Department of Computer Science and Engineering

Course Code: CSE-3636
Course Title: Artificial Intelligence Lab
Project Title: AI based Chess Game using Python

Project Documentation

Week 9: Random Move AI, MinMax Algorithm

Submitted by

Lamisa Mashiat (C201249)

Sohana Tasneem (C201266)

Sadia Haque Chowdhury (C201270)

Submitted to

Subrina Akter
Assistant Professor
Dept. of CSE

13th May, 2023

In this updated version of the code, the chess game has been enhanced with an AI script called **ChessAI.py**, which can generate random moves as an opponent. The findRandomMove() function in the ChessAI.py script selects a random valid move from a given list of valid moves. The **ChessMain.py** script has also been updated to include an AI player option. If the player chooses to play against the AI, then the AI's move is automatically generated using the findRandomMove() function. The humanTurn variable is used to determine whose turn it is, and if it's not the human's turn and the game is not over, then an AI move is generated and executed. Further we improved our Chess AI with implementing MinMax Algorithm so that our chess engine performs more efficiently.

ChessAI.py Script

The ChessAI.py script contains one function called **findRandomMove**(validMoves). This function takes a list of valid moves as input and returns a random move from the list.

The updated ChessAI.py script contains a more efficient algorithm for finding the best move for the AI player. It implements the MinMax algorithm with alpha-beta pruning, which is a common approach for solving two-player games like chess.

The new **findBestMoveMinMaxNoRecursion()** function is an implementation of the MinMax algorithm without recursion. It calculates the best move by considering all valid moves and their corresponding opponent responses up to a certain depth. The function uses a **scoreMaterial()** helper function to evaluate the board position after each move.

The **findBestMove()** function is a helper function for the recursive version of the MinMax algorithm. It calls the **findMoveMinMax()** function with the given parameters, which recursively evaluates the board positions up to a certain depth and returns a score for each move.

The **scoreBoard()** function evaluates the current board position and returns a score based on the relative value of the pieces on the board. Positive scores indicate an advantage for white, while negative scores indicate an advantage for black.

ChessMain.py Script

The ChessMain.py script has been updated to include the AI player functionality. The updated code includes the following changes:

1. A new variable called playerTwo has been added to the code. This variable determines whether the black pieces are controlled by a human player or the AI. If the variable is set to True, the black pieces will be controlled by a human player. If it is set to False, the black pieces will be controlled by the AI.

2. The while loop that controls the game has been updated to include an if statement that checks if it is the AI player's turn to move. If it is, the findRandomMove() function from the ChessAI.py script is called to generate a random move for the AI player. The move is then made using the makeMove() function from the GameState.py script.
3. A new variable called animate has been added to the code. This variable controls whether the move animation is played after a move is made. The animate variable is set to True when a move is made by either the human or the AI player.
4. The updated ChessMain.py script uses the new findBestMove() function to determine the AI player's move. This makes the AI player much more efficient and capable of making more strategic moves.
5. The **DEPTH** constant can be adjusted to control the depth of the MinMax search, which affects the AI player's level of strategic thinking. A higher depth can lead to a stronger AI player, but it also increases the computation time required for each move.

Conclusion

The MinMax algorithm with alpha-beta pruning is a powerful approach for solving two-player games like chess. The updated ChessAI module implements this algorithm to make the AI player more efficient and strategic. The **DEPTH** constant can be adjusted to control the level of difficulty for the AI player. With these updates, the chess game is now more challenging and enjoyable for players of all skill levels.