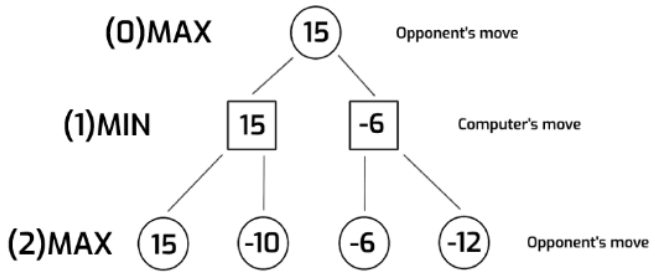
Chess Board Artificial Intelligence

Introduction

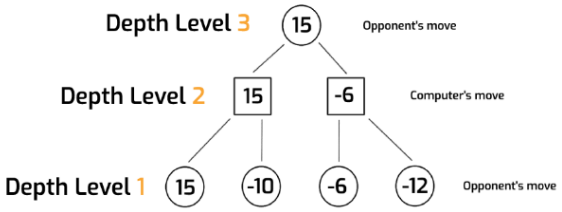
This term for module COMP250 Artificial Intelligence project I have decided to make a chess board AI, that can hold up a decent gameplay against an opponent. This means that setting goals and understanding how this AI works is crucial in my success.

Technologies that I will be using for the AI artefact

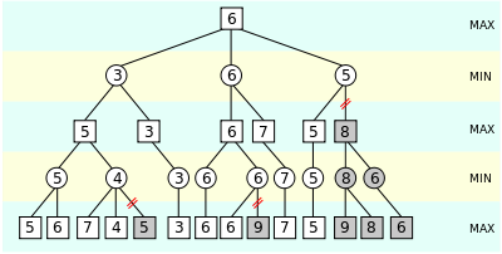
- An algorithm called “Minimax”, developed by John von Neumann in 1928, is used to decide the best move in a two-player game like chess or tic-tac-toe. Assuming that both players would play optimally, it analyses all potential moves and countermoves from the behavior tree and selects the move that results in the best result for the present player. Essentially, the main purpose of this algorithm is to minimize the maximum loss for the player.



- Fixed-Depth search tree. The reason for limiting the depth to which the AI can “see” in the search tree is to optimize the speed of the AI as the estimated size of the search tree is around 40*10^120. We don’t want to wait an eternity for it to pick “the most optimal” move. That’s why I am going to limit the depth of plies.



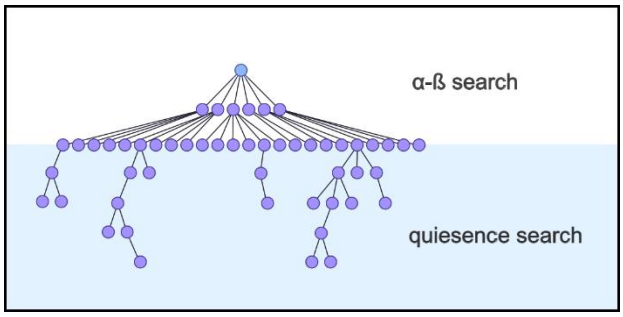
- With limiting the depth of the search tree that the AI can look through, I will encounter a problem of the AI having difficulties finding the best move. To overcome this limiting problem I will



1. Alpha–beta pruning

implement the Alpha-Beta pruning technique with which the AI will be able to discard of any moves that are less rewarding that other, in return having a smaller number of moves to choose from, in-turn making it more efficient.

- I will also be using Quiescence Search (Q-search) algorithm to overcome the Horizon Effect first described by Hans Berliner. The Q-search algorithm will help the AI in looking for the most optimal moves. I will implement this algorithm so that it finds the best Q-value move when the AI reaches the final ply. The algorithm will enter a special search mode that expands the possibilities of certain moves and chooses the one that leads to the most stable position.



2. Quiescence Search

High Concept

The main goal of this chess AI is to make it intelligent enough to hold up a fair and interesting gameplay against the opponent. Also, showcasing my ability to implement different ways of improving the AI as time goes on. A stretch goal would be to allow the user to pick between different difficulty of the AI (Beginner,Intermediate,Advanced) so that people who are not familiar with chess can have a ladder to climb, enjoying the process of learning as they progress through the difficulty levels.

Relation between my specialism and the artefact

As l am studying Computing for Games l expose myself to a broad range of programming concepts. This AI artefact will illustrate my competence in understanding how AI works using and implementing different technologies: Minimax, Alpha-Beta pruning, Q-Search. Not only is this a great way of showcasing my programming, learning and problem solving skills but it's also a great project to have in my portfolio.

Why choose this project?

I chose to develop a chess AI because l find it interesting how AI works and what

perspective you have to take to fully understand it and be able to develop one yourself. This AI artefact will also prove to be helpful to whoever is interested in learning chess or understanding the logic behind the game as I will use an open-source license to license it.

Are there other alternatives out there?

Yes, many great chess AI artefacts are already out there, which can easily beat any person playing the game. Just to name the most popular ones in 2023: Chess.com, Lichess.org. Talking about the history of developing an AI that can beat any person began with a computer called “Belle” developed in 1979 by Ken Thompson and Joe Condom. In 1997, another chess-playing computer named “Deep Blue” developed by IBM beat the world champion Garry Kasparov. Although my AI artefact won’t be able to beat world champions it’s going to prove effective in teaching beginners to play chess whilst demonstrating its complexity with changeable difficulty options(stretch goal).

How will I manage the research and development process?

The key in developing an AI artefact that is intelligent enough to hold up a game of chess against an opponent is to first understand the base logic of the programming techniques used. For Minimax algorithm I will be looking into [1], for fixed depth search tree I will be looking into [2], for Alpha-Beta pruning algorithm I will be looking into [3], and finally for Quiescence Search algorithm I will be looking into [4]. These research papers will prove to be of great use as I will gain the base knowledge of these key facets in creating a chess AI artefact. Also in the future it might be a great way of remembering the core ideas behind a simple AI, this will be a good example of it.

Practice-based research

To develop an AI Artefact that is sufficient to play against different levels of opponents I will need to iterate on each game played by the AI and log its results. This means I will have to tweak and adjust parameters and settings of the artefact to make it as efficient as possible. I will have to iterate through many games to understand the optimization possibilities of the artefact and achieving the stretch goals discussed earlier.

Scope

As far as scope goes l am certain that l will be able to deliver the project until set deadline. Although I've set myself a stretch goal of having AI artefact difficulty options l will first try to deliver on the first objective – implementing Minimax with Alpha-Beta pruning, after that, my next goal will be to implement the Quiescence search algorithm and finally as a stretch goal l will try finding a way of making it possible for the player to set a difficulty option for the AI artefact.

References:

[1] - Sukharev, A. Minimax models in the theory of numerical methods. Vol. 21. Springer Science & Business Media, 2012.

[2] - Plaat, Aske, et al. "Best-first fixed-depth game-tree search in practice." IJCAI. 1995.

[3] - Abdelbar, Ashraf M. "Alpha-Beta Pruning and Althöfer's Pathology-Free Negamax Algorithm." Algorithms 5.4 (2012): 521-528.

[4] - Schadd, Maarten, and Mark Winands. "Quiescence search for stratego." Proceedings of the 21st Benelux Conference on Artificial Intelligence. Eindhoven, the Netherlands. Vol. 102. 2009.