PROJECT REPORT
ON ALPHA-BETA PRUNING

Submitted By

JAGADEESH PRADHAN - Reg no 2241016398
PRATYUS PRADHAN - Reg no 2241016410
PRATYUSH PRITAM - Reg no 2241016462
PRIYADARSHINI MADHUBALA PRADHAN - Reg no 2241016413
SIVA KUMAR PRADHAN - Reg no 2241016427
PRIYANSHU PRADHAN - Reg no 2241016415

Supervised
By

Mr. Shiva Agarwal
Assistant Professor
Computer Science and Engineering
Faculty of Engineering and Technology

Centre For Artificial Intelligence and Machine Learning
Institute of Technical Education And Research(ITER)
SIKSHA ‘O’ ANUSANDHAN (DEEMED TO BE) UNIVERSITY
Bhubaneswar-751030, Odisha, India

Shiva Agarwal
(Supervisor)

Contents

1 Abstract

The classic game of Connect-4 has disappeared from people's sight, especially for young people, due to the ubiquity of digital gaming events, so measures have been taken to control the continuation of this game in the virtual environment. In this paper, the classic game Connect-4 is designed using Alpha-Beta pruning with minmax algorithm to change its parameters to determine its impact on the game.There is no interference in the change process, including deep search and large dashboard, because small changes can make a big difference. While the purpose of the Minimax algorithm is to enable the computer (AI) to place its objects, Alpha-Beta pruning is used to reduce the size of the search tree. All results of the changes are recorded accordingly and a permanent consensus is determined, ensuring successful completion of the game.
Keywords - Artificial Intelligence (AI), Alpha-Beta pruning, Minimax, Connect-4, Optimization

2 Introduction

As technology continues to advance, it is not surprising that what seemed like the brightest entertainment at the time is gradually fading into the background as digital games change. The emergence of computer and mobile games has changed the need for physical games by offering different types of games such as Monopoly, UNO, Connect-4, Scrabble and more. Although some games are still relevant today for people with lots of options, other options are fading into grey. Classic physical games such as Connect-4 and chess have gradually lost their place in society, so the main elements of the game need to be computerized so that they can be created in a virtual environment. So in the future it will be about maintaining the continuity of the game and regaining the fun and joy it once brought to the community.

Connect-4 is a classic two-player board game in which each piece is represented by a yellow and red piece on a board matrix of seven rows and six rows. Players have the freedom to modify their products within the available space by placing them in any compartments they wish, limiting them to six rows. Due to gravity, the pieces will always fill the bottom of the board and form all seven rows. The player's goal is to complete four consecutive pieces horizontally or diagonally, so that when the task is completed the game can end in a win or a draw if there is no one who can find consecutive letters. Link-4 is a game-solving game where players who know the key moves are guaranteed a 100 percentage win rate regardless of where their opponents move, as long as they play according to the winning strategy [2].

In this article, we will examine the changes that affect the time required for the placement decision, the number of lines checked, and the gains in the Alpha-Beta pruning algorithm with Minimax. Minimax is used for the ability to perform a decision process, while Alpha-Beta pruning is used to reduce the size of the search tree in unnecessary searches [2]. Both algorithms respectively play an important role in ensuring the success of the experiment.

The following document is divided into several sections and subsections; The first shows the research topics, the general content of the articles, and a summary of the data analysis. On the other hand, Section 3 focuses on the Literature survey Section 4 methodology. Section 5 focuses the results obtained and their discussion, describes our findings for each variable, and finally Section 6 presents the conclusion of the entire paper.

3 Literature Survey

3.1 Minmax Algorithm

The Minmax algorithm [1,2] is a simple iterative algorithm that determines the next state by analyzing different states in the game tree.

The maximizing player tries to get the point of the next situation, while the minimizing player tries to get the point of the next situation. Try to lower the score of the next situation. However, since the Minmax algorithm cannot search the entire game tree to find the next best state, it uses depth-first search to determine the next state [23].

The working principle of the Minmax algorithm is to create an upward game tree containing the current state as the root node to achieve depth. We then use the index to create a score for each page. Depending on the type of parent node (maximizing or minimizing), we select the value of the page node based on the value of the parent node. Therefore, if the parent node is a maximizing player, it always chooses the highest score state of its children [10, 22]. [7] [6] In this the process is repeated until we reach our root node at depth 0, and we can determine the optimal path that maximizes (or minimizes) our state value. Considering the example in Fig. 1, where each node is the state is the game of chess. A white state represents a maximizing player and a black state represents a minimizing

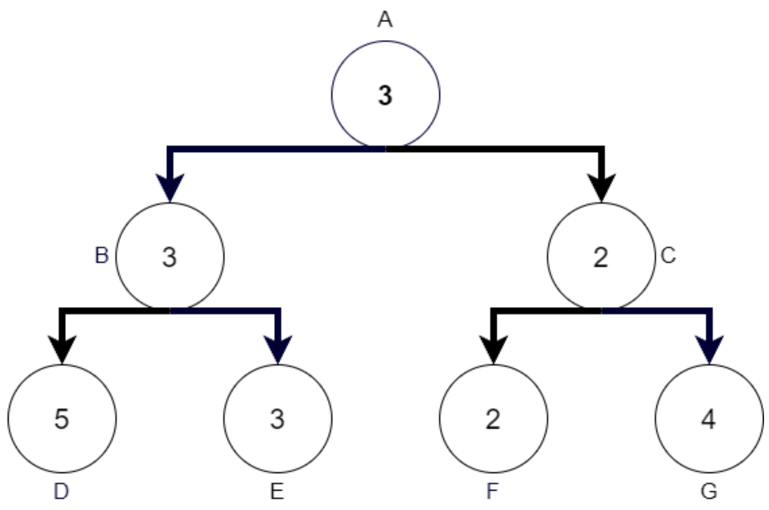


Figure 1: Ex. Game Tree

player. The leaf nodes are D, E, F, and G which have been assigned score values according to the evaluation function.The current state is the root node A [12,13].

Now, considering state B and its child nodes D and E, node B is a minimizing node and hence it will be assigned the minimum value from its child nodes. Hence, node B will be assigned the value 3. Similarly, for the nodes, C, F, and G, node C will be assigned the value 2. Node A is a maximizing node and hence it will select the maximum score value from its child nodes. Hence, node A will be assigned the value 3 [18].

Looking at the above example we can see that the next best state from the current state A is node B. We can assume that the opposing player plays optimally and selects node E. as shown in figure 1 we can draw game tree.

3.2 Alpha-beta pruning algorithm

Considering the example in Fig. 2, we can see that node F has a score value of 2. We know that the parent node C is a minimizing node and hence its score value will be at most equal to 2 (from node F) [11,20]. Now considering the left subtree (node B, D, and E), the value of node B will be equal to 3. We can hence determine that node A being a maximizing node will select node B (value 3) over C (at most 2). Thus, we would never have to calculate the score value of node G, since its parent node will always select the minimum value, which can at most be 2[21][1]

This method of reducing the search space in the game tree is known as alpha-beta pruning [1,2,3], where alpha and beta are the upper and

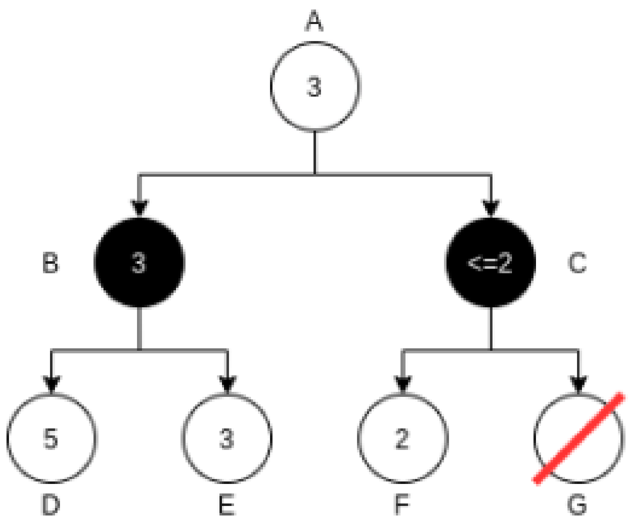


Figure 2: Game tree with Alpha-Beta pruning

lower bounds. With alpha-beta pruning, we only have to explore the most optimal nodes which can lead to a win (considering the opposing player plays optimally) and we can stop exploring nodes that do not change the score of their parent nodes.[5].

alpha and beta are the best scores that either player can achieve where alpha corresponds to the maximizing player and beta corresponds to the minimizing player. Initially, alpha is assigned the minimum possible score for the maximizing player which in minus infinite and beta is assigned the maximum possible score for the minimizing player which is positive infinite .

4 Methodology

4.1 ALGORITHMS

In the context of games, this section discusses Minimax and Alpha-Beta pruning search algorithms. A game tree containing all the possible moves a player can make forms the basis of all algorithms.

4.1.1 Minimax

A well-known method of recovery from game theory is called the minimax method. The minimax algorithm iterates around the game tree to find the optimal move and returns based on the leaf score. You can see a more detailed explanation and pseudocode earlier.
In a two-player zero-sum game, each player tries to minimize his own expected loss (penalty) while maximizing his opponent's damage . The re-

gression technique called minimax is used for decision making in game theory. Let's assume that both players are playing at the highest level. The terms "maximizer" and "minimizer" refer to two players. The first looks for the highest score, the second looks for the lowest score. The name "Minimax" came about because when one player wins, it automatically means the other player loses.
Every zero-sum game has a score at any given time, which will be positive if the maximizer leads, and positive if the minimizer enters first. negative. To better explain this idea, consider a game where player 1 can get V as the maximum reward if player 2 is also the best and chooses the most money-making decision -V. Tic-tac-toe, checkers, Connect 4, etc. Many two-player zero-sum games can be played this way, including Here we will look at a Connect 4 example with the help of pictures.

The game of Connect 4 has two players: X and O. In this case, X is the maximizer and O is the minimizer; Everyone has an equal chance of winning, losing or drawing. They will choose this move if it will put them in a position where they have a good chance of winning. Also if the current player cannot win, the player will try to make a draw. With the help of the picture below, the game has progressed to a certain point and now X needs to participate, let's understand it better

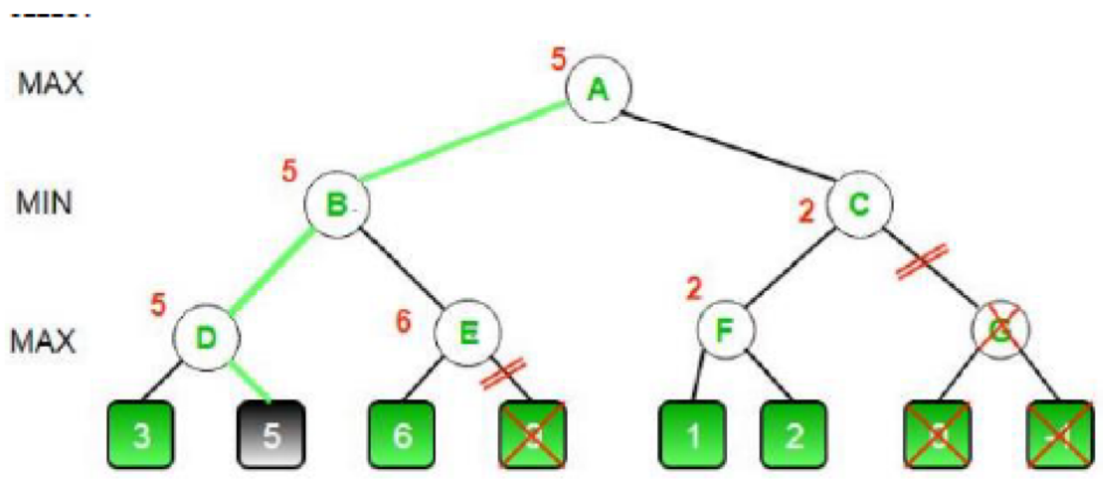


Figure 3: Demonstration of Connect 4 using Minmax Algorithm

As can be observed, in the first level of the game tree, player X has a choice of 3 alternative nodes. However, closer examination shows that if player X chooses to play the left node, [2,0], player O will have two alternatives for the following move, and as both players aim to maximise them. In order to win the game, player O will select the first node since it yields a score of -10, whereas the other move would result in a tie and

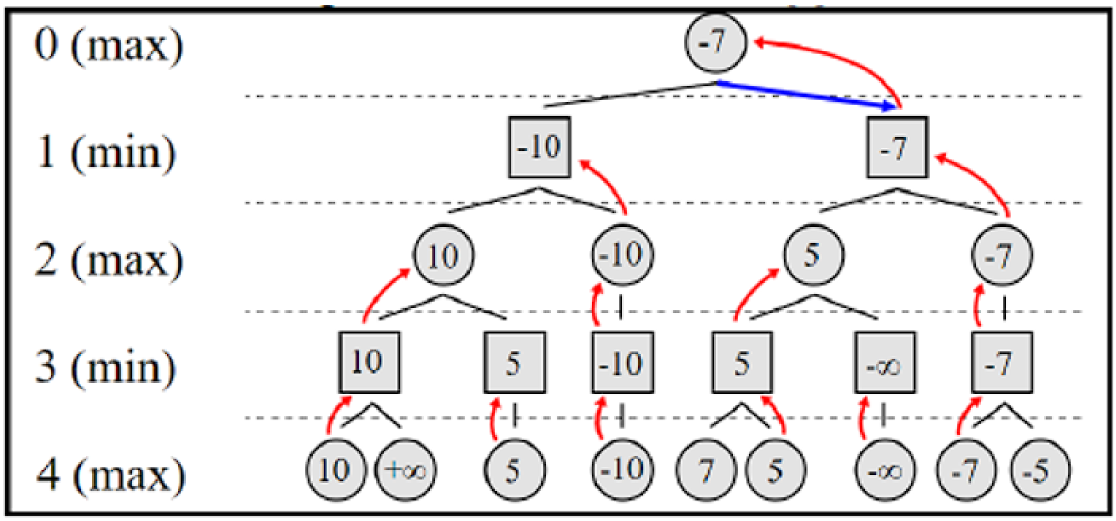


Figure 4: Min-max Generated Tree

a score of zero. Therefore, adopting this action for player X is not ideal since it will result in Player O receiving the most advantage and the least amount of punishment. [4]

Player O will have to select one of two nodes at the following level if player X takes the middle move, [2,1], however, player O cannot win in either of the two moves because the 1st node evaluates to +10 and the 2nd one results in net zero. Player O will try to maximize the maximum possible award for themselves and minimize the maximum possible reward for their opponent as they play optimally as well, and as a result, they will select the second node to conclude the game in a tie. As a result, player X will have the highest chance of drawing by selecting the centre node. However, if player X selects the correct node, [2, 2], then it will immediately result in player X's win with a value of +10, in order to maximise points, the player X will always play ideally.

4.1.2 Alpha-Beta Pruning

To find the optimal solution, the minimax algorithm must traverse the entire game tree. The Minimax method is improved by the Alpha-Beta pruning technique, which prunes tree nodes that are almost impossible to provide better movements and does not measure them [3]. Pruning across all branches of the hunting tree can save a lot of time. However, the best performance of the Alpha-Beta pruning algorithm was compared with the Minimax algorithm.

Minimax examines every possible part of the game tree and grows exponentially as the depth of the tree increases. It prunes away unnecessary branches as better consequences of the action are discovered. Alpha-beta pruning is so named because it does this by adding two parameters (specifically alpha and beta) to the minimax algorithm.

When used on a typical minimax tree, it performs the same operations as a minimax tree but also verifies its correctness by removing branches that are not relevant to the final selection. Alpha-beta pruning has the advantage of allowing branches of the search tree to be removed. This way, deeper searches can be performed while limiting search time for "more" subtrees as shown in figure 5.

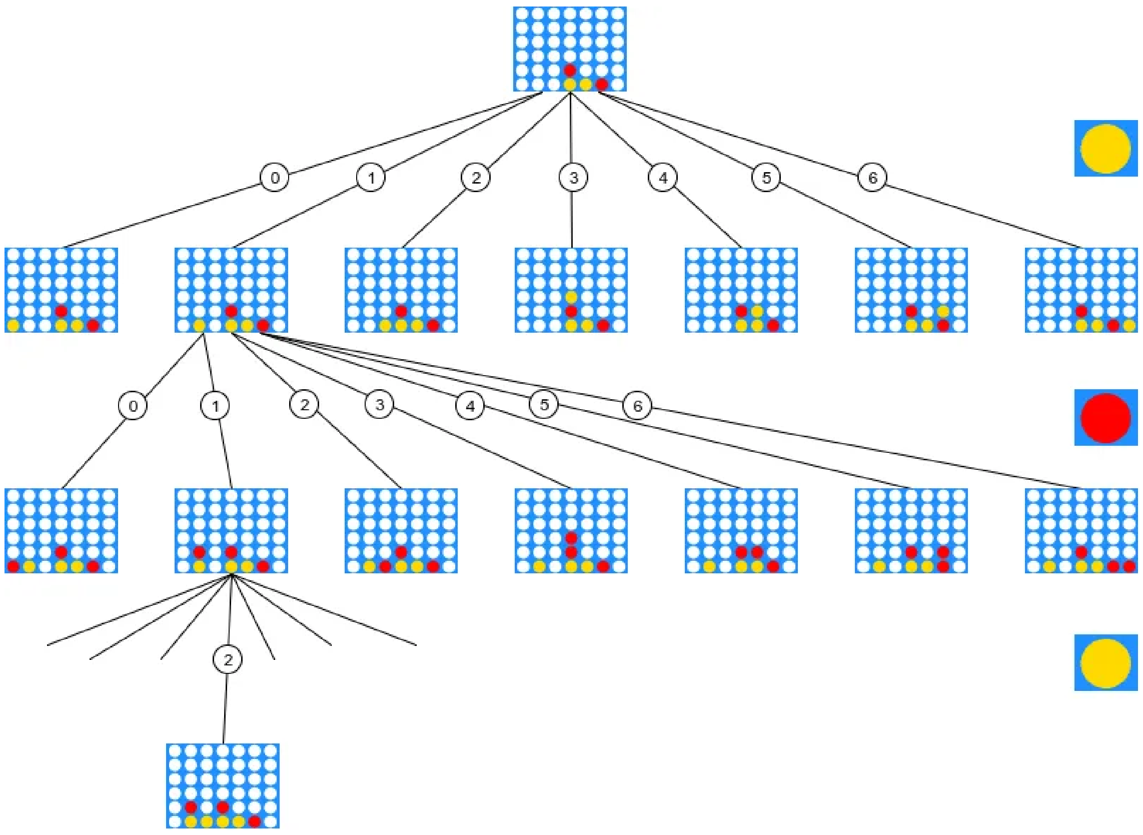


Figure 5: Alpha-beta visualization

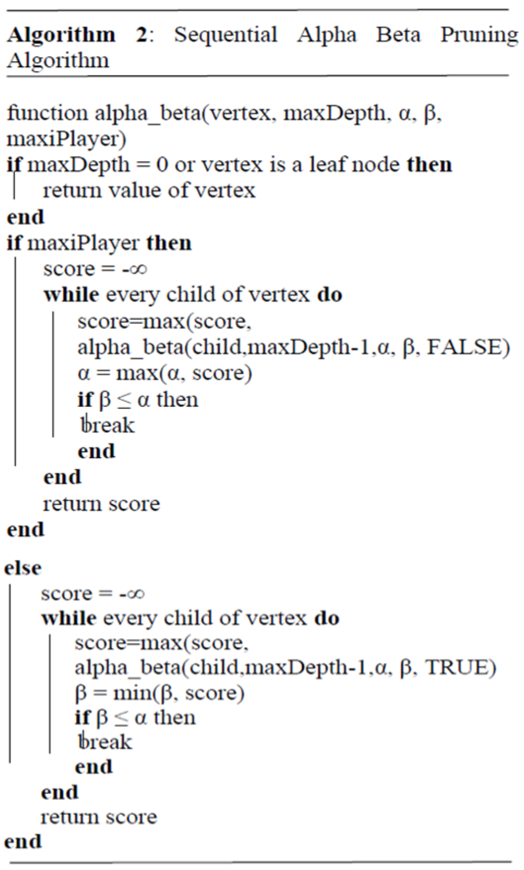
Alpha-beta pruning has two parameters: alpha and beta. The maximizer at or above the specified level has maximum alpha, and the minimizer at or above the specified level has maximum beta.

Let's see how these two restrictions apply in practice. We know that for maximum the score will increase to positive and for minimum the score will increase to negative, so for the first time alpha is set to minimum, all values are negative and beta is set to zero. maximum value, meaning two. Players all start with the lowest score. Now we will check the situation in which a subtree will be deleted. If a level is reached when the maximizer's maximum score is less than the maximizer's lowest score, we can discard that tree and it will be ignored.

4.2 PSEUDOCODES

4.2.1 Alpha-Beta Pruning

Here is Alpha-Beta Pseudocodes Figure 6



Algorithm 2: Sequential Alpha Beta Pruning Algorithm

```
function alpha_beta(vertex, maxDepth, α, β, maxiPlayer)
if maxDepth = 0 or vertex is a leaf node then
    return value of vertex
end
if maxiPlayer then
    score = -∞
    while every child of vertex do
        score=max(score, alpha_beta(child,maxDepth-1,α, β, FALSE)
        α = max(α, score)
        if β ≤ α then
        break
        end
    end
    return score
end
else
    score = -∞
    while every child of vertex do
        score=max(score, alpha_beta(child,maxDepth-1,α, β, TRUE)
        β = min(β, score)
        if β ≤ α then
        break
        end
    end
    return score
end
```

Figure 6: Alpha-Beta Pseudocodes

5 RESULT

This algorithm allows removal of the search tree. It also allows deeper searches by limiting search time to more mature trees. The algorithm also reduces the number of calculations and searches during the minmax algorithm. It also avoids the use of additional calculation time, making the process more responsive and faster. When used on a minimax tree, it produces the same moves as a minimax tree, but also verifies its correctness by removing branches that are not possible. affects the final selection. Alpha-beta pruning has the advantage of allowing branches of the search tree to be removed. This way, deeper searches can be performed while limiting search time for ”more” subtrees.

6 CONCLUSION

In summary, this article helps to use Alpha-Beta pruning and Minimax to find the best parameters to run the Connect-4 game well to get maximum profits. Adjustments were made for search depth and board size to examine how this change affected the overall performance of the algorithm based on different models. All parameters are varied individually while other parameters are kept constant to follow the progress of the results, but successful optimization is possible if all parameters are varied simultaneously. According to our findings, the Alpha-Beta pruning algorithm with Minimax is the most efficient with a search depth of or 5 as it achieves the best time required for the determined and is proven to have a high win rate. Although a deeper search may produce better results, it may not achieve good performance when measured by other criteria such as execution time. The most important consideration for any online game is to ensure that the game runs well and responds quickly to all user interactions; Therefore everything else becomes secondary. The same common sense applies here as Link 4 is a turn-based game, so getting the best timing should be the main theme, follow it and get the reward won.

7 FUTURE SCOPE

Here’s a perspective on the future scope of the alpha-beta algorithm:
Key Areas for Continued Impact:

- Gaming and Ai:
  - Refining strategies in complex games like Go and chess, potentially leading to superhuman AI players.
  - Enhancing decision-making in real-time strategy games and multiplayer online games.
- Optimization and Decision-making:
  - Tackling Np- Hard problems in areas like scheduling, resources allocation, and travel planning.

    - Incorporating domain-specific knowledge to guide pruning and improve efficiency in real-world applications.

- Machine learning and Automated Planning:
    - Integration with reinforcement learning to guide exploration and accelerate training in large state spaces.
    - Efficient generation for optimal or near-optimal plans in dynamic and uncertain environments.

- Parallel and Distributed computing:
    - Scaling alpha-beta pruning to massive search trees using parallel and distributed architectures.
    - Leveraging cloud computing for large-scale problem-solving.

- Specific Research Directions:
    - Selective Pruning Techniques:
        - Developing methods to strategically prune branches based on their potential for pruning others, further reducing search space.
    - Dynamic Alpha-Beta Pruning:
        - Adjusting alpha-beta values dynamically based on the search progress, potentially leading to deeper and more accurate searches.
    - Hybrid Approaches:
        - Combining alpha-beta pruning with other search algorithm (e.g. monte Carlo tree Search) for complementary strengths.
    - Integration with Neural Network:
        - Exploring ways to integrate alpha-beta pruning with neural networks for improved decision-making in complex, uncertain environments.

Overall, the alpha-beta algorithm remains a powerful tool with significant potential for further development and application across various domains.

References

[1] Sinan Ariyurek, Aysu Betin-Can, and Elif Surer. Enhancing the monte carlo tree search algorithm for video game testing. In *2020 IEEE Conference on Games (CoG)*, pages 25–32. IEEE, 2020.

[2] Pranav Gangwar, Satvik Maurya, and Neeta Pandey. Realization of game tree search algorithms on fpga: A comparative study. In *2019 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT)*, volume 1, pages 1–3. IEEE, 2019.

[3] SRAVYA MANDADI, B TEJASHWINI, SANJAN VIJAYAKUMAR, et al. Implementation of sequential and parallel alpha-beta pruning algorithm. *International Journal of Innovations in Engineering Research and Technology*, 7(08):98–104, 2020.

[4] Rijul Nasa, Rishabh Didwania, Shubhranil Maji, and Vipul Kumar. Alpha-beta pruning in mini-max algorithm–an optimized approach for a connect-4 game. *Int. Res. J. Eng. Technol*, pages 1637–1641, 2018.

[5] Meenakshi Pawar. International journal of advances in engineering and management (ijaem). *Algorithms*, 5(9):122–125.

[6] Shubhendra Pal 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)*, pages 115–119. IEEE, 2019.

[7] Maciej Świechowski, Konrad Godlewski, Bartosz Sawicki, and Jacek Mańdziuk. Monte carlo tree search: A review of recent modifications and applications. *Artificial Intelligence Review*, 56(3):2497–2562, 2023.