BITS Pilani, Dubai Campus

Dubai International Academic City (DIAC)

Dubai, U.A.E

Assignment for Artificial Intelligence (CS F407)

On

Connect 4 Game using Minimax Algorithm

Submitted to

Dr. Sujala D. Shetty

By

Aafia Iqbal **2018A7PS0061U**

Mardiyah Khadijah **2018A7PS0257U**

TABLE OF CONTENTS

Abstract

The classic game of Connect 4 is fading from the face of society, especially with the younger generations due to the addictive instance of digital games, available anywhere, hence measures to preserve the continuity of this game lies in the recreation of the game in the virtual environment. The formulation of classic Connect 4 game utilizing the implementation of Alpha-Beta Pruning with Minimax algorithm is carried out with the objective of modifying its parameters to determine its influence on the execution of the game. The parameters involved in the modification process includes the depth of search and size of board as a small change of constant can lead to a drastic difference. The Minimax algorithm serves the purpose of enabling the computer (AI) to place its piece strategically whereas Alpha-Beta Pruning is incorporated to reduce the size of its search tree. All results pertaining to the changes made have been recorded accordingly and the optimal constant of parameters are identified, leading to an ideal execution of the game.

Chapter 1 : Problem Statement

The game is categorized as a zero-sum game. Therefore, the minimax algorithm, which is a decision rule used in AI, can be applied. The project goal is to investigate how a decision tree is applied using the minimax algorithm in this game by Artificial Intelligence.

Chapter 2 : Introduction

With the constant evolution of technology, it comes as no surprise that what seemed like a brilliant source of entertainment back then is slowly fading into the background with a shift towards digital games. The emergence of computer and mobile games, offering a variety of gaming genres have taken over the need for physical board games such as Monopoly, UNO, Connect-4, Scrabbles, etc. Though some games managed to stay relevant in this day and age, namely the ones with multiple players option, others faded to gray. Classic physical games such as Connect 4 and Chess are slowly losing their place in the society, hence it is necessary to computerize the core elements of the games to recreate them in a virtual environment. As such, this would serve as an effort to preserve the continuity of these games in the future and hopefully resuscitate the fun and joy it once brought to the society.

Connect 4 is a centuries-old game even played by Captain James Cook with his officers on his long voyages. Milton Bradley (now owned by Hasbro) published a version of this game called "Connect Four" in 1974. It is also called "Four-in-a-Row" and "Plot Four." Two players play this game on an upright board with six rows and seven empty holes. Each player has an equal number of pieces (21) initially to drop one at a time from the top of the board. Then, they will take turns to play and whoever makes a straight line either vertically, horizontally, or diagonally wins. Due to the nature of gravity, the pieces will always fill in the baseline of the board, eventually building up to all seven rows.

Chapter 3 : Logic and Mathematics behind Connect Four

Decision Tree: A Decision tree is a tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label. In the example below, one possible flow is as follows: If a person has aged less than 30 and does not eat many pizzas, then that person is categorized as fit. On the contrary, if a person is older than 30, and does not exercise in the morning, then that person is categorized as unfit. Decision trees can be applied in different studies, including business strategic plans, mathematics studies, and others. In addition, since the decision tree shows all the possible choices, it can be used in logic games like Connect Four to serve as a look-up table.

Decision tree in Connect Four: When the game begins, the first player gets to choose one column among seven to place the colored disc. There are 7 columns in total, so there are 7 branches of a decision tree each time. After the first player makes a move, the second player could choose one column out of seven, continuing from the first player's choice of the decision tree.

Notice that the decision tree continues with some special cases. First, if both players choose the same column 6 times in total, that column is no longer available for either player. It means that their branches of choice are reduced by one. Second, when both players make all choices (42 in this case) and there are still no 4 discs in a row, the game ends as a draw, and the decision tree stops. Finally, if any player makes 4 in a row, the decision tree stops, and the game ends.

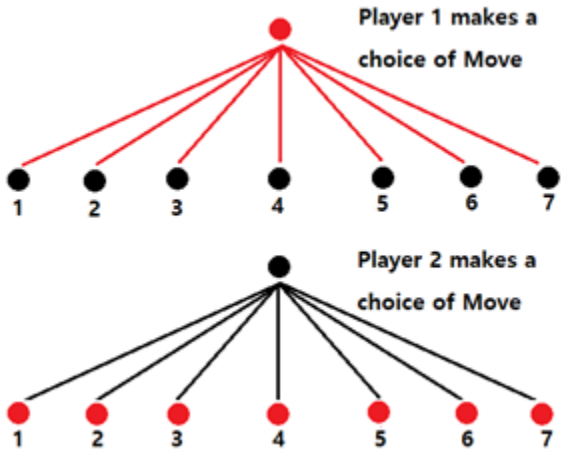

Figure: Possible moves for each iteration of the Connect Four game shown in the decision tree

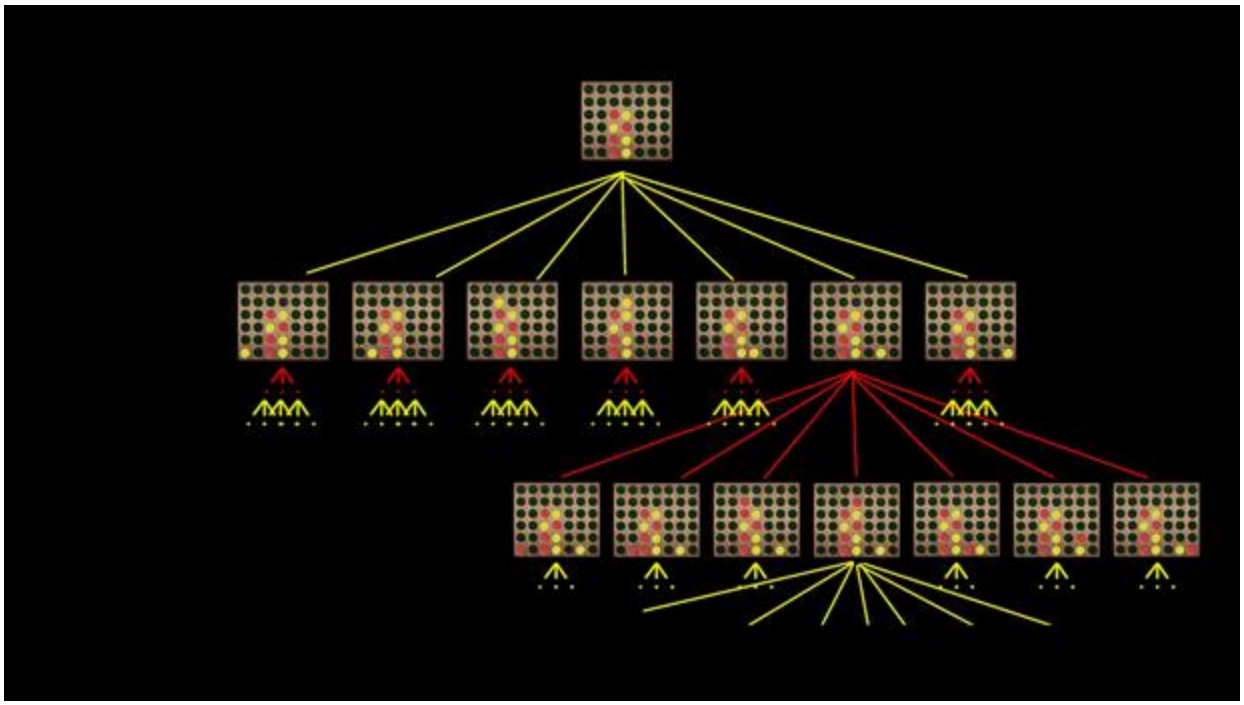

Figure: Decision tree of Connect Four possible moves

Minimax algorithm: Minimax algorithm is a recursive algorithm which is used in decision-making and game theory especially in AI games. It provides optimal moves for the player, assuming that the opponent is also playing optimally. For example, considering two opponents: Max and Min playing. Max will try to maximize the value, while Min will choose whatever value is the minimum. The algorithm performs a depth-first search (DFS) which means it will explore the complete game tree as deep as possible, all the way down to the leaf nodes. The algorithm is shown below with an illustrative example.

```
function minimax(node, depth, maximizingPlayer) is
    if depth = 0 or node is a terminal node then
        return the heuristic value of node
    if maximizingPlayer then
        value := −∞
        for each child of node do
            value := max(value, minimax(child, depth − 1, FALSE))
        return value
    else (* minimizing player *)
        value := +∞
        for each child of node do
            value := min(value, minimax(child, depth − 1, TRUE))
        return value
```

Figure: Minimax Algorithm Pseudocode

Initially, the algorithm generates the entire game tree and produces the utility values for the terminal states by applying the utility function. For example, in the below tree diagram, let us take A as the tree's initial state. Suppose the maximizer takes the first turn, which has a worst-case initial value that equals negative infinity. Then, the minimizer will take the next turn, which has a worst-case initial value that equals positive infinity.

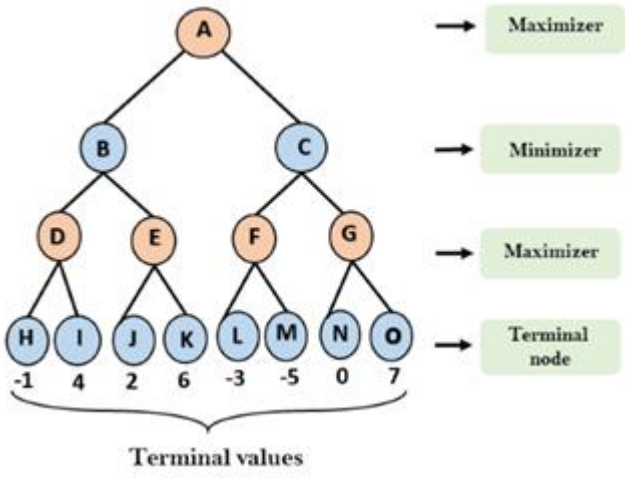

Figure: Minimax Algorithm in tree format — The Initial Step

First, we consider the Maximizer with initial value = −∞. Each terminal node will be compared with the value of the maximizer and finally store the maximum value in each maximizer node. Take the third row (Maximizer) from the top, for instance.

For node D max(−1, −∞) → max(−1,4) = 4

For Node E max(2, −∞) → max(2, 6) = 6

For Node F max(−3, −∞) → max(−3,−5) = −3

For node G max(0, −∞) → max(0, 7) = 7

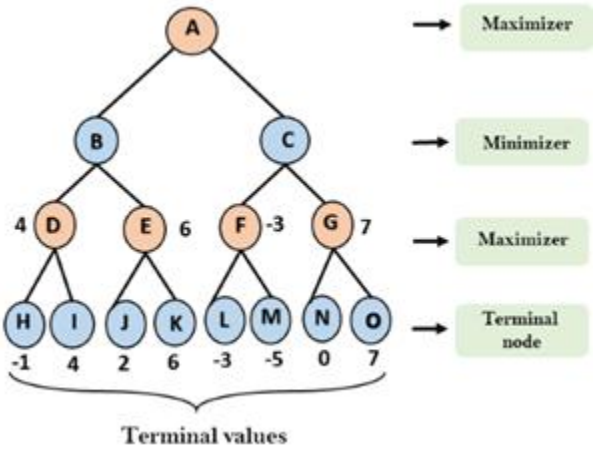

Figure: Minimax Algorithm in tree format — Second Step

Next, we compare the values from each node with the value of the minimizer, which is +∞.

toFor node B= min(4, 6) = 4

For node C= min(-3, 7) = -3

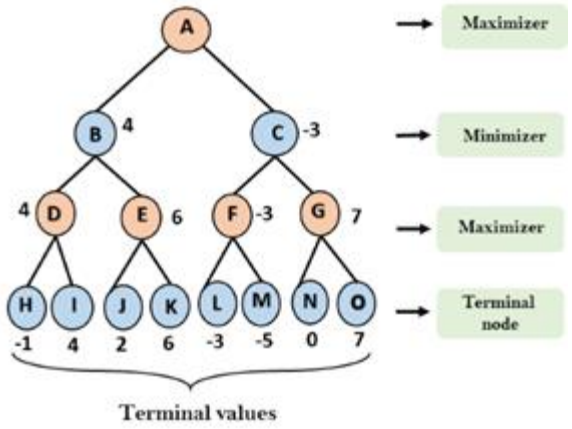

Figure: Minimax Algorithm in tree format — Third Step

Finally, the maximizer will then again choose the maximum value between node B and node C, which is 4 in this case. Hence, we get the optimal path of play: A → B → D → I.

For node A max(4, -3)= 4

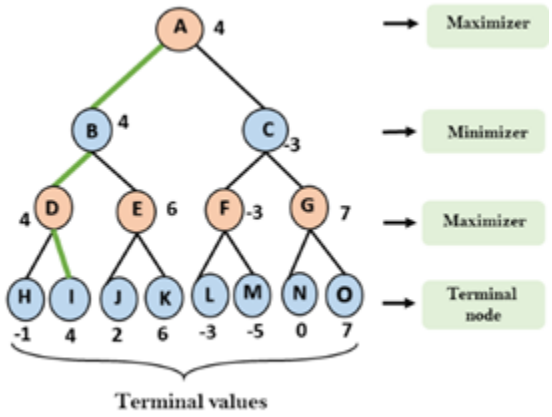

Figure: Minimax Algorithm in tree format — The Final Step

Chapter 4 : AI in Connect4 — Implementing Minimax

Below is a python snippet of Minimax algorithm implementation in Connect Four. In the code, we extend the original Minimax algorithm by adding the Alpha-beta pruning strategy to improve the computational speed and save memory. The figure below is a pseudocode for the alpha-beta minimax algorithm.

```
function alphabeta(node, depth, α, β, maximizingPlayer) is
    if depth = 0 or node is a terminal node then
        return the heuristic value of node
    if maximizingPlayer then
        value := −∞
        for each child of node do
            value := max(value, alphabeta(child, depth − 1, α, β, FALSE))
            α := max(α, value)
            if α ≥ β then
                break (* β cut-off *)
        return value
    else
        value := +∞
        for each child of node do
            value := min(value, alphabeta(child, depth − 1, α, β, TRUE))
            β := min(β, value)
            if β ≤ α then
                break (* α cut-off *)
        return value
```

Figure: Minimax Algorithm with Alpha-Beta Pruning Pseudocode

Explanation of Code:

Let us take the maximizingPlayer from the code. First, the program will look at all valid locations from each column, recursively get the new score calculated in the look-up table, and finally update the optimal value from the child nodes. The alpha here in this section is the new_score, and when it is greater than the current value, it will stop performing the recursion and update the new value to save time and memory.

As mentioned above, the look-up table is calculated according to the evaluate_window function. Here, the window size is set to four since we are looking for connections of four discs. Considering a reward and punishment scheme in this game. If four discs are connected, it is rewarded for a high positive score (100 in this case). When three pieces are connected, it has a

score less than the case when four discs are connected. When two pieces are connected, it gets a lower score than the case of three discs connected. Finally, when the opponent has three pieces connected, the player will get a punishment by receiving a negative score. Indicating that it is not an optimal move for the current player.

With the scoring criteria set, the program now needs to calculate all scores for each possible move for each player during the play. The function score_position performs this part from the below code snippet. The AI player will then take advantage of this function to predict an optimal move.

Chapter 5 : Instructions to Play

- Players take turns dropping their game piece onto the grid by picking a column number 1-7 (inclusive).
- The game pieces fall to the lowest position available on the grid for the column they selected.
- The first player to reach 4 pieces aligned in a row, wins. Rows can be any direction: horizontal, vertical, diagonal up, diagonal down.

Chapter 6 : Results

These are the results of three consecutive games played on the python programmed Connect 4:

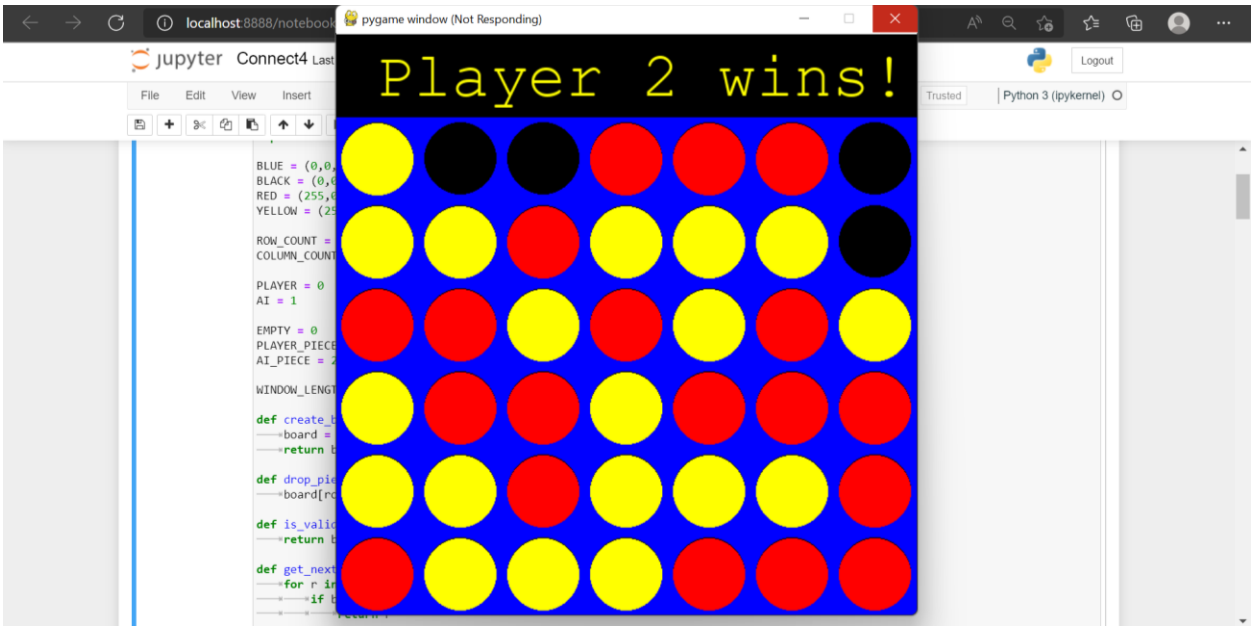

Figure: Result from the 1st game played

Figure: Result from the 2nd game played

Figure: Result from the 3rd game played

Chapter 7 : Conclusion

Looking at how many times AI has beaten human players in this game, I realized that it wins by rationality and loads of information. In this project, the AI player uses a minimax algorithm to check for optimal moves in advance to outperform human players by knowing all possible moves rationally. Interestingly, when tuning the number of depths at the minimax function from high (6 for example) to low (2 for example), the AI player may perform worse. Nevertheless, the strategy and algorithm applied in this project have been proved to be working and performing amazing results.

References

[1] Nasa, R., Didwania, R., Maji, S., & Kumar, V. (2018). Alpha-beta pruning in mini-max algorithm–an optimized approach for a connect-4 game. Int. Res. J. Eng. Technol, 1637–1641.

[2] "Artificial Intelligence At Play-Connect Four (Minimax Algorithm Explained)". Medium, 2022,https://medium.com/analytics-vidhya/artificial-intelligence-at-play-connect-four-minimax-algorithm-explained-3b5fc32e4a4f.

[3] V. K. BC, N. Jashank, and M. S. Nadiger, "Alpha-Beta Pruning–A streamline approach for perceptive game playing," International Research Journal of Modernization in Engineering Technology and Science, 2(6), pp. 1306–1318, June 2020.

[4] E. Alderton, E. Wopat, and J. Koffman, "Reinforcement learning for Connect Four," January 2019.

[5] Galli. (n.d.). KeithGalli/Connect4-Python. GitHub https://github.com/KeithGalli/Connect4-Python

[6] S. P. Singhal, and M. Sridevi, "Comparative study of performance of parallel Alpha Beta Pruning for different architectures," In 2019 IEEE 9th International Conference on Advanced Computing (IACC), pp. 115–119, December 2019.

[7] L. Tommy, M. Hardjianto, and N. Agani, "The analysis of Alpha Beta Pruning and MTD(f) algorithm to determine the best algorithm to be implemented at Connect Four prototype," In IOP Conference Series: Materials Science and Engineering, vol. 190, 2017.

[8] R. Nasa, R. Didwania, S. Maji, and V. Kumar, "Alpha-Beta Pruning in Mini-Max algorithm – An optimized approach for a Connect-4 game," International Research Journal of Engineering and Technology, 5(4), pp. 1637–1641, April 2018.

[9] S. Mandadi, B. Tejashwini, and S. Vijayakumar, "Implementation of sequential and parallel Alpha-Beta Pruning algorithm," International Journal of Innovations in Engineering Research and Technology, 7(8), pp. 98–104, August 2020.

[10] J. Hernandez, K. Daza, and H. Florez, "Alpha-Beta vs Scout algorithms for the Othello game," In CEUR Workshops Proceedings, vol. 2846, pp. 65–79, 2019.