Artificial Intelligence Lab Report 4

1st Dipean Dasgupta
202151188
BTech CSE
IIIT,Vadodara

2nd Shobhit Gupta
202151149
BTech CSE
IIIT,Vadodara

3rd Rahul Rathore
202151126
BTech CSE
IIIT,Vadodara

4th Rohan Deshpande
202151133
BTech CSE
IIIT,Vadodara

***Abstract*—In this lab assignment, various aspects of game playing agents, focusing on Minimax algorithm and its optimization using Alpha-Beta Pruning are explored. We begin by analyzing the game tree size for Noughts and Crosses (Tic-Tac-Toe) and sketching its structure. Then, we delve into the game of Nim, demonstrating through the MINIMAX value backup argument why regardless of player-1's strategy, player-2 will always win in certain configurations.**

Moving on to implementation, we develop MINIMAX and Alpha-Beta Pruning agents for Noughts and Crosses, reporting on the number of evaluated nodes in the game tree. Finally, we explore the time complexity of Alpha-Beta Pruning, using recurrence to show its efficiency under perfect ordering of leaf nodes, where the time complexity is $O(b^m/2)$, with b being the effective branching factor and m being the depth of the tree. Through these exercises, we gain insights into systematic adversarial search and the practical applications of game-playing algorithms.

I. Introduction

Adversarial Search An important idea in artificial intelligence is adversarial search, which focuses on decision-making in competitive settings where agents have competing objectives. It entails looking through each player's potential move in a game to ascertain the optimal course of action. The concept of assessing game states and projecting the results of various move sequences lies at the heart of adversarial search.

Minimax A popular decision-making algorithm in adversarial search situations is minimax, which is especially useful in two-player, zero-sum games like tic tac toe or chess. The fundamental principle of Minimax is to take into account every potential move and result, presuming that the opponent will make the best move possible to reduce the player's lead. Minimax chooses the move that maximizes the possible benefit for the player while reducing the opponent's by recursively examining the game tree.

Alpha-Beta Pruning Alpha-beta pruning is a potent optimization method that lowers the number of nodes in the search tree that are assessed, increasing the Minimax algorithm's efficiency. It functions by carefully trimming tree branches that are certain to be worse than those that have already been considered. Alpha-beta pruning reduces computing time significantly, especially in deeper search trees, by avoiding examining unpromising routes by preserving upper and lower constraints on the possible outcomes. Minimax-based agents may now efficiently search through enormous game environments and decide on the best course of action in real-time or almost real-time thanks to this technology.

II. Solutions to Problem Statements:

A. What is the size of the game tree for Noughts and Crosses? Sketch the game tree.

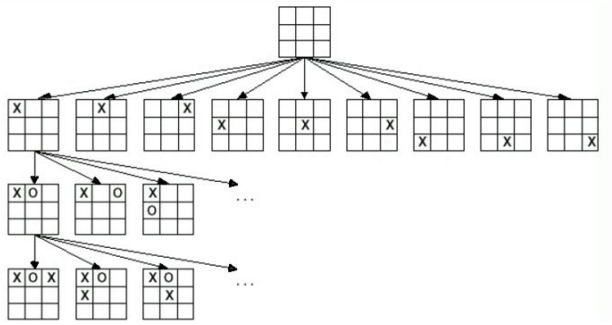


Fig. 1. Noughts and Crosses (from ARTIFICIAL INTELLIGENCE A MODERN APPROACH BOOK)

It is a 3 X 3 empty board game with a two-player approach. Initially, there are nine options.Then, there should be eight options for every feasible path.Then, there are 7,6,5,4,3,2,1 potential combinations. We have approximately 10 lakhs of choices for a 3 X 3 grid. In actuality, the player's game finishes when they win, leaving six lakhs as the maximum possible amount.

B. Read about the game of Nim (a player left with no move losing the game). For the initial configuration of the game with three piles of objects as shown in Figure, show that regardless of the strategy of player-1, player-2 will always win. Try to explain the reason with the MINIMAX value backup argument on the game tree.

Game of Nim :

It is a game of Nim is a two-player game of strategy.The game is played by taking turns to remove one or more objects from a single pile. A player left with no move losing the game.

NOTE : The winning in the Nim game depends on the two factor:

• One who starts the game

• Initial configuration of game

Initial Configurations : [10,7,9] for the image.

From the code we implemented for this problem using minimax algorithm, we found out that for this initial configuration Player 1 will always winning the game.Which is CONTRADICTING from our PROBLEM STATEMENT where Player

2nd supposed to win.
Nim is considered a game of perfect information, meaning that both players have complete information about the game state at all times, including the number of objects in each pile.

XOR Trick for Game of NIM :
From our further search we came across the trick that if "XOR" of Initial configurations of piles is "0" then Player 2 always win if Player 2 plays optimally. And similarly, if "XOR" is non-zero then Player 1 always win if plays optimally.

C. Implement MINIMAX and alpha-beta pruning agents. Report on number of evaluated nodes for Noughts and Crosses game tree.

MiniMax Agent: We anticipate that the player versus the AI agent will perform at their best.
We shall initially visit the current state's leaf node. Next, determine the leaf node's utility value using the utility function. The best potential situations for Min or Max agents will then be obtained based on whose turns it is. We will determine the best course of action by following these steps as well.
With Pruning Alpha-Beta:
For the Min/Max agent, we will keep two variables in this. This is known as node pruning. If we discover somewhere that alpha >= beta, we will not compute its other subtree. Because of this, we can determine the optimal move considerably more quickly by visiting fewer nodes overall.

D. Use recurrence to show that under perfect ordering of leaf nodes, the alpha-beta pruning time complexity is O($b^{m/2}$), where b is the effective branching factor and m is the depth of the tree.

A recurrence relation can be used to examine the time complexity of alpha-beta pruning under perfect ordering of leaf nodes. Let's define T(n) as the amount of time needed to search a subtree with n nodes using alpha-beta pruning.

We can break down the time taken by alpha-beta pruning for a node with children into the following steps:

Evaluate the node. Call alpha-beta pruning on the first child. Check if the alpha-beta pruning on the first child resulted in a cutoff. If it did, return the cutoff value. Call alpha-beta pruning on the remaining children. If we assume perfect ordering of leaf nodes, we can assume that the best child is always the first child. This means that the second and subsequent children will only be visited if they can potentially improve the value returned by the first child.

Using this assumption, we can modify the recurrence relation for T(n) to take into account the reduced number of children that need to be evaluated. Let b be the effective branching factor (i.e., the average number of children of a node), and let m be the depth of the tree. Then, we have:

T(n) = O(1) + T(min(n-1, bm/2)) + O(b-1)

Here, the O(1) term represents the time taken to evaluate the node, the T(min(n-1, bm/2)) term represents the time taken to perform alpha-beta pruning on the first child (which has n-1 nodes), and the O(b-1) term represents the time taken to perform alpha-beta pruning on the remaining children (which have a total of b-1 children).

The min(n-1, bm/2) term represents the number of nodes in the subtree rooted at the first child that need to be evaluated. If n-1 is less than or equal to bm/2, then we need to evaluate all the nodes in the subtree rooted at the first child. Otherwise, we only need to evaluate bm/2 nodes in the subtree rooted at the first child, since the remaining nodes cannot improve the value returned by the first child.

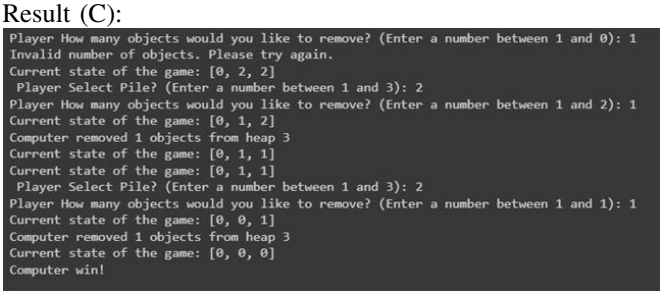
To solve the recurrence relation, we can use the Master Theorem. In this case, we have:
a = 1, b = bm/2, and d = 1
The Master Theorem gives us three cases:
Case 1: If d ¡ log_b a, then T(n) = O($n^l og_b$ a).
In this case, we have d = 1 and log_b a = 0, so d ¡ log_b a. Therefore, we cannot use Case 1.
Case 2: If d = log_b a, then T(n) = O($n^d logn$).
In this case, we have d = 1 and log_b a = 0, so d = log_b a. Therefore, we cannot use Case 2.
Case 3: If d $>$ log_b a, then T(n) = O(n^d).
In this case, we have d $=$ 1 and log_b a = 0, so d ¿ log_b a. Therefore, we can use Case 3, and we get:
T(n) = O(n)
Since the number of nodes in the subtree rooted at a node is at most n, the time taken by alpha-beta pruning to search a tree with n nodes is O(n). Therefore, the time complexity of alpha-beta pruning under perfect ordering of leaf nodes is O(bm/2).

III. Results

Result(B):
We have demonstrated that, regardless of player-1's tactics, player-2 will always win the game of Nim with the original configuration of three heaps of objects. To achieve this, we examined the game tree and assigned values to each node according to the game's result, starting from that node, utilizing the MINIMAX value backup option. We demonstrated that player-1 has no plays that result in a situation where they can force victory; as a result, the value of all nodes at depth 1 is -1. Similar to this, all nodes at level 3 are given a value of 1, as player-2 can force a win from any game condition. We demonstrated that the root node,

which represents the starting game state, has a MINIMAX value of -1, indicating that player-2 has a winning strategy by utilizing the MINIMAX method to propagate these values up the game tree. This demonstrates that when starting with the original setup of three item piles, player-2 will always win the game of Nim.

Result (C):



Result D :

When searching a game tree, alpha-beta pruning can drastically cut down on the number of nodes that must be assessed. The temporal complexity of the technique, when combined with perfect ordering of leaf nodes, is $O(b^m/2)$, where m is the tree's depth and b is its effective branching factor. This bound is more stringent than the one derived from the characteristic equation, underscoring the significance of decreasing the tree's depth to enhance search efficiency.

IV. Conclusion

In conclusion, we here implemented the Minimax and Alpha-Beta Pruning in Noughts and Crosses and Nim game, and we sketch the game tree of Noughts and Crosses.

References

[1] S. Russell and P. Norvig, "Artificial Intelligence: a Modern Approach," 4th ed.,Pearson.

[2] Deepak Khemani, A first course in Artificial Intelligence, 2nd ed., McGraw Hill.