ISSN No. 0976-5697

Volume 4, No. 4, March-April 2013

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

# Artificial Neural Network Based Adaptive Chess Playing Machine

Diwas Sharma*
Computer Science and Engineering Department
Sikkim Manipal Institute of Technology
Sikkim, India
diwas.sh@gmail.com

Udit Kr. Chakraborty
Computer Science and Engineering Department
Sikkim Manipal Institute of Technology
Sikkim, India
udit.kc@gmail.com

***Abstract:*** Various research projects have attempted to build a program that learns to play chess game, given little or no prior knowledge beyond the rule of the game. A typical chess playing engine exhaustively explores the moving possibilities from a chessboard configuration to choose what the next best move to play is. The brute- force method used by the Deep Blue chess machine has made huge impact in the field of artificial intelligence, but is immensely resource hungry. This paper presents a very simple and efficient approach to develop an intelligent chess engine which will hint at the best possible move using the evolutionary and adaptive computing technique on learning from the human grandmasters.

***Keywords:*** Artificial Neural Network; Championship game; En passant; legal move; Octavius

## I. INTRODUCTION

Man has long been taking pleasure out of challenging his intellect. Various methods, tests, procedures have been developed over the ages for the purpose. Board games remnants have been excavated even from the ruins of Indus Valley Civilization at Harappa and Mohenjodaro, which prove that even as early as 2600 B.C man tested is mind. These games, mostly played between two individuals tested the prowess of tactical planning of the players. The most popular of such games, currently known worldwide in its present form, as Chess, is by far the most challenging, intriguing and difficult game to play. The level of difficulty or complexity lies in simply the number of possible moves for each state. To this if tactical planning and deceit are added, we get brain teasers.

Chess, known as the game of "perfect information", because both players are aware of the entire state of the game at all time, just by looking at the board, has since been considered as standard of testing intelligence. Artificially intelligent programs have been developed by such computing majors as IBM, specifically for the purpose of playing chess, and after Deep Blue, (Mid 1990's) [1], defeated the world chess champion Garry Kasparov, machines have been consistently beating man in the rapid version of the game. The kind of victory of a machine over a human grand master has led to make chess one of the most used games to promote artificial intelligence and Computer Science. Nevertheless, chess engines are not able to strategically plan moves or to explain why they compute such a combination of moves. In classical versions however (where time limit is higher) man still reigns. This limitation in the power of the machine exists due to the fact that the decision tree based hardware and software takes too much time to search the exhaustive option set for the correct move.

## II. LITERATURE SURVEY

The chess machine is found to be the ideal one for testing the machine intelligence because of the following facts,

a. The problem is sharply defined both in terms of the allowed operations (the moves) and in terms of the ultimate goal (checkmate);
b. It is neither so simple as to be trivial nor too difficult for satisfactory solution;
c. Chess is generally considered to require "thinking" for skillful play; a solution of this problem will force us either to admit the possibility of a mechanized thinking or to further restrict our concept of "thinking";
d. The discrete structure of chess fits well into the digital nature of modern computers." [2]

The classical approaches to chess playing by machines have evolved over the years. The earliest approach was to evaluate and to store [3], for each chessboard configuration the best move to play. This simplistic and idealistic approach of the problem seems technically difficult as it required huge comparisons and evaluations. The evaluation part is particularly difficult due to the fact that, considering, that a human might purposefully sacrifice a piece to gain advantage in future moves, which is unpredictable and hence indefinable in terms of mathematical functions. The chess engines commonly use the tables of best moves but for several restricted conditions [3]. Even under such restricted conditions, the needed storage size for each table is often several gigabytes.

The next approach is the use of Brute-force method. IBM developed a world class machine, named Deep Blue, which defeated the world champion of the time Garry Kasporov. Deep Blue was the culmination of a multiyear effort to build a world-class chess machine developed using custom hardware. The Deep Blue chess-playing system explores a huge search tree when selecting a move to play, typically examining on the order of 200 million positions per second

[4]. Today's chess-playing programs uses the rules of the game to compute a huge amount of possible moves and the resulting game situation in order to find an optimal move. Deep Blue used the brute force decision tree scheme [3], which requires a large and extensive knowledge. Here the problem is no longer storage but the exponential growth of search space which was successfully solved only due to the use of custom built hardware. [1]

Only one Artificial Neural Network based approach for developing a chess engine is reported in literature by the name of Octavius. Octavius attempts to gain an understanding of chess through a positional analysis of master and grandmaster games. Its training is based on the assumption that any position reached during such games must be positionally superior to any alternatively available position during that game. It has absolutely no hard- coded knowledge of chess. Octavius was originally programmed on a Pentium 266 MHz with 32Mb RAM, but it shows gradual degradation in performance. [5]

## III. PROBLEM DEFINITION

The classical approaches for devising a chess engine using evaluate and store and brute force method were found to have non-definability and large search spaces as problems. We assume a model of humans applying patterns learned from experience instead of employment of brute-force search as in the top chess systems. Brute-force search and human supplied static knowledge bases still dominate the domain of chess as well as other more complex domains. Therefore, this work attempts to create such a system, which conforms more to cognitive models of chess. Many attempts have been made to create a chess program that is both autonomous and adaptive, while still being competitive. A wide range of machine learning techniques have also been tested, but most have met with limited success, especially when applied to playing complete games. [4]

The rules of the game determine which moves (if any) the side to play is allowed to make. In some games, it is easy to look at the board and determine the legal moves: for example, in tic-tac-toe, any empty square is a legal move.

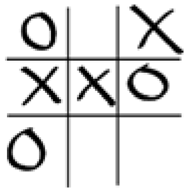
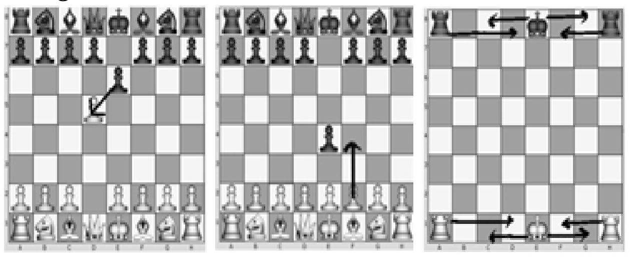


Figure 1.Tic-Tac-Toe game

For chess, things are more complicated, since each piece has its own movement rule. Pawns capture diagonally and move along a file. It is illegal to leave a king in check. The "en passant" captures, pawn promotions and castling moves all requires very specific conditions to be legal. For developing a good chess machine we need to focus on, i) some way to represent a chess board in memory, so that it knows what the state of the game is. ii) Rules for generating legal moves. iii) A technique to choose the move to make amongst all legal possibilities, so that it can choose a move instead of being forced to pick one at random. iv) A way to compare the moves and positions, so that it can makes intelligent choices.



Figures 2.Pawn capture Figure 3.En passant Figure 4.Castling

[Snapshots from ChessBin Version: 0.0.8.0, Adam Berent]

This work is aimed at developing a chess playing program, built using adaptive techniques, that is able enough to compete with humans". However, since there is no absolute solution defined for the problem under consideration, the objectives are better spelled out as:

a. Short term or intermediate – generating legal moves and outcomes following the rules of the games.
b. Long term or final – to reach a decision point for the game.

The problem can therefore be defined as:

To generate a function f, which take as input a chess board position denoted by,

$$\wedge\ (\ \wedge(\boldsymbol{X_i B_{kj}}), \wedge(\boldsymbol{Y_i B_{kj}})\ )$$

Such that,

$$\boldsymbol{f}\Big(\wedge\big(\wedge(\boldsymbol{X_i B_{kj}}), \wedge(\boldsymbol{Y_i B_{kj}})\big)\Big) = \boldsymbol{X_i B_{kj}\ /\ Y_i B_{kj}};(1)$$

Here, X and Y denotes white and black pieces respectively, i = 1,2,…,16 represents total number pieces for each side, k=1,2,…,8 denotes files and j=1,2,…,8 denotes ranks, $X_iB_{kj}/Y_iB_{kj}$ represents the current position of X (white) and Y (black) pieces accordingly.

## IV. ARTIFICIAL NEURAL NETWORK

An Artificial Neural Network (ANN), an emulation of a biological neural network is a parallel system, capable of resolving paradigms that linear computing cannot. An artificial neural network is a system that receives an input, process data and provides an output. The input can be anything such as a data from an image file or any kind of data that can be represented in an array.

The artificial neural networks can be explicitly programmed to perform a task. The topology can be manually created and then the weights of each link and threshold are set. This is the unique strength of neural nets. The supervised learning method trains the artificial neural network to learn from the given dataset of the desired output for many inputs, making up the training set. By the use of the supervised learning technique, the ANN is trained until it maps the given input with the actual output. Since, this project proposes in making the use of the moves made by the human grandmasters in the championship game, hence, ANN is likely to be suitable and effective for such scenario.

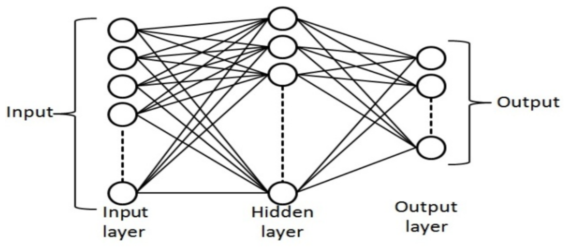


Figure 5.An Artificial NeuralNetwork

## V. PROPOSED METHODOLOGY

The proposed work is aimed at using evolutionary and adaptive techniques in creating a chess playing program. The program would initially learn from games played by others and make moves when faced with fresh scenarios. Learning initiation would be done by presenting the moves played by grandmasters during the championship games from which the system would learn. Further the system/program should evolve strategies and moves, abiding by the rules that govern them and find out ways to beat the opponent.

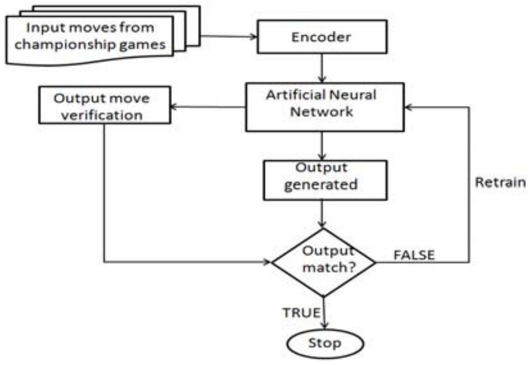


Figure 6.Training Phase

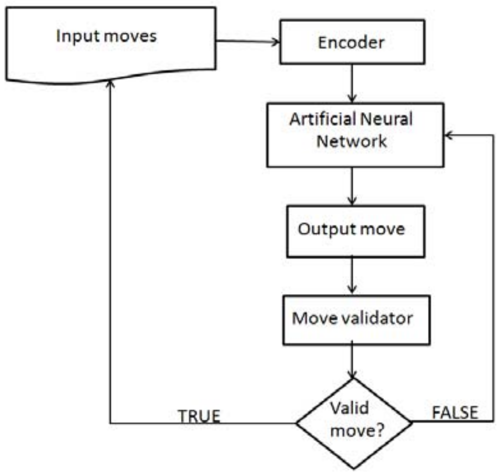


Figure 7.Testing Phase

The strategy is pivoted on an ANN based approach, where when trained, a network responds to literate before unknown scenarios in an intelligent manner. We are trying to device a chess engine using the understanding of the game through a positional analysis of master and grandmaster games. Similar to Octavius proposed, we would try to train the system by means of the patterns generated by the master and grandmaster during the championship games. The use of adaptive Neural Network techniques allows us to train the system to construct a winning position, as well as to develop its pieces which result with the best possible move. The trained system is then deployed in the testing environment where the result will be examined based on its performance. The network structure is of high importance in such ANN based approaches, and the encoding of the problem into ANN understandable format is essential. The strategy is to have 16 inputs and 16 output system where each input connections signifies a piece of the one player (in order) and each output connection stands for a piece of the other player. Each input will be active for only one input, showing a particular piece movement and have values showing from and to squares in the chess board. The output will be encoded likely. The output generated will have to be checked against the rules of the game and would be accepted if i) there is no rule violation, ii) there is no direct benefit granted to the opponent, as visible without using predictive technique (en passant, castling rules, etc.).

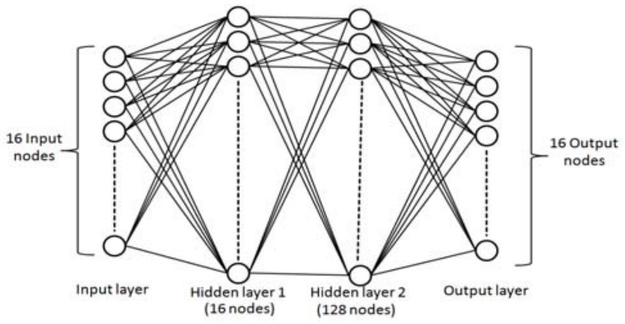
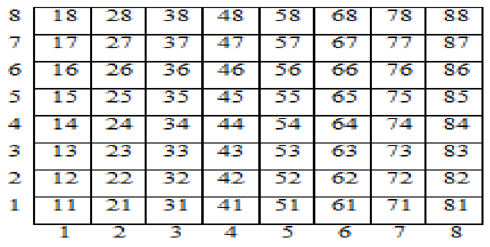


Figure 8.Proposed Neural Network Structure

## VI. EXPERIMENTAL SETUP

Firstly, the ANN Understandable Board is to be obtained where each square in the chessboard is represented with the unique ANN understandable numerals, so that every input and corresponding output moves made by the human players in the championship games can be encoded and utilized further for training the ANN. The idea here is the use of numerical notational technique of the board as shown below,



| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 8 | 18 | 28 | 38 | 48 | 58 | 68 | 78 | 88 |
| 7 | 17 | 27 | 37 | 47 | 57 | 67 | 77 | 87 |
| 6 | 16 | 26 | 36 | 46 | 56 | 66 | 76 | 86 |
| 5 | 15 | 25 | 35 | 45 | 55 | 65 | 75 | 85 |
| 4 | 14 | 24 | 34 | 44 | 54 | 64 | 74 | 84 |
| 3 | 13 | 23 | 33 | 43 | 53 | 63 | 73 | 83 |
| 2 | 12 | 22 | 32 | 42 | 52 | 62 | 72 | 82 |
| 1 | 11 | 21 | 31 | 41 | 51 | 61 | 71 | 81 |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

Figure9.ANN understandable chessboard representation

In the numeric notation all the squares are numbered with a two-digit number [7]. In this simple coordinate system the first digit describes the file and the second one the rank. A move is defined by pairing two of these two-digit coordinates together: the move that would be written as shown in the table.

Table1: Encoding scheme for ANN*representation*

| Notation Representation | White Piece | | Black Piece | |
|---|---|---|---|---|
| | Algebraic | Numeric | Algebraic | Numeric |
| | e4 | 5254 | d5 | 4745 |
| | Nf6 | 7163 | Nf3 | 7866 |

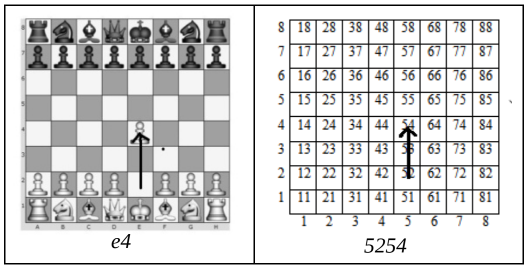


Figure 10.Representation of move e4 in Algebraic and inNumerical notation

The encoding of the moves depicts the combination of the current position of the piece as well as the next position to be moved together within the board. As shown in the figure 10, if a move e4 is to be represented numerically then, the numeric notation 5254 indicates that a piece is to be moved from position 52 to next position 54. The artificial neural network is then trained using these encoded moves so that it learns itself to generate moves when tested in some unknown scenario.

## VII. RESULT AND DISCUSSIONS

The methodology presented above is implemented using the Artificial Neural Network toolbox in Matlab software. The result of the system is found to be very simple and effective as it realizes and hints the intermediate move of a particular piece on the board to be moved next. After training the system using the data (moves) played by the human grandmasters, the system when tested with the set of untrained moves, produced the result that is sufficient enough to justify which particular piece is to be moved.

Table 2: Set of Training data

| Sr. No. | Input | Target Output |
|---|---|---|
| 1 | 5254 | 5755 |
| 2 | 7163 | 7866 |
| 3 | 6355 | 4746 |
| 4 | 5563 | 6654 |
| 5 | 4244 | 4645 |
| 6 | 6143 | 5446 |
| 7 | 8283 | 2847 |
| 8 | 2223 | 4726 |
| 9 | 3122 | 3865 |
| 10 | 2142 | 4654 |

On testing the system with the dissimilar moves, it was found that the trained system clearly signifies which piece is to be moved by denoting the current position of the piece on the chess board. Analyzing the results obtained, it was found that system is able enough to hint the possible move.

Table 3: Experimental results

| Sr. No. | Input move | Output move | Analysis |
|---|---|---|---|
| 1 | 5253 | 5726 | Move from position 57 |
| 2 | 7152 | 7827 | Move from position 78 |
| 3 | 6375 | 5573 | Move from position 55 |
| 4 | 3233 | 4624 | Move from position 46 |

In the above table, it is clear that, when a move 5253 is supplied the system resulted with the output move 5726 suggesting that a piece is to be moved from the position 57 on the chess board. The output generated is found to be an appropriate suggestion for making a move in response.

## VIII. CONCLUSIONS

The ANN's can be trained to learn moves as complex as those of chess games have been long debated upon. Apart from Octavian there has been no mention of ANN applications in chess playing. Even Octavius reports degradation in performance. Our work brings out the fact that ANN's can indeed be used for playing, teaching or helping human in playing chess. We believe that the same approach can be used to develop a full-fledged chess playing machine, which is our aim at present.

## IX. REFERENCES

[1]. Murray Campbell, A. Joseph Hoane Jr., Feng-hsiungHsu, "Deep Blue". IBM, August 1, 2001.

[2]. Claude E. Shannon, "Programming a computer forplaying chess". Philosophical Magazine, Ser.7. Vol. 41,No. 314 – March 1950.

[3]. Nicolas Lassabe, Stephane Sanchez HerveLuga andYves Duthen, "Genetically Programmed Strategies for Chess Endgames". GECCO'06, July 8-12, 2006.

[4]. Robert Levinson and Ryan Weber, "Chess Neighborhoods, Function Combination and Reinforcement Learning". University of California Santa Cruz, Springer, October 2000.

[5]. Robert Epstein, Gary Roberts, Grace Beber, "Parsing the turing test philosophical and Methodological Issues in the Quest for the Thinking Computer". Springer, 03-Dec-2008.

[6]. JoonatanManttari and Jonas Larsson, "Applications of Artificial Neural Networks in Games; An Overview". School of Innovation, Design and Engineering.

[7]. http://en.wikipedia.org/wiki/ICCF_numeric_notation