Building Fitness Value Improvement using Evolutionary Process through Genetic Machine Learning Approach

Dharm Singh
College of Technology and Engineering, MPUAT, Udaipur, India

Chirag S Thaker
Faculty of Engineering
Suresh Gyan Vihar University
Jaipur, India

Sanjay M Shah
Faculty of Engineering
Suresh Gyan Vihar University
Jaipur, India

ABSTRACT

Since decades developing programs for board games has been part of AI research and this field has attracted computer developers and researchers world-wide. Board games have a novel feature of simple, precise, easily formalized rules which makes them perfect launch vehicle to make computer game playing in a suitable development environment. The paper focuses on the two players, full knowledge, alternate move, deterministic, zero-sum game of Checkers. Genetic algorithmic approach is been applied in evolving computer player for the game of Checkers.

The notion of this paper is to incorporate systematic game playing approach by analyzing game of checkers. Expert game players reveal three major playing strategies to make game winning moves. The game moves are divided into three stages opening game, middle stage and endgame. An evolutionary program plays game of checkers with an intention to build resilient middle stage and a set of predefined rules are incorporated to make calculated moves in an endgame.

General Terms

Evolutionary checkers ,game programming, et. al.

Keywords

Open four, split three , game learning Board Game, Genetic Algorithm, Checkers, Game Configuration, Fitness Function.

1. INTRODUCTION

Evolution by natural selection is first and foremost nature's truth very well explained with scientific methodology by Darwinian principles. The process of evolution is a very major and effective source for ideas [1]. Though it is by no guarantee that an idea that works in the natural world will work in our artificial programming environment, but it can be seen as an sign that it might work. Researchers are mindful of evolutionary theory, particularly when it is connect to the genetic-centered view of evolution [2].

As the computer age has arrived, developers started feeling the need to create an intelligent game program which can be made capable of defeating their counterpart human experts. Many different approaches have been used for different games including neural networks for backgammon, special-purpose hardware called Deep Blue for chess and the application of expert knowledge with relatively small computational power for Othello [3][4]. Many of the mentioned techniques exploited expert knowledge domain as much as possible by exploring various dimensions like proper learning algorithm for training the evaluation function, its relevance factors for the evaluation, the weights analysis and mapping to include betterment of the evaluation factors, opening knowledge, middle game and an endgame strategies of the board game. Building such knowledge base requires the expert help and advice of game professionals simultaneously it also demands good amount of computational power for processing the knowledge extracted and method to find the best game-suitable approach [5][6].

Age old traditional methods divide board games into three winning –move making strategies like opening, middle and endgame stages. Each stage requires different philosophies. For example, in an opening game making a very appropriate choice is very difficult to make where an opening book from experts proves to be very constructive [7]. Likewise in the middle stage, a visionary mind set look for a specific characteristics game tree with a reasonably limited depth which constructs the platform to have high-end mobility. This game phase also believes in evaluating functional features with their significance of each board square and disc of that square by applying to estimate the relevance of each move. End game always aims at maximizing the number of pieces or minimizing mobility of opponent's or moves which gives deterministic calculation of the final move(s) [8].

2. INTRODUCTION TO CHECKERS

Checkers (also known as "draughts") is played on an eight-by-eight board with squares of alternating colors. There are numerous variants (more than 150 worldwide) of the game played around the world. There are two players who take sides denoted by "red" or "white" (or "black" and "white"). Each side has 12 pieces which are also called checkers that begin in the 12 alternating squares of the same color that are closest to that player's side, with the rightmost square on the closest row to the player remaining open (Figure 1) [9].

It is noteworthy that Checkers are allowed to move forward diagonally one square at a time only. In the case that a jump condition is satisfied, one can jump diagonally over an opposing checker and the opposing checker is removed. Jumps are mandatory. When a checker advances to the last row of the board, it becomes a king which is capable of moving diagonally one square at a time in any direction. The game reached an end when a player has no more available moves (that player without moves is the loser) and the game can also end when one side offers a draw and the other accepts. Checkers has a relatively smaller search space than chess. It is such small enough that one can consider solving the problem.

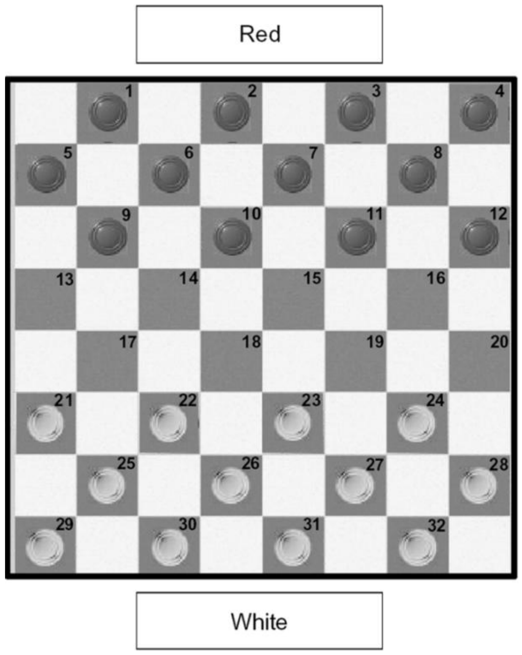

Fig. 1 Checkers Board – Initial Configuration

Checkers move one square at a time forward diagonally or, when possible, may jump over an opposing checker into an empty square. This jump moves are forced, although if more than one possible jump is available, the player may choose which jump to execute. When any checker advances to the last row of the board it advances to become a king and now may move forward or backward diagonally. The game ends when any one side cannot make a legal move, which is most commonly brought about by removing that player's final piece. The player who cannot move loses and the other player wins. Alternatively, a draw may be declared upon mutual agreement of the players, or in tournament play at the discretion of a third party under certain circumstances. In some tournament play, a time restriction (60 minutes for the first 30 moves) is also introduced which if violated results in a loss. This paper is restricted to the 8X8 variant, but many of the ideas presented here also apply to the 10 X 10 game[10][11].

For the game of checkers, the approach is to search a game position state to find an optimal move at each play. This simple viewpoint faces many challenges in the opening, middle, and endgame stages. Sometimes it is observed that a computer checkers program fails to defeat a human player because it makes a mistake which is not common for human players because of cognitive side of their brain. The root cause of fault is found that tree-search made by the computer program was not sufficient and deeper search on other hand consumes a lot of move finding time. Generally board game related opening knowledge and endgame databases are not involved in the evolutionary process because evolutionary approach aims to explore all dimension centric possibility of pure evolution. Though this can be very time consuming and as a result of that, it might take a very long evolution time to create a world-level champion program without a predefined knowledge base [12].

An another methodology is to incorporate a priori knowledge, which is as an expert knowledge, metaheuristics, human preferences and most importantly, domain knowledge revealed during evolutionary search. Thus these evolutionary algorithms (EAs) have gained increasing interest in recent years. It is found by a common belief that the combination of diverse well playing strategies can defeat the best one because they can complement each other for higher performance. This is also very instrumental component of evolutionary algorithms. This paper proposes a systematic method to have into evolutionary checkers framework at all the opening, middle, and endgame stages [13].

2.1 Game Complexity

The game of checkers has roughly 500 billion- billion possible positions (5×10^{20}). The task is very daunting to solve the game, determining the finishing result in a game with no error made by either of the player. Since last three decades, almost incessantly, dozens of computers have been working on solving Game of Checkers, applying state-of-the-art soft computing based techniques to improve the learning process [14].

Game of Checkers represents the most computationally challenging game to be solved to date. Evolutionary Learning challenges in Game of Checkers are:

(1) The space to be searched is huge. It is estimated that there are up to $5X10^{20}$ possible positions that can be searched. So any search algorithm based method which is based on exhaustive search for the problem space is infeasible.

(2) The search space volume is not smooth and straight forward. An evaluation function's parameters which is feature construction based are highly inter-dependent. In some cases increasing the values of optimization parameters will result in a worse performance, but many a times the controlled set of evolutionary parameter is also increases performance, then an improved overall performance would be obtained.

(3) The problem is not well understood by researchers. Even though all top performing programs parameters are hand tuned by their program designers, finding the best value for each parameter is mostly based on operational genetic alternatives [15][16].

Chinook, one of the best checkers program has not completely solved the game but it is playing at a steady level that makes it almost unbeatable. The average number of moves to consider in a checkers at every move making position (called the branching factor) is less than that for chess. A typical checkers position without any captures has eight legal moves (for chess it may be 35–40). As a result, checkers programs can search deeper than their chess counterparts. Checkers provides an easier domain to work with, and provides the same basic research opportunities as does chess or Go [17].

2.2 Traditional Game Programming

A board game can usually be divided into three general phases: the opening, the middle game and the endgame. Inflowing thousands of game positions in published books into the program is a way of creating an opening book. The checkers program Colossus has a book of over 34 000 positions that were entered manually. A problem with this approach is that the program will follow published play, which is usually familiar to the humans. But without using an opening book, some programs find many interesting and promising opening moves that standoff a human quickly. It has a danger of producing incurable mistakes and enter a losing board configuration quickly as a deeper search would have been necessary to avoid the mistake. Thus humans have an advantage over computers in the opening stage because it is difficult to quantify the relevance of the board configuration at an early stage [18].

To be more competitive, an opening book can be very supportive but a huge opening book can make the program stubborn and without originality.

One of the important parts of game program development is to design the evaluation function for the middle stage of the game which is a linear combination of features based on human knowledge-expertise like such as the number of kings, the number of checkers, the piece differential between two players, and pattern-based features. Based on the functional features of the specific board game function weight determining components can be devised. Then evolutionary cycles tune the weights of the evaluation function through algorithms [19].

2.2.1 Evolution and Games

Checkers is the board game for which evolutionary computation has been used to evolve strategies. Fogel *et al.* have explored the potential for a coevolutionary process to learn how to play checkers without relying on the usual inclusion of human expertise in the form of features that are believed to be important to playing well [20].

3. INCORPORATING KNOWLEDGE INTO EVOLUTIONARY CHECKERS

The game of Checkers has been classified into three stages: opening, middle and endgame stages. In the opening stage, about 80-100 previously evaluated and summarized openings are used to determine the initial moves. In the middle stage, a move search game tree is used to search for an optimal move and an evolutionary genetic algorithm is employed to evaluate leaf nodes of the tree. Genetic algorithms are very suitable to optimize the fitness value based leaf nodes. As the fitness landscape of an evolutionary game evaluation function is greatly dynamic. The performance of evolutionary GA for creating a checkers evaluation function has been demonstrated by many researchers. The result of the game is measured in win/loss/draw after the number of remaining pieces is smaller than a predefined number (usually from 2 to 10) [21].

1) Evolutionary Concept of a Game Tree: To find the next move of a player, a game tree is constructed with a limited depth. Each node in a game tree embodies the configuration of the board at some stage of the game. The quality factor of the terminal nodes is measured with the evaluator. These evaluated values of the terminal nodes are been proliferated upward using min/max search algorithms. Here the algorithm working is very simple ,the max operation chooses the max value of all children nodes and the min operation chooses the min value. The current configuration of the board is represented as a root node and the arc represents a move. At an odd number level, the max operation is used and *vice versa*.

*2) Evaluation of a Board Configuration: A*n evaluation function is the linear sum of the values of relevant board game features selected by experts. The input of the evaluation function is the configuration of the board and the output of the function is a significance parameter of quality. To Design evaluation function manually requires board game expertise and tiresome tuning process with feedback attitude. Board game features tend to be modeled into using some machine learning techniques. Though it is not easy and can have its own share of problems for learning and tuning the evaluation function. The problems are in the form of architectural determination and configuration transformation of features into numerical form.

This way the evolved evolutionary program exhibits a flexibility that can be achieved with Checkers playing program in evolutionary approaches. This is heavily dependent on all of their "intellect" being put forwarded in a programmed form. The evolutionary algorithm is also made capable of adapting its game play to meet more demanding challenges from better-quality opponents based on the stages of the game as the game follows different principles in different stages of the game play. It can conceive new and untraditional procedures to evolutionary adapt to novel situations [22].

Evolutionary algorithms discover a large number of points simultaneously in a search space. This phenomena avoids the chances of poor move stagnation. It results in a faster and fertile search. Here coding a given Checkers problem into an Evolutionary Genetic algorithm computation context, these canny algorithms becomes capable of evolving winning solutions to such board games.

4. PROGRAM IMPLEMENTATION

In an evolutionary genetic algorithmic style, each checkerboard can be represented by a genetic vector with length of 32 with each vector representing one side to an available position on the board. As the evaluation function is in a linear form here vector components are elements having the values −1, 0, +1, where 0 corresponded to an empty square, 1 was the value of a regular checker, and *-1* was the number assigned for an opponent's checker piece king. Here the sign of the value for component was indicative for the piece belongingness, either to the player (positive) or his/her opponent (negative) [23].

Any move from a player's is determined by evaluated fitness function parameters of the board game positions multiplied by their related vector weight values. Thus evaluation function is formulated for the evolutionary landscape parameter values. The first set of vector values is randomly set using experienced human expertise. These random set of values are designed to indicate the spatial characteristics of the game board. Then after the remaining subsequent values are computed that are based on evolutionary computation process. It primarily evolves the genetic weight parameters. These weight values and the functional features' values are in dot product formation gives final fitness values which forms the basis for the move selection. These values become base for search algorithm like min-max and the better value is chosen to make the next move. Above procedure is repeated for a reasonable ply depth of three. Here it is noteworthy that more depth selection makes the algorithm slower as exploration of all possibilities for all depth values take a lot of time. It hampers the evolutionary benefits of better move selection in reasonable time [24].

In an initial board set up the complexity enhancement is because of board weights and their respective holdings which are worth of the board squares importance (based on its substantial features) and its relative importance in game tree. If these findings are favorable to the player's interest then these pieces take positive values. If very initially the potential move itself is set closer to the final output then it takes value 1.0, which denotes that corresponding board square is better. But if the move takes potential output to an opponent's favor the square is set to –1.0, which means the board square situation is worsening which may result in combined loss over a long run of moves. All budding winning positions that were wins for the player (e.g., no remaining opposing pieces) were assigned a value of exactly 1.0, and similarly all losing positions were assigned a value of –1.0 [25][26].

In a genetic evolutionary process each "parent" is generated as an off springs by changing all of the associated weights and related square values. Here all parents and their offspring compete for survival by playing games moves of checkers and receiving points (weights) as the results of their play. Each player scored –1, 0, or +1 points for a loss, draw or win, respectively. These figures are stand-alone and contain no optimality parameter into it. A draw is declared in case if a game lasts for 100 moves. In total, each game has an average of 40-60 moves and such 50 games are been played. In each generation within a game, each checker piece is observed for its participation and corresponding fitness values it possesses. After all move possibilities are explored, all the participating checkers squares that received the greatest total fitness weight points across all fitness value analysis and computation are reserved as parents for the next generation and the evolutionary genetic process gets repeated for considerable number of time to accomplish better fitness results.

Each game tree is explored by applying a min-max alpha-beta search algorithm with a look ahead over a selected number of moves. The ply depth of the search, *d*, was set at six (i.e., three depth move travelling for each side) to allow for reasonable execution times (30 generations). The depth ply is extended in units of three any time a forced jump move was encountered because in these situations no real decision is available.

The every move making place the best move is selected on values collected over running min-max repeatedly for all possibility in a given depth. These moves positive and negative depth are dependent on the side of the player. For the player the weight notion is positive and for opponent the weight sign is set to negative.

5. CHECKERS COLLECTED RESULTS

Total 50 Checkers games against human players are been played with evolutionary algorithmic to determine best moves. No opponent was told that they were playing against a computer program, Games were played mainly using a ply of *d* = 6 to ensure reasonably mature depth exploration and thus limiting only 50 games as the used to consume a reasonably good amount of time.

With a population size was set to 70 and genetic operator parameters are selected as follows:

- Selection rate = 0.20
- Crossover rate = 0.80
- mutation rate = 0.01 (occasional)
- Population size= 100

Above mentioned parameters reveal that evolutionary genetic computation is employed effectively for the game of Checkers program and its board squares weight parameters underwent evolutionary progress through their evaluation function. Their evolved program managed to compete and came up with strong results because of the mature search depth of ply =6.

Apart from the above mentioned genetic operators described above, a standard implementation of GA with comparative selection and single point crossover was used. And results are collected.

The time line chart in minutes for the initial twenty five generations for the ply depth of six is shown in fig. 3. This figure is clearly indicative that initially up to game of five it takes less time but then after the time consumption in minutes increases. For the game no of 6, 15 and 25 the time taken in playing the game is maximum. For the game number between 9 to 15, the time used to play a full game sees a steep rise which sows the maturity the game attains in playing with a ply depth of six. The slump in game no. 2 ,9 and 21 shows that opponent player has made very solid moves which does not allow the computer program player to lengthen the game and take the result in its favor.

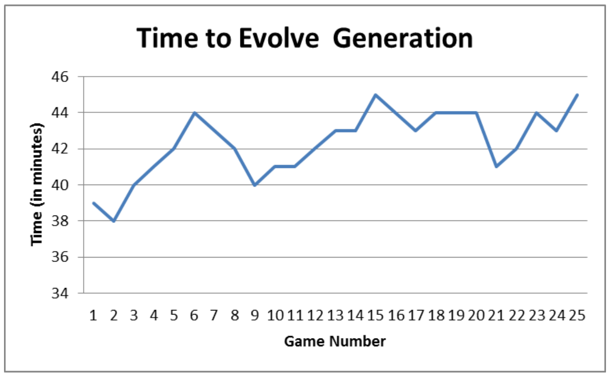

Fig. 2 Time Graph to play one Game of Checkers

Fig 4 shows the collected set of min. and max fitness value (shown by blue and red bars respectively) in each generation s for final set of twenty five generations. The max .board square fitness values shows a steady and comparatively sluggish rise the game of Checkers. But the same values collected for min. fitness shows a very strong and consistent rise which is very clear that the game play is getting matured over a span of twenty five generations and gives positive rise to the board square fitness in a given set of generations.

Here GAs are acting as a clear Evolutionary Algorithms were used for evolving evaluation function parameters through a mature move selection. This game playing approach facilitates the use of GA for efficiently evolving an evaluation function's parameters. As our results will demonstrate, this method is very fast, and the evolved program is on par with the world's strongest checkers programs.

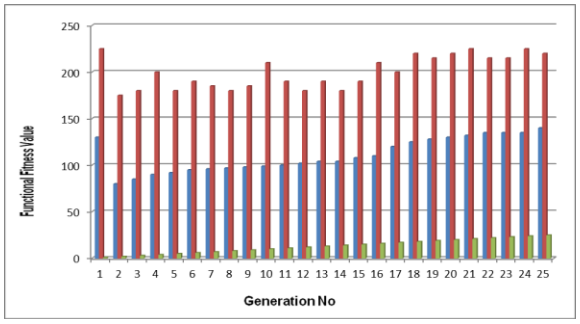

Fig 3 Fitness Value Analysis for 25 Generations

6. CONCLUSION

This paper has shown two results graphs, one on time required to play one full checkers game and second graph shows Max and Min. fitness values collected over a span of twenty five generations. Both these results are clearly indicative of the positive learning impact of genetic parameters in evolving the game play and positive increment of fitness values respectively. These results clearly envisage and justify the effect of evolutionary genetic approach on all the game of checkers.

The paper has confined it's research reach to 50 generations only as the search ply depth of six. For which the results are collected. This search depth is reasonably good and evolving games are taking more time to finish which is due to better move selection evolution.

It is found that for many board game domains, 50-100 generations are good enough to find near-optimal good solutions. This is better than an exhaustive search based on some simple game heuristics. The collected results show very clear fitness function improvement in Max. and Min. values in each of the generations. The paper concludes the right and mature choice of genetic algorithm as evolutionary optimization technique and selected genetic parameters for the specified board game.

7. REFERENCES

[1]. M. Campbell, A. J. Hoane Jr., and F.-H. Hsu, "Deep blue," Artif. Intell.vol. 134, no. 1–2, pp. 57–83, 2002.

[2]. J. Schaeffer, One Jump Ahead: Challenging Human Supremacy in Checkers. New York: Springer-Verlag, 1997.

[3]. M. Buro, "The othello match of the year: Takeshi Murakami vs. Logistello,"Int. Comput. Game Assoc. J., vol. 20, no. 3, pp. 189–193, 1997.

[4]. D. B. Fogel, Blondie24: Playing at the Edge of AI. San Mateo, CA:Morgan Kaufmann, 2001.

[5]. S. Y. Chong, D. C. Ku, H. S. Lim, M. K. Tan, and J. D. White, "Evolved neural networks learning Othello strategies," in Proc. Congr. Evol.Comput., vol. 3, 2003, pp. 2222–2229.

[6]. R. Fortman, Basic Checkers (http://home.clara.net/davey/ basicche.html, 2007).

[7]. Seo, Y.G., Cho, S.B., Yao, X.: Exploiting coalition in co-evolutionary learning. In:Proceedings of the 2000 Congress on Evolutionary Computation. Volume 2., IEEE Press (2000) 1268-1275.

[8]. Chellapilla, K., Fogel, D.: Evolving a neural network to play checkers without human expertise. In Baba, N., Jain, L., eds.: Computational Intelligence in Games. Volume 62. Springer Verlag, Berlin (2001) 39-56.

[9]. J. Schaeffer et al., "Solving Checkers" (www.ijcai.org/ papers/ 0515.pdf, 2005).

[10]. Fogel, D., Hays, T., Hahn, S., Quon, J.: A self-learning evolutionary chess program. Proceedings of the IEEE 92 (2004) 1947-1954.

[11]. Kusiak, M., Waledzik, K., Mandziuk, J.: Evolution of heuristics for give-away checkers. In Duch, W., et al., eds.: Arti¯cial Neural Networks: Formal Models and Their Applications - Proc. ICANN 2005, Part 2, Warszawa, Poland. Volume 3697 of LNCS Springer (2005) 981-987.

[12]. Kendall, G., Whitwell, G.: An evolutionary approach for the tuning of a chess evaluation function using population dynamics. In: Proceedings of the 2001 Congress on

Evolutionary Computation CEC2001, IEEE Press (2001) 995-1002.

[13]. Mandziuk, J., Osman, D.: Temporal deference approach to playing give-away checkers. In Rutkowski, L., et al., eds.: 7th Int. Conf. on Art. Intell. and Soft Comp.(ICAISC 2004), Zakopane, Poland. Volume 3070 of LNAI., Springer (2004) 909-914.

[14]. Osman, D., Mandziuk, J.: Comparison of tdleaf(λ) and td(λ) learning in game playing domain. 11th Int. Conf. on Neural Inf. Proc. (ICONIP 2004), Calcutta, India. Volume 3316 of LNCS., Springer (2004) 549-554.

[15]. Osman, D., Mandziuk, J.: TD-GAC: Machine Learning experiment with give-away checkers. In Draminski, M., Grzegorzewski, P., Trojanowski, K., Zadro_zny, S., eds.: Issues in Intelligent Systems. Models and Techniques. Exit (2005) 131-145.

[16]. Pollack, J.B., Blair, A.D., Land, M.: Coevolution of a backgammon player. In Langton, C.G., Shimokara, K., eds.: Proceedings of the Fifth Artificial Life Conference, MIT Press (1997) 92-98.

[17]. T. M. Mitchell. Generalization as search. Artificial Intelligence, 18:203 - 226, 1982.

[18]. Paul Marcos Siqueira Bueno and Mario Jino. Identification of potentially infeasible program paths by monitoring the search for test data. In Proceedings of the 15th IEEE International Conference on Automated Software Engineering (ASE), Grenoble, France, September 2000.

[19]. S. Edelkamp, A. L. Lafuente, and S. Leue. Directed explicit model checking with hsf-spin. In Proceedings of the 2001 SPIN Workshop, volume 2057 of Lecture Notes in Computer Science, pages 57–79. Springer-Verlag, 2001.

[20]. M.S. Campbell and T.A. Marsland. A comparison of minimax tree search algorithms. Artificial Intelligence, 20(4):347–367, 1983.

[21]. O. David-Tabibi, A. Felner, and N.S. Netanyahu.Blockage detection in pawn endings. Computers and Games CG 2004, eds. H.J. van den Herik, Y. Bjornsson, and N.S. Netanyahu, pages 187–201. Springer-Verlag, 2006.

[22]. R. Gross, K. Albrecht, W. Kantschik, and W.Banzhaf. Evolving chess playing programs. In Proceedings of the Genetic and Evolutionary Computation Conference, pages 740–747. Morgan Kaufmann Publishers, New York, 2002.

[23]. C.-T. Sun and M.-D. Wu, “Multi-stage genetic algorithm learning in game playing,” NAFIPS/IFIS/NASA, pp. 223–227, 1994.

[24]. G. Kendall and C. Smith, “The evolution of blackjack strategies,” in Proc. Congr. Evol. Comput., vol. 4, 2003, pp. 2474–2481.\

[25]. Chirag S. Thaker , Dharm Singh, Sanjay M. Shah, Performance Improvement in Game Playing Using Evolutionary Computation by Large Search Space Exploration 2011 ETNCC, pp 148-152, IEEE Catalog Number CFP1196N-CDR , ISBN 978-1-4577-0238-9 and IEEE Catalog Number CFP1196N-ART , ISBN 978-1-4577-0240-2.

[26]. N. Richards, D. Moriarty, and R. Miikkulainen, “Evolving neural networks to play go,” Appl. Intell., vol. 8, pp. 85–96, 1998.