Artificial Intelligence – A Paradigm of Human Intelligence

Mr. Saurabh S. Maydeo*1, Mr. Amit S. Hatekar*2

*1 Undergraduate student, Department of Information Technology, Thakur College of Engineering and Technology
Kandivali East, Mumbai, Maharashtra, India

*2 Assistant Professor, Department of EXTC, Thadomal Shahani Engineering College
Bandra West, Mumbai, Maharashtra, India

Abstract—In today’s world, Artificial Intelligence is evolving at great speed. In 1997, Deep Blue’s win against the chess master Garry Kasparov was seen as very symbolically significant, depicting that artificial intelligence was reaching up to human intelligence [6]. This research paper presents the new approach for chess engine implementation which can be layered on the top of Deep Blue algorithm and compares it with different existing types of Chess Engine implementation algorithms such as Minimax algorithm, Alpha-Beta Pruning and Deep Blue approach. All the four approaches take current chess board as an input and give best move as an output. This research provides a detailed study of how all the three approaches work and their advantage, disadvantage and then compares them on the basis of various parameters such as branching factor, space complexity, time complexity, etc.

Keywords—Alpha-Beta Pruning, Artificial Intelligence, Branching Factor, Chess Engine, Deep Blue, Minimax.

I. INTRODUCTION

According to the father of Artificial Intelligence, John McCarthy, it is “The science and engineering of making intelligent machines, especially intelligent computer programs”.

While exploiting the power of the computer systems, the curiosity of human, lead him to wonder, “Can a machine think and behave like humans do?”

In our day to day life for achieving something we think of all possible ways or actions to do it. Then at the back of our mind we assess every action and try to figure out its possible reactions. Our mind rejects all those actions which can cause some harm to achieve desire and select best possible action amongst them.

A similar case is there in chess game. We take into consideration all possible moves of all pieces. Then at the back of our mind we try to figure out what will happen if we make this move.

We try to judge from our opponent’s side too. We try to figure out if we make a certain move, what will be the next move of our opponent?

In this way we compare every move and its consequences with other moves and their consequences, and play the best possible selected move.

The basic job of a chess engine is to analyse the given board ‘B’ and return best move ‘M’, i.e. F (B) = M.

There 3 popular algorithms for implementing chess engine:

A. Minimax Algorithm
B. Alpha-beta Pruning Algorithm
C. Deep Blue Algorithm

II. RELATED WORK

A. Minimax Algorithm

The Minimax algorithm can be applied in the games that are played by two players, such as tic-tac-toe, chess, checkers, etc.

In chess, we can come to know from any given point, what are the next available moves of every piece on the board.

Thus, logically tee-like structure gets formed though recursion. Each node in this logical tree represents move. Search trees are a way to represent searches. There are two players involved, MAX and MIN. The search starts from the root node. At each decision point, nodes for the available moves are generated, until no more decisions are possible. A search tree is traversed in depth-first search approach. The nodes that belong to the MAX player are assigned the maximum value of its children. The nodes for the MIN player will select the minimum value of its children. How good a move is depends on these values. Thus, the MAX player selects the move having highest value in the end, whereas MIN player selects the move having lowest value.

At a node, MAX would pick the move which gives and MIN would pick the node which gives the lowest Minimax value (thereby giving MIN the highest utility).

Minimax tends to be too slow for games such as chess [1]. For each turn, at any instance of time, the player has many choices of pieces and their corresponding moves. This increases branching factor and due to large branching factor the deeper we go, the time consuming it gets. The average
branching factor for the game of chess is normally 30. This is, 30 sub trees per turn are created

Advantages:
1. It was the first algorithm that made it possible to implement decision making of artificial intelligence in chess.

Disadvantages:
1. Takes huge amount of time for generating best move.
2. Game tree time complexity (b^n) \approx 10^{154} in chess.
3. Practically exploring entire tree is not feasible since there are restrictions on space and time.
4. Branching factor is too high, i.e. average number of children to a node is \approx 35 in chess
5. Search and evaluation of unnecessary nodes degrades overall performance and efficiency of the engine.

B. Alpha-beta Pruning Algorithm
This algorithm overcomes drawbacks of Minimax algorithm. Minimax searches entire tree even if in some cases the rest of sub tree can be ignored.

Alpha-beta pruning is an algorithm that tries to reduce the number of nodes which are explored by the Minimax algorithm in its search tree. Alpha-beta is layered on the top of Minimax algorithm. In the best case, we can get the same optimal solution as Minimax in O(b^{d/2}) instead of O(b^d). Equivalently, we can reduce the branching factor to \sqrt{b} from b.\[2\]

The basic idea of alpha beta pruning of cutting or avoiding search and evaluation of sub tree which can be ignored makes it very much faster than Minimax algorithm where we have to go through all ways. Also it reduces space complexity as fewer function calls take place as we cut unnecessary parts of tree.

Advantages:
1. It reduces branching factor from b to square root of b.
2. As there are lesser function calls than Minimax algorithm, it has reduced space complexity
3. This approach doesn’t search and evaluate unnecessary nodes of game search tree.

Disadvantages
1. This approach doesn’t suggest to maintain dictionary or look up table for starting moves.
2. It is slower than Deep Blue as it doesn’t use observed strategically proven moves from grandmasters’ game move database.

Fig. 2 Game Tree search and evaluation in Alpha-Beta Pruning
Alpha represents the maximum score the maximizing player is assured of, Beta the minimum score the minimizing player is assured of. If beta \leq alpha, it means a maximizing parent node is guaranteed higher score on another branch, or a minimizing parent node a lower score on another branch. In such scenario, this branch can be culled and the rest of this node's children can be skipped.[8] The order of exploration of the branches affects the algorithm. The sooner the best moves are discovered the sooner worse branches can be discarded. In the optimal case alpha-beta pruning can search to twice he depth with the same amount of computation speed as Minimax.[7]

Advantages:
1. It reduces branching factor from b to square root of b.
2. As there are lesser function calls than Minimax algorithm, it has reduced space complexity

Disadvantages
1. This approach doesn’t suggest to maintain dictionary or look up table for starting moves.
2. It is slower than Deep Blue as it doesn’t use observed strategically proven moves from grandmasters’ game move database.

C. Deep Blue Algorithm
Deep Blue computer chess system, developed at IBM Research during the mid-1990s. Deep Blue is the chess machine defeated the World Chess Master, Garry Kasparov, in a 6 game match in the year 1997. There were number of factors, which contributed to this AI success, such as a single-chip chess search engine, a lot of parallel computing, a strong focus on search extensions and effective use of a Grandmaster game database.

Deep blue approach can be viewed as:-
Minimax + Alpha-Beta + Progressive Deepening + Parallel Computing + Opening Book + End-Game Strategies + Uneven Tree.

In Deep-Blue, Alpha-Beta is layered on the top of Minimax with the concept of progressive deepening. When we use the depth-first search, even if the nearest nodes of the root node on the right sub tree is having checkmate condition, then also it will be evaluated after full evaluation of left sub tree. Thus, parallel computing speeds up the decision making process in such cases. This parallel computing can be implemented by the means of multithreading [5]. Just like human chess masters, Deep-Blue system also uses some strategically-proven opening moves without evaluating the whole board on the starting of the game. These opening moves are stored in the database, which can be called as opening book[4]. Similarly, it also maintains strategically-proven end-game moves and uses them for ending the game in less time. Along with opening book and end-game book, it also uses extended book. These books or databases comprise of strategically-proven moves from Grandmasters’ games. Using Alpha-Beta pruning that is layered on the top of Minimax, this system already guarantees of avoiding unnecessary moves by cutting the sub trees, which are not required for evaluation, but still if we specify depth of search as say 8, then that is the maximum limit for search and say in some cases opponent is defeating after 8th deepness level. To handle this, Deep-Blue system is made capable of deciding where to search further and on which depth to stop. This makes the game search tree uneven.

Advantages:
1. There are lesser function calls than Minimax algorithm, it has reduced space complexity.
2. As this system maintains opening moves’ book it is faster than simple Alpha-Beta and Minimax approach.
3. Parallel computing reduces time complexity in some check mate situations.

Since it has capability to determine whether searching further is needed or not (uneven tree), it increases performance.

D. Proposed Work
This approach can be layered on the top of Deep Blue approach.

As we know, deep-blue system uses opening book, end-game book and extended book, which contains strategically-proven best moves based on observations of Grandmasters’ games. If the rating of the specific move in similar situation is high, then the more emphasis is given on evaluating that move first. This approach avoids search and evaluation of whole game tree and saves time.

Normally, while playing chess, whenever any human plays certain moves, if he finds that this move is successful; he tries to remember that move for playing it for the next time.

Though human being has a limitation of not being able to memorize huge number of moves, whereas in computerized chess game system, this limitation can be dealt with.

Computerized chess game will be played with billions of players. We can make a centralized database, which can analyse the play and record the successful and timesaving moves, and it can maintain the rating of success of all these moves.

Thus, along with Grandmasters’ games, the chess engine can also use its own database of successful moves in the next game.

Initially chess engine searches for current situation of board in database. If similar situation is found, then the move having highest rating is evaluated. After evaluation, if chess engine also finds it successful move then it increments its rating in database and plays that move.

If the similar situation is not present in database, chess engine finds best move through deep blue approach and stores current situation and successful move with its initial rating into database.

Since many engines can interact with server, the higher the rating of the move in particular situation, the higher the probability of it being successful. This is because when engine finds current situation in database, it evaluates the moves stored corresponding to it and increments the rating if it finds particular move successful. Thus this approach can save a lot of time.
E. Comparison

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Minimax</th>
<th>Alpha-Beta</th>
<th>Deep Blue</th>
<th>Proposed Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Branching Factor</td>
<td>The effective branching factor is equal to the mean branching factor, i.e. equal to $b$.</td>
<td>The branching factor is reduced to about square root of the average branching factor, i.e. $\sqrt{b}$</td>
<td>The branching factor is reduced to about square root of the average branching factor, i.e. $\sqrt{b}$</td>
<td>The branching factor is reduced to about square root of the average branching factor, i.e. $\sqrt{b}$</td>
</tr>
<tr>
<td>Speed of searching best move</td>
<td>Very slow (Hence impractical to implement)</td>
<td>Moderate</td>
<td>Fast</td>
<td>Faster than Deep Blue</td>
</tr>
<tr>
<td>Run time Memory requirement</td>
<td>Too large, since many function calls take place due to inability to avoid searching unnecessary nodes.</td>
<td>Less than Minimax</td>
<td>Less</td>
<td>Less</td>
</tr>
<tr>
<td>Strategic moves’ Database</td>
<td>Not present</td>
<td>Not present</td>
<td>It uses database for moves such as opening move book, extended move book</td>
<td>It creates &amp; uses databases of strategic moves generated by its own observations</td>
</tr>
</tbody>
</table>

Fig.3 Flowchart for recording and using observations made by computer chess engine.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Minimax</th>
<th>Alpha-Beta</th>
<th>Deep Blue</th>
<th>Proposed Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ability to avoid search and evaluation of unnecessary nodes</td>
<td>Unable.</td>
<td>It is capable.</td>
<td>It is capable.</td>
<td>It is capable.</td>
</tr>
<tr>
<td>Parallel computing</td>
<td>Not implemented.</td>
<td>Not implemented in pure Alpha-Beta approach.</td>
<td>Implemented by the ways like multithreading</td>
<td>Implemented by the ways like multithreading</td>
</tr>
<tr>
<td>Ability to learn from self observation</td>
<td>unable</td>
<td>unable</td>
<td>unnable</td>
<td>It is capable.</td>
</tr>
</tbody>
</table>

Table 1: Comparison between Minimax, Alpha-beta Pruning, Deep Blue and Proposed algorithm

### III. CONCLUSION

In this paper, different approaches for chess engine implementation are discussed and compared. After analysing the working of all the four approaches, Deep blue seemed to be better than Minimax and Alpha-Beta Pruning algorithms and the proposed approach seemed to be the best choice among all four algorithms if it is layered on the top of Deep Blue.

### REFERENCES


