Bitaniya Nassir Id GSR/5394/16 AI principle assignment 2

1.Implement the minimax algorithm for the game of tic-tac-toe.

2. Extend the minimax algorithm to handle heuristic evaluation functions.

1. Introduction:

a.Provide an overview of the lab assignment.

The purpose of this lab report is to provide an understanding of the minimax algorithm in the tic tac toe game. The goal of this lab experiment is to develop an AI player that can perform optimally in the game by applying the minimax algorithm. The algorithm will be expanded to accommodate heuristic evaluation functions. We will test the algorithm with alpha-beta pruning.

Min-max is an algorithm for making decisions that determines the best course of action by utilising statistics, game theory, decision theory, and philosophy. This game is for two players. The minimum loss and maximum profit are assessed by the mechanism.

Minimax algorithm for tic tac toe we have two player which is player\_X and player\_O, in lab I use player\_X as MAX and player\_O as MIN , means that player\_X is human and player\_O is computer(AI)

player \_X is the opponent of player\_O assigned as MAX means that player\_X select the maximised value and player\_O select the minimised value.

Minimax algorithm performs depth first search algorithm for exploration of the complete game tree.

b. State the objectives and goals of the experiment.

The main objective is implementing a minimax algorithm to identify the best next move to win the tic tac toe game. And as an assignment understanding how minimax algorithms work.

The challenges that i find during implementing the minimax algorithm for tic tac toe

1. Problem Statement:

* Creating board to implement the algorithm and play with ai

Identifying the terminal node . I didn't understand where the terminal nodes were starting in a case of tic tac toe. As our implementation is a minimax algorithm it is a kind of backtracking algorithm so I should have known where and how to implement the terminal node.

* To extend the minimax algorithm to handle heuristic evaluation functions, I was

Background

Tic tac toe is a popular and well known game. it is one of the paper-and-pencil games.This game requires two players in a 3x3 grid with Player 1 acting as “O” and Player 2 acting as “X”, or vice versa

The user has the option to fill these nine spaces with noughts (O) or crosses (X). To win, the player must arrange three identical markers in a row that are either horizontal, vertical, or diagonal.

The two players play optimally, the one maximises the probability of his winning and the second player plays to minimise the probability of the first player .

Methodology

As we know that the minimax algorithm applies the depth first search with backtracking. The minimax algorithm is a recursive algorithm that considers all possible moves and evaluates the game state at each level of the search tree. Both players are play optimally and tries to find the best move for themselves and I assume the one player as AI player as maximizer and human as minimizer

The algorithm starts by evaluating the current game state, if there is a winner displaying the name of the winner or if the game board is full of “X” and “O” it displays the game is drawn.

The minimax algorithm assumes that both players play optimally, so it recursively explores all possible moves until it reaches a terminal state (win, loss, or draw). It assigns scores to each terminal state based on the outcome of the game. The maximizer player tries to maximise the score, while the minimizer player tries to minimise the score.

The minimax algorithm is described.

1) The game board's current state is fed into the algorithm.

2) It first determines whether the game has reached a conclusion, which could be a tie, win, or loss. The algorithm returns a score based on the game's result if it's a terminal state.

3) The algorithm calls itself recursively for every move the current player is capable of making if it's not in a terminal state.

4) The algorithm predicts that the player in the lead will make every move that is feasible, and that the other player will counter with the best move available.

5) After that, assuming that players take turns, the algorithm calls itself recursively to determine the score of every possible move.

6) The algorithm selects the move with the highest score if the player at that moment is maximising their score. The algorithm selects the move with the lowest score if the current player—that is, the opponent—is minimising their score.

7) The algorithm gives the player's move the highest possible score.

Minimax algorithm with heuristics included:

1) Identify the players who maximise and minimise.

2) Put into practice a heuristic evaluation function that gives each game state a numerical value. For the current player, this evaluation function ought to yield an approximation of how desirable the current state is.

3) Add the current player and the tree's depth as parameters to the recursive minimax function.

4) Return the heuristic evaluation of the current game state in the minimax function if the depth is 0 or a terminal game state is reached.

4. Initialise the best value to negative infinity if you are the maximizer. Recursively call the minimax function with the depth reduced by 1 for each potential move, switching the player to the minimizer. If a higher value is discovered, update the optimal value.

5. Initialise the best value to positive infinity if the minimizer is taking the turn. Recursively call the minimax function with the depth reduced by 1 for each potential move, switching the player to the maximizer. If a lower value is discovered, update the optimal value.

6. Give back the best value discovered.

· Board Representation:

- The Tic Tac Toe board is typically represented as an array or list of cells. In the provided implementation, the board is represented as a list of length 9, where each index corresponds to a position on the board. The value of each index represents the current state of that cell, with '-' representing an empty cell, 'X' representing a move by the AI player, and 'O' representing a move by the human player.

· Evaluation Function:

- The evaluation function (`evaluate`) is a crucial part of the minimax algorithm. It determines the score of the current board state. In Tic Tac Toe, the evaluation function checks for a win, loss, or draw situation.

· Implement the minimax algorithm with alpha-beta pruning:

Board Representation:

o Create a suitable data structure to represent the chessboard and store the current game state, including the positions of the pieces, castling rights, en passant squares, and other relevant information.

Move Generation:

o Develop a move generator that generates all possible legal moves for the current player. This includes generating pseudo-legal moves, accounting for piece movement rules, captures, promotions, castling, and en passant captures.

Evaluation Function:

o Implement an evaluation function that assigns a numerical value to a given chess position. The evaluation function should consider various factors such as material balance, piece activity, pawn structure, king safety, development, and other positional considerations.

Minimax Algorithm:

o Implement the basic minimax algorithm with alpha-beta pruning. The algorithm recursively searches the game tree, alternating between maximizing and minimizing players.

o At each level of the tree, maintain alpha and beta values, which represent the best known maximum and minimum scores, respectively, for the maximizing and minimizing players.

o During the search, if the current node's alpha value becomes greater than or equal to the beta value, pruning occurs, and the algorithm stops exploring further branches, as the opponent would never choose this path.

Transposition Tables:

o Utilize transposition tables to store previously evaluated board positions along with their associated scores. This helps avoid redundant evaluations and improves the efficiency of the search.

Iterative Deepening:

o Implement iterative deepening, which involves performing multiple depth-limited searches with increasing depth. This technique allows the engine to return the best move found so far within a given time constraint.

Quiescence Search:

o Incorporate quiescence search to handle positions where the game is quiet and stable. Quiescence search extends the search beyond terminal nodes and evaluates capturing moves, checks, and other critical moves to avoid the horizon effect.

Advanced Techniques:

o Consider implementing additional advanced techniques, such as move ordering heuristics, killer move heuristics, history heuristics, and null move pruning, to further improve the search efficiency.

Result and conclusion

In my case I assign the grid as index of the array starting from the number 0 and ending with number 8(0-8) the winning row numbers are [0, 1, 2], [3, 4, 5], [6, 7, 8], the winning column numbers are [0, 3, 6], [1, 4, 7], [2, 5, 8] and the diagonal numbers are [0, 4, 8],

[2, 4, 6] if player\_O or player\_X draw either of these numbers .

References

1. Garg, R., & Nayak, D. P. (2017). GAME OF TIC-TAC-TOE: SIMULATION USING MIN-MAX ALGORITHM. *International Journal of Advanced Research in Computer Science*, *8*(7), 1074–1077. <https://doi.org/10.26483/ijarcs.v8i7.4409>