Chess Engine based on Alpha-Beta Pruning, enhanced with Iterative Deepening

Aayush Parashar
Department of Computer Engineering
B.Tech Undergraduate
Delhi Technological University
Delhi, India
2K18/CO/005
ayushparashar14@gmail.com

Abhinav Gupta
Department of Computer Engineering
B.Tech Undergraduate
Delhi Technological University
Delhi, India
2K18/CO/013
abhinavg247@gmail.com

Aayush Kumar Jha
Department of Computer Engineering
B.Tech Undergraduate
Delhi Technological University
Delhi, India
2K18/CO/004
aayushjha34@gmail.com

***Abstract*--- Chess is a two-player strategy board game played on a chessboard, a checkered game board with 64 squares arranged in an 8×8 grid. The current technological advancements have transformed the way we not only play chess, but study chess techniques and analyze the beautiful game. The idea of making a machine that could beat a Grandmaster human player was a fascination in the artificial community for decades. And with the rapid advancement of artificial learning it has now become possible to create strong chess engines capable of competing at the global level.**

***Keywords*— Chess Engine, Alpha-Beta Pruning, min-max algorithm, Game tree, IDDFS**

I. Introduction

The goal of our project is to design and implement a strong AI chess playing engine. We have implemented the project using Java, Eclipse RCP(for building GUI for the chess game) and using Artificial Intelligence concepts like minimax and alpha-beta pruning algorithms. For creating stronger heuristics, we have extended that model with implementation of Iterative Deepening Depth First Search (IDDFS).

A. Background

The first chess automaton was created in 1770. It had to contain a human inside to do the thinking that does however illustrate the point that people have been fascinated by the complexity of the game and have been trying to create automated chess.
The first electro-mechanical device dates back to 1890 and it could play chess end-games with just the kings and one rook, the so-called KRK end games.
In terms of computer history, computer chess has a long history. Since the 1940's chess has been in the picture of automation and computer science. Much effort was put in developing theory and creation of chess programs on early computers. The paper "Programming a Computer for Playing Chess" by Claude Shannon is recognized as a primary basis for today's computer chess theory.
In 1997 a chess computer called Deep Blue defeated chess grandmaster Gary Kasparov. Recently (June 2005), another super chess computer, Hydra, defeated Micheal Adams with big numbers – the era that computers dominate humans in chess has begun.
Many researchers in the field of computer chess argue that chess is seen as the Drosophila of Artificial Intelligence, but that in contradiction to the insights that the fruit fly has brought to biology, not much progress has been achieved in knowledge of chess for A.I. (Donald Michie ,Stephen Coles).
This observation certainly reflects practice, where chess computers get their strength from brute force search strategies, rather than specific A.I. techniques.
Research on chess mainly focuses on two fields.
– The quest for search efficiency, which resulted in many optimizations to the search algorithms of chess computers. This field concentrates mainly on pruning (cutting) branches from the game tree for efficient search. Donskoy and Shaeffer argue that it is "unfortunate that computer chess was given such a powerful idea so early in its formative stages" . They also note that new ideas,where the search is guided by chess knowledge are being investigated – that the research becomes more A.I. oriented - again.
– Representing chess knowledge in chess computers. This field concentrates on the evaluation of a position on the chess board, in order to differentiate which of the given positions is better. Techniques used to represent knowledge in the evaluation function of chess computers range from tuning by hand, least squares fitting against grandmaster games (used for Deep Thought and later Deep Blue
), to Genetic algorithms [2, 3].

B. Innovation

The motivation behind this project is the research paper published at the Chinese Control and Decision Conference, that explored the features of Artificial Intelligence on game models and developed a Checkers game using Alpha-Beta Pruning.
With the game of checkers board games, the paper proposed an algorithm based on the alpha-beta search and iterative deepening optimization. Through adjusting the search depth dynamically, it solved the problem caused by changes in checkers.
Also, this paper designed a kind of zoning evaluation function considering pawn difference, in which the board is divided into six regions and assessed respectively. To quantify the pawn difference, experiments are conducted to get the reasonable scaling factor of the king and ordinary
The research paper inspired us to extend that model to a Chess-Board and examine the results.

Also, the research paper, Alpha-Beta Pruning in Minimax Algorithm –An Optimized Approach for a Connect-4 Game(2018), presented an enhancement of Alpha Beta Pruning with Min-Max Algorithm for a Connect-4 game. According to that, a user will be able to log in to the system where he / she can challenge various levels of AI with increasing degree of difficulty. The moves of the AI will be optimized by using minimax algorithm and alpha beta pruning.
We have analyzed the process and implemented some part of that algorithm to our model to make the model better.

II. Game Tree of the system

Chess is a zero-sum, deterministic, finite, complete information game.
Zero-sum means that the goals of the competitors are opposite – a win is a loss for the opponent. In case of a draw, the sum of the game is zero. Suppose a position could be valued as +10 for one player, then the score for its opponent is and must be -10.
Deterministic means that all possible games are either winning for white, winning for black, or a draw.
Finite, because games cannot not take forever (three repetitions of a position result in a draw, also the 25 move rule, where 25 moves occur without a hit or a pawn move is a draw). Complete information means that there is no hidden information or uncertainty like in a card game such as poker and the game is played in a sequential fashion: both players have the same information.
A two player game such as chess, can be represented by a tree. Each node is a position on the board, starting at the root with the beginning position of the game. The vertices then are the possible moves, leading to the next nodes (positions). This way, every possibility after one move can be generated and added to the tree. A game after it has been played can be viewed as a path in the "game tree".
A single move by black or white is called a ply. A (full) move contains the move from both white and black. Sometimes a ply or half move is also called a move; the terminology in the literature is inconsistent.
In the example below a simplified game tree.

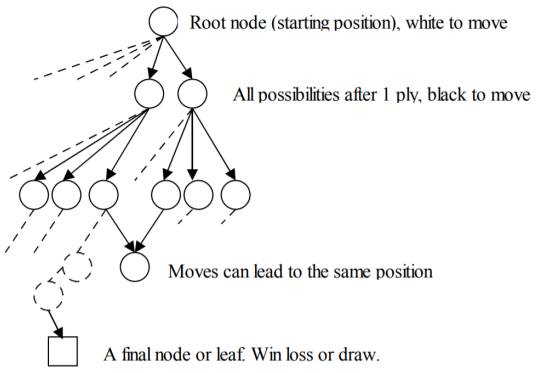

Fig. 1. Game Tree of the System

III. Proposed Work

For the development of an Intelligent Model that can take valid and optimized moves on a Chess Board, we use the following concepts.

A chess program usually contains three modules that make playing chess possible. These are:
– move generator
– evaluation function
– search algorithm
The move generator module is used to recursively generate the (legal) moves and form a game tree.

The evaluation function assigns a value to the leafs of the tree. The search method finds the path to the best position (value) in the tree. Logically, when a winning position is found, the evaluation function will return the highest possible number within the system, because it is the best position possible.

The values for the opponent should be the exact opposite, negated value (zero sum property). All other non winning or non final leafs will get a numerical value between these maximum values, assessed by the evaluation function. How the search engine uses this information to find a move will be covered in the section about search algorithms.

A. Algorithms Used

Min-Max Algorithm: Min-max algorithm is a recursive or backtracking algorithm that is used in decision-making and game theory. It provides an optimal move for the player assuming that the opponent is also playing optimally. Min-Max algorithm uses recursion to search through the game tree and is mostly used for game playing in AI.
In this project, we use the min-max algorithm to create a Game-Tree with which the AI system will take a valid step.

Alpha-Beta Pruning: Alpha-beta pruning is a modified version of the minimax algorithm. It is an optimization technique for the minimax algorithm. It is a technique by which without checking each node of the game tree we can compute the correct minimax decision, and this technique is called pruning. This involves two threshold parameters Alpha and beta for future expansion, so it is called alpha-beta pruning. It is also called the Alpha-Beta Algorithm.
In the Game-Tree Designed for the chess board, we will use Alpha-Beta Pruning method to remove the nodes which are not necessary, and hence optimizing our search.

Iterative Deepening Depth First Search (IDDFS): There are two common ways to traverse a graph, BFS, and DFS. IDDFS combines depth-first search's space efficiency and breadth-first search's fast search (for nodes closer to root). Iterative deepening depth-first search (IDDFS) is an algorithm that is an important part of an Uninformed search strategy just like BFS and DFS. We can define IDDFS as an algorithm of an amalgam of BFS and DFS searching techniques.

In the project, we use IDDFS to optimize our AI system and make moves faster. We start with a search depth of 5 and constantly increments the depth and restarts the search as long as there is time remaining for a certain level.

B. Code Work

In the project, the following technologies have been used.

Java: Java is a class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible.

JFrame: The JFrame class is slightly incompatible with Frame. Like all other JFC/Swing top-level containers, a JFrame contains a JRootPane as its only child. The content pane provided by the root pane should, as a rule, contain all the non-menu components displayed by the JFrame.

Main.java :This is the first program file of the project. Here, the functions of all the classes are combined and presented in the JPanel. The main function collaborates the functionality of the User Interface, JFrame, JPanel, Menu Panel and the Depth Panel. The Game Function is initiated from this class and all the other classes are integrated.

Evaluate.java :This class is responsible for the evaluation of every move taken by any player and provides the findings in the format specified by the user at the start time. The Evaluation can be done on the basis of 4 factors: Material, Attack, Movability and Position. As and when the player makes a move, that move is evaluated and stored in the game.

User Interface.java :This class is responsible for providing a User Interface to the game. Here, the png images of chess pieces are rendered on specific locations according to the chess board image. The Icons are loaded from the assets and rendered using JFrame library functions.

Move Generator.java :This class is the heart of the program. This class contains the application of algorithms like Alpha Beta, Min-max Algorithm, Game Trees and Iterative Deepening Search.
The Game Tree is constructed on the basis of the depth provided by the users at the beginning and it is evaluated in a similar function according to the value of every chess piece provided at the initiation of the game.

C. Pieces Value and Evaluation Criteria

Chess piece values indicate the value of the different chess pieces and how they relate to each other. Every piece has different strengths and weaknesses, so they are valued differently. Chess piece values give us a relative worth for each piece. This information helps us determine what piece should be traded for another piece, how we evaluate an exchange, and even how computer engines evaluate a position!

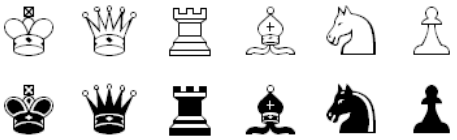

Fig. 2. Chess Pieces

A pawn is worth one point, a knight or bishop is worth three points, a rook is worth five points and a queen is worth nine points.
The king is the only piece that doesn't have a point value. This is because the king cannot be captured (an attacked king is in check), and also because checkmating the king is the true goal of any chess game.

Engine Evaluations - Engines give evaluations based on a numerical assignment. These evaluations are directly correlated to chess piece values! For example, a +1 evaluation means that White is ahead one point (the value of a pawn) while a -1 evaluation translates to Black being ahead one point. If a computer engine evaluates a position as +5, then White is leading by the value of a rook.
Engines often give an evaluation with a decimal, like -1.5. This should be interpreted as Black being ahead by one and a half pawns.

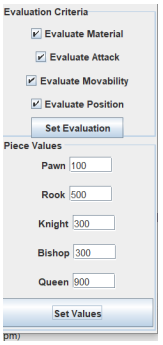

Fig. 3. Evaluation Criteria

IV. FINDINGS

Following the steps mentioned above, we successfully execute our chess engine based on alpha beta pruning.

We observe that as the value of IDDFS search depth is increased or decreased, the computing time for the next AI move i.e. opponent is increased or decreased respectively, since the game tree is explored deeper for higher depths before finalizing the move.

Fig. 4. Screenshot of the Chess Game

Fig. 5. A Knight Making a Move

Fig. 6. White Queen Making a Move

V. Future and Scalability

The project can be extended to implement the concepts of Deep Learning and Neural Networks. Decision Making Deep Reinforcement Learning is a concept that uses Deep Reinforcement Learning to optimize decision making through previously defined strategies. The main idea is that from a diverse set of good strategies, the algorithm can choose the best of them for each situation faced. These concepts can be used to extend the implemented AI Model in Chess.

Another way the Artificial Intelligence Model can be enhanced is by using the *Sibling Prediction Pruning*. The maximum positional difference (MPD) between siblings is the property that will be used by SPP. SPP can be used as a replacement for AB(Alpha-beta) on a system with insufficient or slow memory, such as a mobile phone. As a single enhancement it could be more efficient than AB for positions with a high branching factor.

VI. Conclusion

The aim of the project was to build a chess game using Alpha-Beta Pruning , min max algorithm and enhancement using iterative deepening. The proof of concept on Alpha-Beta pruning is a success. It is a new forward pruning method without information loss regarding the minimax value of the game tree. This is an important property, because information loss is the down-side of other forward pruning mechanisms.The resulting algorithm is a forward pruning method on leafs of the game tree, which gives correct minimax results. A maximum positional difference of the evaluation function on siblings must be correctly measured or assessed for the algorithm to work properly.The results of this project cannot be generalized because of the dependencies on the evaluation function, but are intended as a proof of concept and show that it is worthwhile to investigate Pruning Alpha-Beta in a broader context.

References

[1] Zhao, Z., Wu, S., Liang, J., Lv, F., & Yu, C. (2014). The game method of checkers based on alpha-beta search strategy with iterative deepening. The 26th Chinese Control and Decision Conference (2014 CCDC) : https://ieeexplore.ieee.org/document/6852758

[2] Rijul Nasa, Rishabh Didwania, Shubhranil Maji, Vipul Kumar. Alpha-Beta Pruning in Minimax Algorithm –An Optimized Approach for a Connect-4 Game. International Research Journal of Engineering and Technology (IRJET) (2018): https://www.irjet.net/archives/V5/i4/IRJET-V5I4366.pdf

[3] Study on the algorithm based on the important region of board: https://ieeexplore.ieee.org/document/8408295

[4] A New Paradigm for Minimax Search: https://arxiv.org/ftp/arxiv/papers/1404/1404.1515.pdf

[5] Comparison of A* and Iterative Deepening A* algorithms for non-player character in Role Playing Game: https://ieeexplore.ieee.org/document/8167134