An Integration of BP-Pool and Social Learning in the Opening of Go

Razali Yaakob
Department of Computer Science
Faculty of Computer Science and IT,
Universiti Putra Malaysia, 43400 UPM Serdang,
Selangor, Malaysia.

Graham Kendall
School of Computer Science,
Jubilee Campus, The University of Nottingham,
Wollaton Road, NG8 1BB Nottingham,
United Kingdom.

Abstract

In this paper, we investigate an integration of a best population pool and social learning, utilising evolutionary neural networks. The experiments are divided into several intervals, and we keep the best player from each interval in the best population pool (BP-pool). Social learning allows poor performing players to learn from those players, which are playing at a higher level. The feed forward neural networks are evolved via evolution strategies and no knowledge is incorporated into the players. The evolved neural network players play against a rule-based player, Gondo, at the beginning of the match. The remainder of the games then copied by another Gondo and they continue the game by playing against themselves. Our results demonstrate that learning is taking place.

1. Introduction

Alan Turing [24], Claude Shannon [22] and Arthur Samuel [17] are among the pioneer researchers who have used games as a domain for artificial intelligence research. Deep Blue [4], Chinook [19, 20, 18], Logistello [3] and TD-Gammon [23] are automated players whose play is superior to better than humans. Deep Blue is an automated Chess player which defeated Gary Kasporov in 1997. It is an example of a program designed to play at World Championship level. Deep Blue uses a database of over 700,000 Grandmaster chess games from previous matches (including the matches against Kasparov in 1996 and 1997) [4] and the best opening moves known.

Chinook is a good example of an automated game that incorporates knowledge, developed by Jonathan Schaeffer′s team at The University of Alberta. It won the world checkers title in 1994 [20]. There are four aspects that contribute to the success of Chinook; these being search, an evaluation function, a database of endgame positions and an opening book.

In contrast to Chinook, Blondie24 does not incorporate any domain knowledge [10]. The player learned to play the game by using a combination of an evolutionary algorithm and artificial neural networks.

The aim of our work is to propose a learning methodology that does not rely on human expertise Our objective is to investigate the integration of a *best population pool* (BP-pool) and social learning in evolutionary neural networks for the game of Go. A feed forward neural network player is played against the rule-based player, called Gondo, and an evolution strategy is used to evolve these networks. The work in here is an extension of [13] and [25].

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 [5, 10] that an automated player can learn to play checkers by competing against other computer players, and in [9], the evolved player played against the nearly perfect player in the game of tic-tac-toe. 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.

978-0-7695-3507-4/08 $25.00 © 2008 IEEE
DOI 10.1109/CSIE.2009.875

2 Background

The techniques we present in this paper are motivated from [12, 13, 25] with some minor modifications. In [12], 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 perform better than a baseline buy-and-hold strategy.

Kendall et al. in [13] uses evolutionary neural networks to learn how to play Go. In [13], the evolved player had played against Gondo only for the opening of the game. After 30 steps, the remainder of the game is copied and played by Gondo against itself. In general, the evolved players have learned how to play the game through their own experience and did not have any chances to copy better strategy from other players. We call this individual learning.

In [25], the evolved neural network players have a chance to create their own strategy through individual learning and copy a better strategy if they are not happy with their current strategy. The integration of individual and social learning has shown that the evolved players are capable of learning how to play Tic-Tac-Toe. The work in here is inspired from [25]. However, in this work we extend our previous investigation by creating another pool, called the best population pool (BP-pool), to store the best player at each interval from each experiment and use them as a population for the next interval. Further discussion about BP-pool are presented below.

Previous authors have investigated evolving neural networks to play Go. Richard et al. evolved neural networks using the SANE (Symbiotic, Adaptive Neuro-Evolution [14]) method [16]. SANE required 20 generations to defeat Wally on a 5x5 board, 50 generations on a 7x7 board, and on a 9x9 board, 260 generations were needed.

Donelly et al. also evolved neural networks, via self-play, where three-layer feed forward networks and a game tree with one ply was used to choose the best move [6]. In terms of performance, the end result was very poor and not even able to beat the weakest Go programs [6].

Other researchers have used neural networks to learn to play Go using machine learning methods such as supervised learning (training from moves from professional games) and are capable of playing at level 13 kyu [7], and reinforcement learning [8, 21]. In [8], the player called *NeuroGo*, was capable of playing at a medium level. In [21], Yen et al. took a different approach to learn to play Go openings with Jimmy 4.0, which has been ranked 4 kyu when they adopted 100 questions from Go books to test the rank of their opening game [26].

Further reviews of the state of the art in computer Go playing can be found in [15] and [2].

2.1 Best Population Pool

The purpose of the integration of BP-pool and social pool is to improve the learning process presented in [13] by having better strategies in the population. In this work, the experiment is divided into T intervals, and each interval has p generations and the experiment is run for M trials. The BP-pool will maintain the best player at the end of each interval (M best players in the pool at the end of 10 trials). The next interval will start with the M best players as the new population and they will also publish in the social pool. Figure 1 shows that the individual player plays against Gondo at every generation. The social activities take place at every 50^{th} generation, where the best player(s) is published into the social pool. At the end of each p interval, the best player among the population is copied into BP-pool. Figure 1 also shows that after 10 trials, all 10 players in the BP-pool are published into the social pool and also used as a population for the new interval.

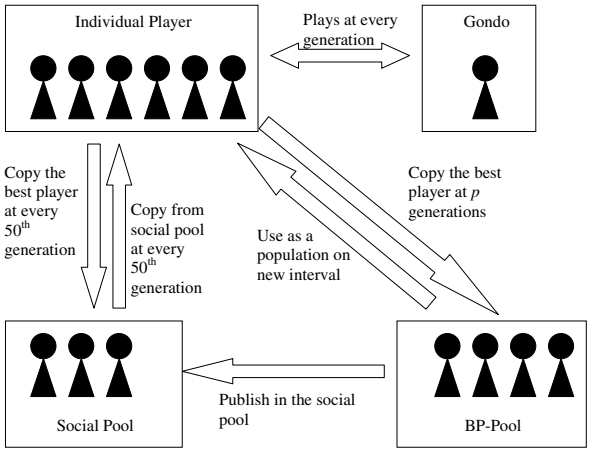

Figure 1. Integration of individual player, social pool and BP-pool

The same structure of spatial neural networks in [13] is used in this experiment and we use the same techniques as described in [13] (see the section *Spatial Neural Network without Minimax*), to search for the best move for the neural network players. The feed-forward neural networks have nine input nodes, nine hidden nodes and an output node. In this work, the evolved neural networks will search for the best move using 1-ply search using the following algorithm:

1. Read the current state of the game.
   (a) Put a marker 'X' on any available square.
   (b) Pass the new current state to the neural network.
   (c) Store the real value from the output.

(d) Remove the marker placed in step 1a from the square.

(e) Repeat steps 1a to 1d until all available squares have been evaluated by the neural network and has an associated evaluation.

2. Select the highest output as the best move.

3. Experimental Setup and Result

The evolved neural network players will play against Gondo, for the opening phase only and the remainder of the game will be played against Gondo. The game is divided into three phases (opening, middle and end game). We set the total opening moves to 56 steps (28 by each player). This figure is arbitrarily chosen, as there is no fixed values for the number of moves in the various phases of the game. It depends on the nature of the game during the match. In this work, Japanese scoring is used to calculate the score for each player and used as the fitness to rank the players (please refer [11] for further review).

The experiment is divided into 10 intervals and each interval has 10 trials, where each trial is run for 100 generations (1,000 generations in total). In [13], the evolved players were played against Gondo as black and white. Each of them played only once before selection for mutation. This is because both of them are deterministic. In this new experiment, we try to give each player the opportunity to play the game several times before the selection is made for mutation. The algorithm for this work is as follows:

1. Initialise a random population of neural networks, P_i, where P is a neural network player, i = 1, ..., N, and N is a total number of neural networks in the population. Each strategy has its associated self-adaptive parameter, s_i, i = 1, ..., N, and were initialised randomly.

2. Generate an offspring for each parent, where for each P_i, i = 1, ..., N, an offspring, P_i' is created by using Eq. (1a) and (1b) [1]. The total neural networks now is N+N.

$$s_j' = s_j.exp(\tau.N_j(0,1)) \quad (1a)$$

$$w_j' = w_j + s_j'.N_j(0,1), \quad (1b)$$

3. The evolved neural network player makes the first move.

4. Gondo determines the four possible second moves (the reason why a value of four is used is explained below).

5. Gondo will choose one of the possible second moves that has not been played. The subsequent move from Gondo will follow its own rules.

6. At the end of each match, the neural network players receive a score and this score is used as a fitness measure for selection.

7. After 56 turns (28 turns by each player), the current position for the neural network player is copied to another version of Gondo and it plays against itself until the end of the game.

8. After finishing the game, the same evolving player will place its initial stone on the same square for a new game and repeat steps 3 to 7 until all possible four second moves have been played.

9. Repeat steps 3 to 8 for 4 times.

10. If (number of generations is divisible by 50)

    - Enter the social learning activities, else

11. At the end of each generation, the N strategies (neural networks) with the highest scores are selected as parents and retained for the next generation. These parents are then mutated to create another N offspring using Eq. 1a and 1b.

12. If (number of generations is divisible by p),

    - Keep the best individual player in the BP-pool

13. Repeat steps 3 to 12 until complete one interval (100 generations).

14. Repeat steps 3 to 13 until the stopping criterion is reached (i.e. generations = 1,000).

The value of four possible second moves is chosen because the Go board has four edges. In Go, one of the basic techniques to conquer the board is by starting at the corner of the board. Actually, we can have more than four moves, however, due to computational time, we limited the value to four. In fact, the Gondo player always starts somewhere in the corner at the opening of the game.

This experiment was run on Intel(R) Pentium(R) 4, CPU 3.20GHz,and 2.00 GB of RAM, and it takes about a month to complete. In each trial, we kept the highest score at each generation, and Figure 2 shows the mean best scores of 10 players from each generation. There are two regression lines in Figure 2, where the first regression line is for the first interval (marked as First Interval) - from generation 1 until 100. The second regression line is from generation 101 until 1,000 (marked as After First Interval). We plot these two types of regression lines to analyse the result before and after integration of BP-pool and social learning.

[1] taken from [5, 10]

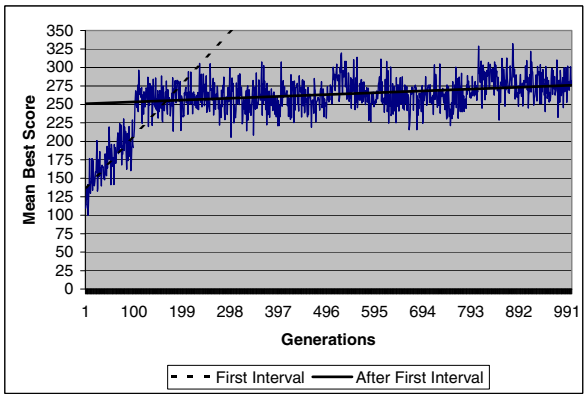

Figure 2. An average score of best player at each generation

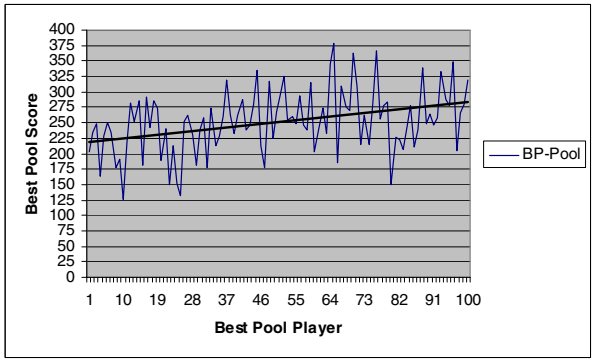

Figure 3. Best population score in the BP-pool for each interval

The regression line can be used to check whether learning has occurred in this experiment by testing whether the slope of the regression line is significantly different from zero. We run a t-test on the highest score of each generation, and we found that the slope of the regression is statistically different.

Based on Figure 2, the first interval (1 to 100 generations), appears to exhibit an upward trend. At this point, the players start the game with a random strategy. After the first interval, the program starts with a better strategy from the BP-Pool and the result looks much better. The progress looks static between the second and forth interval (between generation 100 and 400). The evolution appears to be getting better after the eighth interval (at generation 800).

The other way to check whether learning has occurred in this experiment is through the BP-pool, where logically, the score in the BP-pool should increase through the experiment. Figure 3 shows the score in the best pool from the first interval until the last interval (10 intervals in total). The first 10 best players are from 10 trials in the first interval, the next 10 players (indexed 11 to 20) are the 10 best players in the second interval and so on. The straight line on the graph is the regression line for the BP-pool scores and the equation as in Eq. 2. Based on the t-test, the hypothesis that the slope of the regression line for the pool score in the BP-pool is statistically different from zero.

$$y = 218.87 + 0.648x \quad (2)$$

Both trend lines (in Figure 2 and 3) show that a process of learning is taking place. Even though the lines look chaotic, the regression lines show that they have significantly increased, therefore, learning has occurred in this experiment.

4 Conclusions

Learning has occurred through the integration of BP-Pool and social learning, even though no knowledge has been incorporated and there is no look ahead. Even though the graphs looks chaotic, the slope is significantly different from zero at the 95% confidence level.

Go is very complex game and we might need a huge database if we want to integrate knowledge into the program. That is why we believe that an evolutionary approach is suitable in automated Go programs. However, the learning structure has to be well-designed. The integration of a BP-pool and social pool has the potential to be used in learning to play Go. However, more generations are needed to give enough time for the player to learn. We also need to look at the individual phase in the learning process. In [25], the program went through individual learning 100 times. However, we are not sure how long players actually need to create their own strategy in individual learning before social activities should take place.

In this work we only used four possible second moves. However, after the first move, there are 168 possible second moves, and it would take too long to consider all these moves. Maybe in the future we can start with high handicap which reduces as play improves. In this case, the number of possible second moves would be reduced, thus, reducing the computational times.

In conclusion, the integration of a BP-pool and social learning has the potential to be used as a learning technique in complex games. However, computational time is a limiting factor for games such as Go. We hope, in the future, that we are able to evolve an automated Go program with a larger population, more generations and appropriate individual learning duration. The game-tree complexity of Checkers is 10^{21} and Go is 10^{169} [1]. Therefore, if Fogel needed 8 months for Blondie24 [10], Go, at the present

time, appears to be intractable.

References

[1] L.V. Allis. *Searching For Solutions in Games and Artificial Intelligence*. Ph.d. thesis, University of Limburg, Maastricht, July 1994.

[2] B. Bouzy and T. Cazenave. Computer Go: An AI oriented survey. *Artificial Intelligence*, 132(1):39–103, 2001.

[3] M. Buro. The othello match of the year: Takeshi murakami vs. logistello. *International Chess Computer Association Journal*, 20(3):189–193, 1997.

[4] M. Campbell, A.J. Hoane, and F.-H. Hsu. Deep blue. *Artificial Intelligence*, 134:57–83, 2002.

[5] K. Chellapilla and D.B. Fogel. Anaconda defeats hoyle 6-0: A case study competing an evolved checkers program against commercially available software. In *Proceedings of Congress on Evolutionary Computation*, pages 857–863, La Jolla Marriot Hotel, La Jolla, California, USA, July 2000.

[6] P. Donelly, P. Corr, and D. Crookes. Evolving go playing strategy in neural networks. In *AISB Workshop on Evolutionary Computing*, Leeds, England, 1994.

[7] H.D. Enderton. The golem go program. Technical report cmu-cs-92-101, Carnegie Mellon University, 1991.

[8] M. Enzenberger. The integration of a priori knowledge into a go playing neural network. http:/www.markus-enzenberger.de/neurogo.ps.gz, 1996.

[9] D.B. Fogel. Using evolutionary programming to create neural networks that are capable of playing tic-tac-toe. In *Proceedings of IEEE International Conference on Neural Networks*, pages 875–880, Piscataway, New Jersey, 1993. IEEE Press.

[10] D.B. Fogel. *Blondie24: Playing at the Edge of AI*. Morgan Kaufmann, San Florida, California, 2002.

[11] I. Ishigure. *In the Beginning: The Opening in the Game of Go*. The Ishi Press, 1973.

[12] G. Kendall and Y. Su. The co-evolution of trading strategies in a multi-agent based simulated stock market through the integration of individual and social learning. In *Proceedings of IEEE 2003 Congress on Evolutionary Computation*, pages 2298–2305, 2003.

[13] G. Kendall, R. Yaakob, and P. Hingston. An investigation of an evolutionary approach to the opening of go. In *proceedings of Congress on Evolutionary Computation*, pages 2052–2059, Portland, Oregon, 20-23 June 2004.

[14] D. Moriarty and R. Mïkulainen. Forming neural networks through efficient and adaptive co-evolution. *Evolutionary Computation*, 5(4):373–399, 1997.

[15] M. Müller. Computer go. *Artificial Intelligence*, 134:145–179, 2002.

[16] N. Richards, D.E. Moriarty, and R. Mïkulainen. Evolving neural networks to play go. *Applied Intelligence*, 8:85–96, 1998.

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

[18] J. Schaeffer. *The Games Computers (and People) Play*, pages 189–266. Academic Press, 2000.

[19] J. Schaeffer, J. Culberson, N. Treloar, B. Knight, P. Lu, and D. Szafron. A world championship caliber checkers program. *Artificial Intelligence*, 53(2-3):273–290, 1992.

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

[21] N. Schraudolph, P. Dayan, and T. Sejnowski. Temporal difference learning pf position evaluation in the game of go. *Advances in Neural Information Processing 6*, 1994.

[22] C.E. Shannon. Programming a computer for playing chess. Philosophical Magazine, March 1950.

[23] G. Tesauro. Temporal difference learning and TD-gammon. *Communications of the ACM*, 38(3):58–68, 1995.

[24] A.M. Turing. Digital computers applied to games. pages 286–310. Pittman, London, UK, 1953.

[25] R. Yaakob and G. Kendall. An investigation of an integration of individual and social learning in an evolutionary approach to the game of tic-tac-toe. In *Proceeding of the Second International Conference on Informatics*, pages 72–77, 2007.

[26] S-J. Yen, W-J. Chen, and S-C. Hsu. Design and implementation of a heuristic beginning game system for computer go. In Association for Intelligent Machinery, editor, *Joint Conference on Information Sciences (JCIS)*, pages 381–384, 1998.