AI Chess Algorithms And Chessus

Egemen Kaya
Mendax Software/Ai Exploration
x@mendaxyazilim.com

Abstract

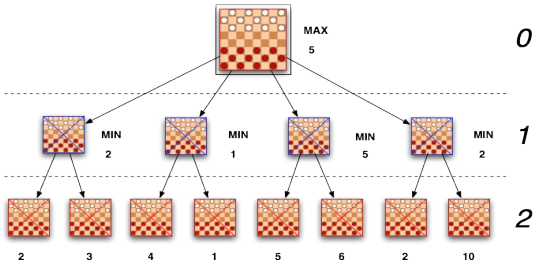
Ever since the advent of Artificial Intelligence (AI) [1], game playing has been one of the most interesting applications of AI. The first chess programs were written by Claude Shannon and by Alan Turing in 1950, almost as soon as the computers became programmable. Games are very appealing and writing game-playing programs is perhaps even more exciting. Unlike other chess engines, we do not allow algorithms to focus on beating the opponent. With the help of the algorithms we use (minimax and alpha-beta pruning), we have developed a chess engine that tries to find the worst situation and get rid of it. We should not expect game playing algorithms to be perfect for every situation. Remember the main purpose of the algorithm before play. The aim of the algorithm is not to beat the opponent but to get rid of the worst situation.

1 Introduction

Minimax is a kind of backtracking algorithm that is used in decision making and game theory to find the optimal move for a player, assuming that your opponent also plays optimally. It is widely used in two player turn-based games (Tic-Tac-Toe, Chess, etc.). Alpha-beta pruning is a search algorithm that tries to reduce the number of nodes evaluated by the minimax algorithm in the search tree. It is a widely used opponent search algorithm for machine play of two-player games. Chessus algorithm blends alpha-beta pruning and minimax algorithms, it aims to find out how to get out of the worst situations instead of defeating or drawing against other chess algorithms.

2 Minimax Algorithm

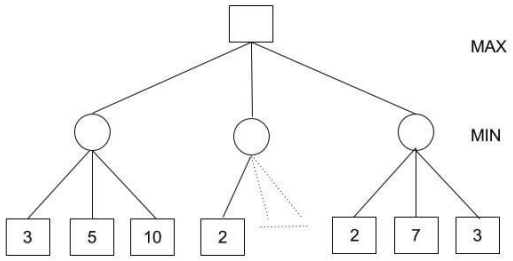
Minimax algorithm [2] aim to create a search tree from which the algorithm can chose the best move. In this algorithm, the recursive tree of all possible moves is explored to a given depth (Chessus has five search depth), and the position is evaluated at the ending “leaves” of the tree. After that, we return either the smallest or the largest value of the child to the parent node, depending on whether it’s a white or black to move. Algorithm try to either minimize or maximize the outcome at each search depth level.



Minimax Algorithm Diagram [3]

3 Alpha-Beta Pruning

Alpha-beta pruning [4] is an optimization method to the minimax algorithm that allows us to disregard some branches in the search depth tree. Alpha-beta pruning helps us interpret the minimax search tree much deeper, while using the same resources. The alpha-beta pruning is based on the situation where we can stop evaluating a part of the search tree if we find a move that leads to a worse situation than a previously discovered move. The alpha-beta pruning does not influence the outcome of the minimax algorithm — it only makes it faster. The alpha-beta algorithm also is more efficient if we happen to visit first those paths that lead to good moves. We developed the Chessus algorithm to provide all this and increase the functionality of the algorithm.



Alpha-Beta Pruning Diagram [5]

3.1 Move Ordering in Alpha-Beta Pruning

The effectiveness of alpha-beta pruning is highly dependent on the order in which each node is examined. Move order is an important aspect of alpha-beta pruning [6]. It can be of two types:

Worst ordering: In some cases, alpha-beta pruning algorithm does not prune any of the leaves of the tree, and works exactly as minimax algorithm. In this case, it also consumes more time because of alpha-beta factors, such a move of pruning is called worst ordering. In this case, the best move occurs on the right side of the tree. The time complexity for such an order is O(bm).

Ideal ordering: The ideal ordering for alpha-beta pruning occurs when lots of pruning happens in the tree, and best moves occur at the left side of the tree. We apply DFS hence it first search left of the tree and go deep twice as minimax algorithm in the same amount of time. Complexity in ideal ordering is O(bm/2).

4 Chessus Heuristic

Chessus is the chess engine that we use for moves and chess logic. We use 5 levels of search depth to find the best move based on the following heuristics:

- material (total piece count for each player)
- number of possible legal moves with emphasis on center squares
- check/checkmate status
- pawn structure

There are many factors when calculating the heuristics [7] of a board. As we developed our heuristic formula to consider more factors, the computations required to calculate the best move increased exponentially. At the moment, the AI considers the following 4 aspects of a board in its heuristic function.

5 Conclusions

Games are very appealing and writing game-playing programs is perhaps even more exciting. We should not expect game playing algorithms to be perfect for every situation. With the methods (minimax, alpha-beta pruning) we programed a chess algorithm to play chess. We developed the Chessus algorithm with the help of the minimax and alpha beta pruning algorithms. Chessus algorithm (only ai part) is just 200 lines of code. Remember the main purpose of the algorithm before play. The aim of the algorithm is not to beat the opponent but to get rid of the worst situation. You can check out the final version on our website [8].

References

[1] https://courses.cs.washington.edu/courses/csep590/06au/projects/history-ai.pdf

[2] https://www.baeldung.com/java-minimax-algorithm

[3] https://iq.opengenus.org/minimax-theorem-algorithm-von-neumann/

[4] http://web.cs.ucla.edu/~rosen/161/notes/alphabeta.html

[5] https://www.hackerearth.com/blog/developers/minimax-algorithm-alpha-beta-pruning/

[6] https://www.javatpoint.com/ai-alpha-beta-pruning

[7] https://link.springer.com/article/10.1007/BF00974979