutated parameters of the neural network [15].

Initially, a Gaussian random generator is used to generate all the weights. The self-adaptive parameter and the weights are then adapted by

$$\sigma'_j = \sigma_j . exp(\tau . N_j(0,1)), \quad (1)$$

$$w'_j = w_j + \sigma'_j . N_j(0,1), \quad j = 1, ..., N_w \quad (2)$$

Where:

N_w is a number of weights, w_j, in the neural network

$N_j(0,1)$ is a standard Gaussian random variable resampled anew for every j

τ is a learning rate, which is $\tau = (2N_w^{0.5})^{-0.5}$

In fact, formula (1) and formula (2) are taken from [15, 16, 26, 27, 28].

3.1 Individual Learning

All players (10 in this work) go through individual learning and play in pairs, e.g. player 1 plays against player 2, player 3 against player 4 and so on. In fact, player 2 is just a copy of player 1 with some mutation. Likewise, 3/4, 5/6, 7/8, 9/10 are, essentially the same player. All these players will play and mutate over 200,000 generations with social learning (see below) occurring at every 1000th generations.

In individual learning, a single parent will mutate and produce an offspring, and they will play against each other for survival to the next generation. The same players will be evolved until end of one interval. Therefore, we keep evolving the same player and this player learns to play the game via just its own experience.

Each player will play two games, i.e. as white and black, and receives points based on whether they win, lose or draw, +2, -1 and 0 respectively. The winning score, +2, is a bonus for the winner and the losing point, –1, is a penalty for the loser. Based on these scores, which is used as a fitness measure for each player, the poor player will be eliminated from the game and replaced with the offspring mutated from the superior player. One of the player will be chosen randomly if the result is a draw.

3.2 Social Learning

At the end of one interval (1000 generations), the program will have five superior players to enter social learning. We create a central pool to group all the best players after each interval.

In social learning, the players from individual learning have an opportunity to compare their strategy with the other players in the central pool. The structure of social learning as follows:

1. Each superior player will play against all the others. The same scoring mechanism is used as in individual learning.
2. Rank the players from the highest to the lowest.
3. Normalise each player from 0.0 to 1.0, and based on the normalisation value, V_i, the player might choose to:
   a. If (V_i = 1.0):
      i. If never been published:
         Publish the player into the central pool and set initial score, B_i, to 1.0 for the player.
      ii. If already been published before:
         Do not publish the player, but update (see below) the current score for the player in the central pool.
   b. If (V_i >= 0.9 and V_i < 1.0):
      The player remains for the next generation.
   c. If (V_i < 0.9):
      The player has two probabilities either copy a strategy from the central pool or generate a new random strategy. If the player decides to copy a strategy from the central pool, roulette wheel selection is used to choose a player.

The score for each player in the central pool will be updated at every interval,

$$B_i^{new} = \begin{cases} B_i^{old} - 0.1, & \text{If the player has not been selected for copy} \\ B_i^{old} + 1.0, & \text{If strategy has been copied from the pool and subsequently becomes the best player in a future generation} \end{cases}$$

Where,
B_i is a score for each player,
i is a player (i = 0, 1, ..., 9)

If ($B_i^{old} < 0.1$), we will set $B_i^{new} = 0.01$. This still allows a small probability for the player to be selected and avoids the player having a negative value. When the player in the central pool reaches a value of 0.01, we call this player an "old player" in the rest of this paper. All "old players" are retained in the central pool until the end of the experiment.

In our initial experiment, we have tested with variety of decrement values and lowest scores, and we found that 0.1 and 0.01 are the best values to use.

4. EXPERIMENTS AND RESULT

In the experiment, a population of 10 networks played in pairs in the individual learning phase. This lasted for 1000 generations before a period of social learning was entered. The experiment was run for 200,000 generations (200 intervals) and at every 1000th generation, the best player was kept in the central pool and these players were played against each other at the end of the experiment. This resulted in 262 "best" players. The number of best players is not the same as the number of intervals because some intervals might produce more than one best player, depending on the normalised value.

Figure 5 shows the total points for each best player in the central pool after playing against each of the other players when played as white and black, i.e. each player played 261 games in total. The same scoring mechanism as in individual learning is used.

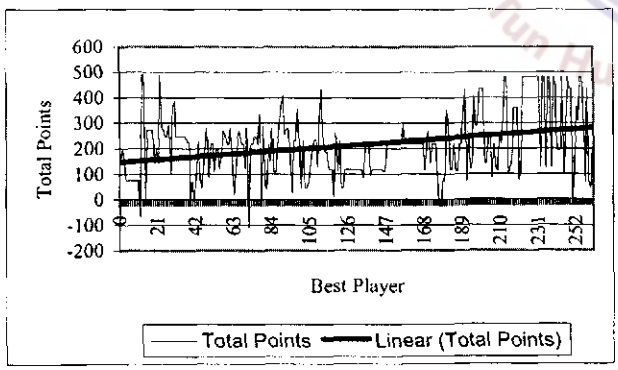

Figure 5: Total points for each best player in the central pool

Based on Figure 5, the regression line indicates that learning has occurred (i.e. players in later generations perform better than players from earlier generations) even though the graph looks chaotic. From 262 best players, a t-test shows that the regression line is statistically significant different from zero at the 99% confidence level.

5. CONCLUSIONS AND FUTURE WORKS

Even though the graph looks chaotic, the positive regression line shows that some learning has occurred at the 99% confidence level.

When any player in the central pool has been chosen for copy, we do not have any mechanism to evaluate whether the player in the pool is better than the player it is going to replace. This, we believe is one of the reason why the graph looks chaotic. That is, a poorer player in the central pool could be chosen for copy.

In social learning, if the normalisation value, V_i, is less than 0.9, the player has two probabilities, copy a strategy from the pool or generates a new strategy. When we create a new strategy, the new strategy should be better than or at least equal with the current strategy. However, in this experiment, we do not evaluate how good the new strategy is before we use it for the next generation. This may also go some way to explain the chaotic graph.

Another possible reason could be due to the fact that the best player in a given generation is not actually that good. Based on our normalisation, the highest score will always get value 1.0 and publish to the central pool (if it has never been published). Player A might be the best among the superior players in individual learning, but we do not know how good that player is when compared against the other players in the central pool.

The game of Connect-Four has previously been solved using knowledge-based approaches. We have shown that evolving neural networks integrated with individual and social learning has the potential as a technique for evolving a good automated Connect-Four. However, more generations are needed to determine whether the program can become a good player.

Even though the playing performance of our program is not very good, the program has the ability to block the opponent from making two open lines, which is one of the strategies to win the game.

Based on our initial findings, our future research will consider:

Increased Population Size: A larger population may increase the diversity, but will lead to increase computation time.

Evaluate the player in the central pool before copy: To avoid copying a poor strategy from the central pool, we will evaluate whether it is better than the current strategy before copying.

Evaluate the new random strategy: Before using a new random strategy, we will evaluate whether it is better than, or at least equal to the current strategy.

Evaluate how good the player is before publishing to the central pool: Before publishing the player to the central pool, we will apply a new mechanism to evaluate the player to determine that the player is good enough to publish.

In this paper, we have introduced a new technique for learning how to play games. This technique of individual and social learning has previously, successfully, been applied in an economic domain but has not been used as a learning mechanism for game playing.

The contribution of this paper is to introduce this technique and we show that learning does take place. As yet, we cannot beat a commercial program, even at a beginner level but we hope that the work in this paper can serve as a platform to evolve strong game playing agents.

6. REFERENCES

[1] T.S. Anantharaman, M. Campbell and F.H. Hsu, "Singular Extensions: adding selectivity to brute force searching," Artificial Intelligence, vol. 43, no. 1, pp. 99-109, 1989.

[2] M. Campbell, A.J. Hoane Jr. and F-H Hsu, "Deep Blue," Artificial Intelligence, vol. 134, pp. 57-83, 2002.

[3] F.H. Hsu, "Behind Deep Blue: Building the Computer that Defeated the World Chess Champion," Princeton University Press, 2002.

[4] J. Schaeffer, R. Lake and P. Lu, "Chinook the World Man-Machine Checkers Champion," AI Magazine, vol.17, no.1, pp. 21-30, 1996.

[5] J. Schaeffer, "One Jump Ahead - Challenging Human Supremacy in Checkers," Springer Verlag, 1997.

[6] L.V. Allis, H.J. van den Herik, M. Huntjens, "Go-moku Solved by New Search Techniques," Computational Intelligence, 11(4), 1995.

[7] M. Buro, "Methods for the Evaluation of Game Positions Using Examples," Ph.D Thesis of the University of Paderborn, Germany, 1994.

[8] M. Buro, "The Othello Match of The Year: Takeshi Murakami vs. Logistello," NECI Technical Note #012N, NEC Research Institute, New Jersey, USA, 1997.

[9] G. Tesauro, "Temporal Difference Learning and TD-Gammon," Communications of the ACM, 38(3), pp. 58-68, 1995.

[10] G. Tesauro, "Neurogammon Wins Computer Olympiad," Neural Computation, vol. 1, pp. 321-323, 1989.

[11] J.D. Allen, "A Note on the Computer Solution of Connect-Four," in D.N.L. Levy and D.F. Beal (Eds.), Heuristic Programming in Artificial Intelligence 1: the First Computer Olympiad, Ellis Horwood, Chichester, England, pp. 134-135, 1989.

[12] V. Allis, "A Knowledge-based Approach of Connect-Four, The Game is Solved: White Wins," Master Thesis, Department of Mathematics and Computer Science, Vrije Universiteit Amsterdam, The Netherlands, October 1988.

[13] J.W.H.M. Uiterwijk, H.J. van den Herik and L.V. Allis, "A Knowledge-based Approach of Connect-Four, The Game is Over: White to Move Wins," in D.N.L. Levy and D.F. Beal (Eds.), Heuristic Programming in Artificial Intelligence 1: the First Computer Olympiad, Ellis Horwood, Chichester, England, pp. 113-133. 1989.

[14] F. Hsu, "IBM's Deep Blue Chess Grandmaster Chips," IEEE Micro, (March-April): pp. 70-81, 1999.

[15] K. Chellapilla and D.B. Fogel, "Anaconda Defeats Hoyle 6-0: A Case Study Competing and Evolved Checkers Program against Commercially Available Software," in Proceedings of Congress on Evolutionary Computation, La Jolla Marriot Hotel, La Jolla, California, USA, July 16-19 2000, pp. 857-863.

[16] D.B. Fogel, "Blondie24: Playing at the Edge of AI". Morgan Kaufmann, SF. CA, 2002.

[17] N. Vriend, "An Illustration of the Essential Difference Between Individual and Social Learning, and Its Consequences for Computational Analysis," Technical Report, Queen Mary and Westfield College, University of London, 1998.

[18] L. Tesfatsion, "Notes on Learning," Technical Report, Department of Economics, Iowa State University, Ames, Iowa, 30 March 2004. http://www.econ.iastate.edu/tesfatsi/learning.pdf

[19] G. Kendall and Y. Su, "The Co-evolution of Trading Strategies in A Multi-agent Based Simulated Stock Market Through the Integration of Individual Learning and Social Learning," Proceedings of IEEE 2003 Congress on Evolutionary Computation, pp. 2298-2305, 2003.

[20] D.B. Fogel, "Evolutionary Computation: Toward a New Philosophy of Machine Intelligence," 2nd. Edition, IEEE Press, New York, 2000.

[21] Wikipedia, The Free Encyclopedia, http://en.wikipedia.org/wiki/Main_Page

[22] D.A. Mc Allester, "Conspiracy Numbers for Min-Max Search," Artificial Intelligence, 35(3), pp. 287-310, 1988.

[23] J. Schaeffer, "Conspiracy Number," In: D.F. Beal (Ed.), Advances in Computer Chess, Elsevier Science, Amsterdam, vol. 5, pp. 199-218, 1989. Also published in: Artificial Intelligence, 43(1), pp. 67-84, 1990.

[24] M.O. Schneider and J.L.G. Rosa, "Neural Connect 4 – A Connectionist Approach to the Game," VII Brazilian Symposium on Neural Networks (SIBRN 02), 1992.

[25] R.P. Jones and D.J. Thuente, " The Role of Simulation in Developing Game Playing Strategies," Proceedings of the 23rd Annual Symposium on Simulation, Nashville, Tennessee, US, IEEE Press., pp. 89-97, 1990.

[26] K. Chellapilla and D.B. Fogel, "Evolving an Expert Checkers Playing Program Without Using Human Expertise," IEEE Transactions on Evolutionary Computation, 5(4), pp. 422-428, August 2001.

[27] K. Chellapilla and D.B. Fogel, "Co-Evolving Checkers Playing Programs using only Win, Lose or Draw," in Aerosense99, Symposium on Applications and Science of Computational Intelligence II, vol. 3722, pp. 303-312, 1999.

[28] K. Chellapilla and D.B. Fogel, "Evolving Neural Networks to Play Checkers Without Relying on Expert Knowledge," IEEE Transactions on Neural Networks, 10(6), pp. 1382-1391, November 1999.