Artificial Intelligence Project Report

A.I. Chess

(UCS411)

Submitted by:

(102103630) Dilbagh Singh Sidhu

BE Second Year, COE

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

Submitted to:

Mr. Amardeep Singh

Index

Introduction

Artificial intelligence is a revolution in itself with numerous feats of accomplishments. The use of AI in the real world and real-life scenarios is ample. It has a wide array of use cases to improve the quality of life in general. Another wonderful use of artificial intelligence is in chess.

Computer chess programs consider chess moves as a game tree. In theory, they examine all moves, then all counter-moves to those moves, then all moves countering them, and so on. This evaluation continues until a certain maximum search depth or the program determines that a final "leaf" position has been reached (e.g., checkmate).

In this project I aim to create a simple chess artificial intelligence-based chess engine.

Link to code repository in references[1].

Literature Survey

1) According to Maharaj, Shiva, Nick Polson, and Alex Turk. "Chess AI: competing paradigms for machine intelligence." *Entropy* 24.4 (2022): 550.[2]
   Chess engine research has implications for problems requiring large-scale tree search. The Shannon number, 10^120, provides a lower bound on the total number of possible games, making chess a daunting computational challenge. Since it is not practical to consider such large number of cases methods like alpha-beta pruning must be used to find solutions.
2) According to Gobet, Fernand, and Guillermo Campitelli. "Educational benefits of chess instruction: A critical review." (2006): 124-143.[3]
   Factors like Richness of the problem-solving domain, knowledge representation challenges, readily available expertise make chess an interesting domain of ai.
3) According to Scheucher, Anton, and Hermann Kaindl. "Benefits of using multivalued functions for minimaxing." *Artificial Intelligence* 99.2 (1998): 187-208.[4]
   Minimaxing has been very successful in game-playing practice, although a complete explanation of why it has been that successful has not yet been given.
4) Droste, Sacha, and Johannes Fürnkranz. *Learning of piece values for chess variants*. Tech. Rep. TUD–KE–2008-07, Knowledge Engineering Group, TU Darmstadt, 2008.[5]
   Elaborates on relative values of chess pieces.

Methodology

The project is split into two python files. The first one majorly just contains the driver code.
The second file contains chess engine code and is implemented using oops paradigm.
Step by step procedure is discussed below.

1) Once game starts engine creates an instance of its class. __init__ constructor is called to initialise depth of search tree and colour the player has chosen.
2) For every computer turn getBest () is called. It is responsible for choosing best possible move. It then calls engine ().

a) Working of engine ()

1) Engine is implemented using recursion and on first call creates list of all legal moves then recursively calls itself for every legal move.
2) It takes two parameters first one is the value used for alpha-beta pruning for possible move that can be made, on first call it is none. The second parameter is depth of the move and on first call is equal to one.
3) Once it reaches leaf node i.e., either max depth or there is no possible move after that state eval () is called. eval () assigns a numerical value to that state.
4) Minmax and alpha-beta pruning are used to choose moves.

b) Working of eval ()

It just adds up value of every square on chessboard as returned by SquareValue () and also adds to it value returned by CheckMate ().

Since values of two nodes can be same it also adds a random number less than one two make values different. This is done as it increases chance that two values will be different and will make code efficient as more pruning can be done.

c) Working of SquareValue ()

It returns value of piece on a particular square.

There are several conventions to define relative values.

Hans-Berliner is used by us.

d) Working of CheckMate()

It returns infinite value if board would be put in checkmate condition else returns zero.

Result And Future Scope

An A.I. based chess engine using python has been created. It uses minmax algorithm and alpha-beta pruning.

Since now we have a working engine future improvement like using variable piece values can be implemented to further improve the results.

References

1. https://drive.google.com/drive/folders/1rG0737JNFuORAsOoNL7wr_8sfr6X5UTx?usp=sharing

2. https://www.mdpi.com/1099-4300/24/4/550

3. https://researchrepository.murdoch.edu.au/id/eprint/53710/

4. https://www.sciencedirect.com/science/article/pii/S0004370297000738

5. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=00307ba02e3fa2e23eac5e3c35dabeb054054fe3