BACH KHOA UNIVERSITY

COMPUTER SCIENCE AND COMPUTER ENGINEERING

Mini-Project

Alpha-beta algorithm for the game: Tic-tac-toe

Lecturer: Duong Tuan Anh

Student: Le Hoang Hiep – 1450222
Tran Duc Trng - 1552402

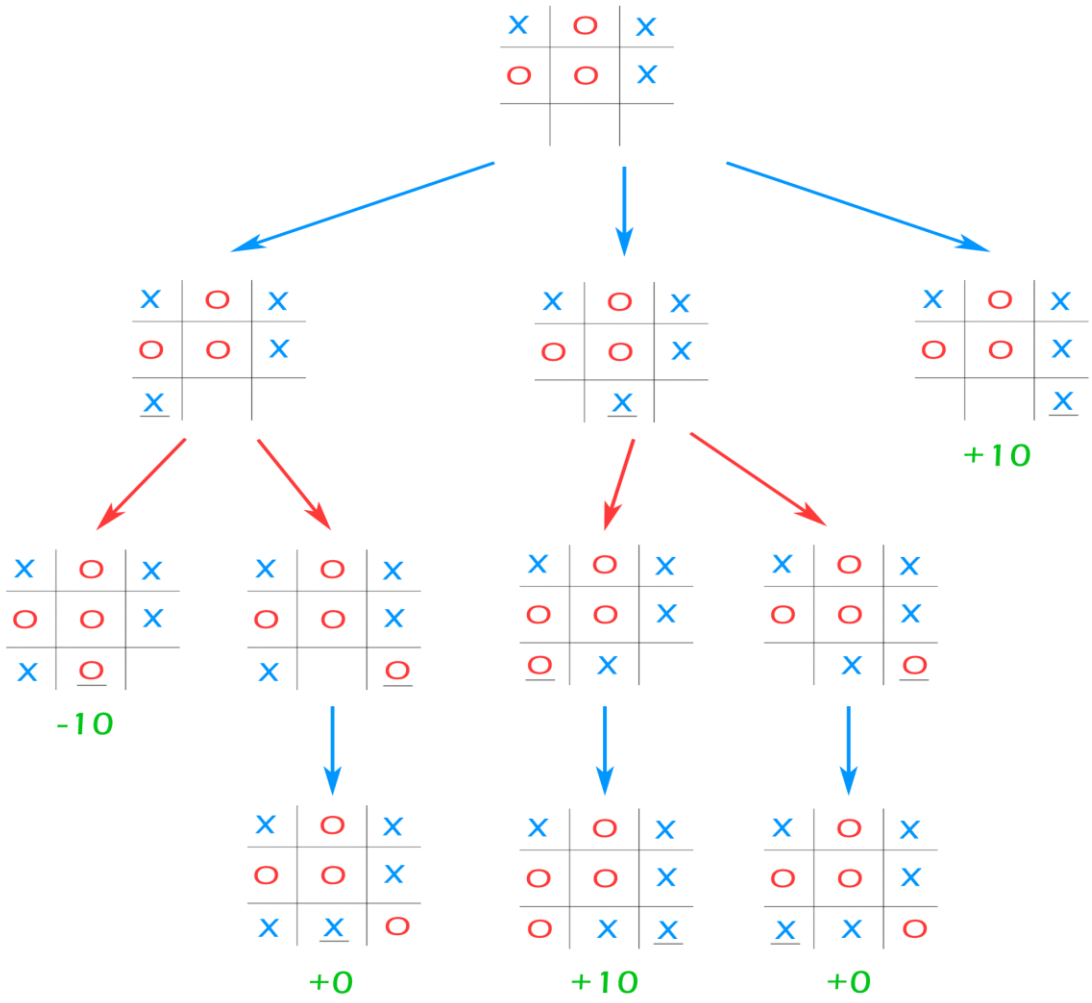
Topic 11: Alpha-beta algorithm for the game: Tic-tac-toe

List of Figures

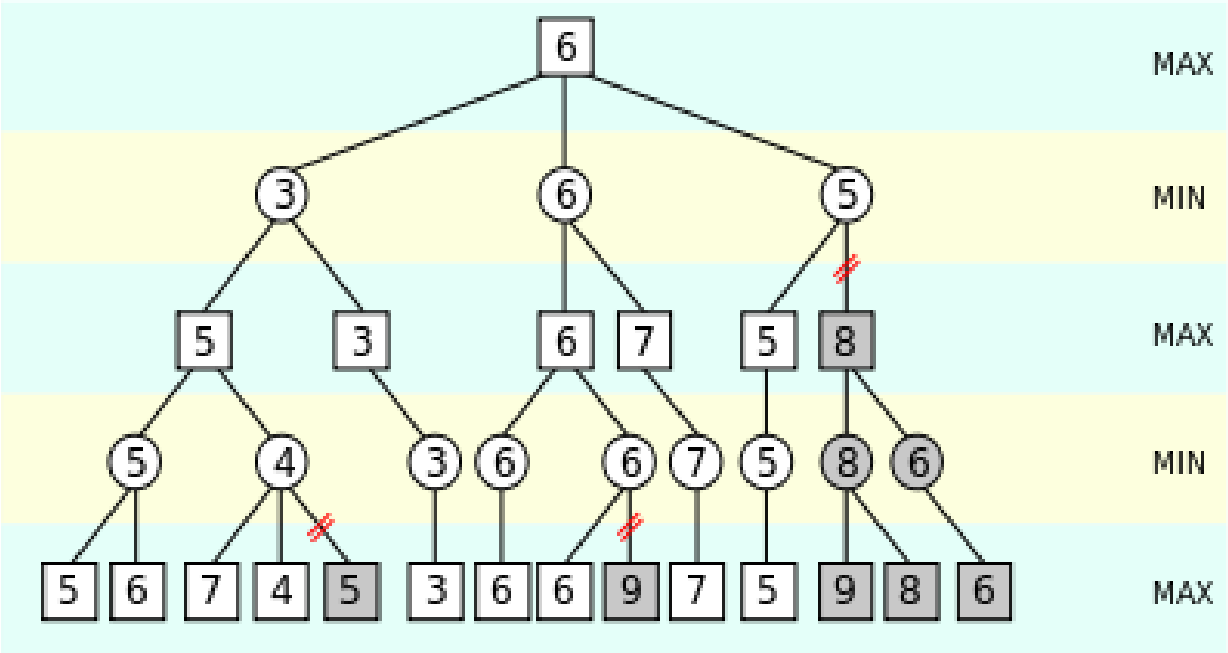
1. Theory:

A **minimax** algorithm is a recursive algorithm for choosing the next move in an n-player game, usually a two-player, back and forth game. A value is associated with each position or state of the game. This value is computed by means of a position evaluation function and it indicates how good it would be for a player to reach that position. The player then makes the move that maximizes the minimum value of the position resulting from the opponent's possible following moves. If it is A's turn to move, A gives a value to each of his legal moves.

A simple version of the minimax algorithm, stated below, deals with games such as tic-tac-toe, where each player can win, lose, or draw. If player A can win in one move, his best move is that winning move. If player B knows that one move will lead to the situation where player A can win in one move, while another move will lead to the situation where player A can, at best, draw, then player B's best move is the one leading to a draw. Late in the game, it's easy to see what the "best" move is. The Minimax algorithm helps find the best move, by working backwards from the end of the game. At each step it assumes that player A is trying to maximize the chances of A winning, while on the next turn player B is trying to minimize the chances of A winning (i.e., to maximize B's own chances of winning).



Alpha–beta pruning is a search algorithm that seeks to decrease the number of nodes that are evaluated by the minimax algorithm in its search tree. This allows us to search much faster and even go into deeper levels in the game tree. It cuts off branches in the game tree which need not be searched because there already exists a better move available. The algorithm maintains two values, alpha and beta, which represent the maximum score that the maximizing player is assured of and the minimum score that the minimizing player is assured of respectively. Initially alpha is negative infinity and beta is positive infinity, i.e. both players start with their lowest possible score. It can happen that when choosing a certain branch of a certain node the minimum score that the minimizing player is assured of becomes less than the maximum score that the maximizing player is assured of (beta <= alpha). If this is the case, the parent node should not choose this node, because it will make the score for the parent node worse. Therefore, the other branches of the node do not have to be explored.

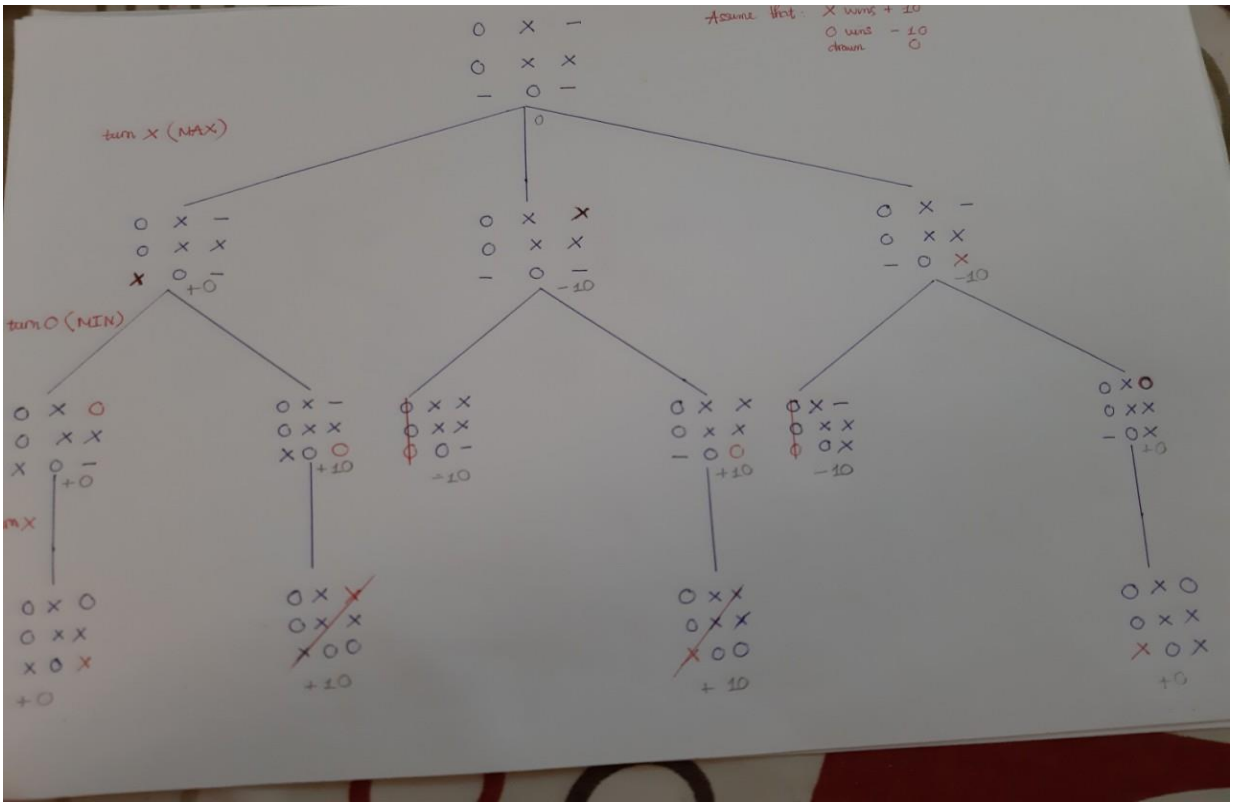


2. Pseudocode:

```
function minimax(node, depth, isMaximizingPlayer, alpha, beta):

    if node is a leaf node :
        return value of the node

    if isMaximizingPlayer :
        bestVal = -INFINITY
        for each child node :
            value = minimax(node, depth+1, false, alpha, beta)
            bestVal = max( bestVal, value)
            alpha = max( alpha, bestVal)
            if beta <= alpha:
                break
        return bestVal

    else :
        bestVal = +INFINITY
        for each child node :
            value = minimax(node, depth+1, true, alpha, beta)
            bestVal = min( bestVal, value)
            beta = min( beta, bestVal)
            if beta <= alpha:
                break
        return bestVal

// Calling the function for the first time.
minimax(0, 0, true, -INFINITY, +INFINITY)
```

3. Demonstration:

- For example, we get this board position in the game:



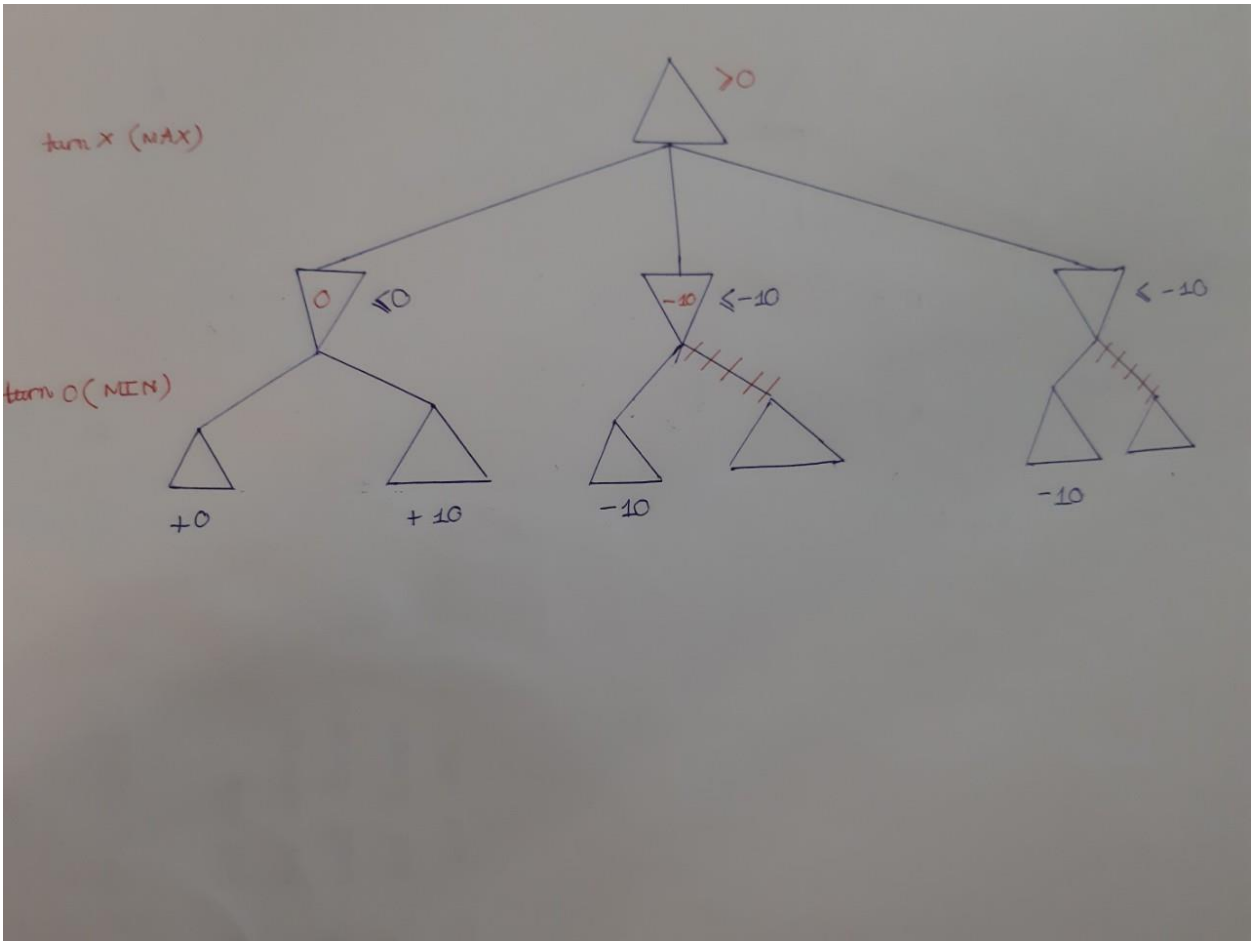
- We run in program with the first situation: (turn X) and the program reply as in we expected by putting in the position that lower our chance of winning:

```
O | X | -
----------
O | X | X
----------
- | O | -

Row pl2 :
Col play: 0


O | X | O
----------
O | X | X
----------
X | O | -

Row play: 2
Col play: 2
```

- Finally, the game is draw:

```
O | X | -
----------
O | X | X
----------
- | O | -

Row pl2 :
Col play: 0


O | X | O
----------
O | X | X
----------
X | O | -

Row play: 2
Col play: 2


O | X | O
----------
O | X | X
----------
X | O | X

********** GAME OVER **********

PLAYER DRAW
```

Alpha-Beta Pruning:


turn X (MAX)
turn O (MIN)
>0
0
≤0
-10
≤-10
< -10
+0
+10
-10
-10