CS364
Artificial Intelligence Laboratory
Week-05
Instructor: Dr. Pratik Shah

Group Name: AI Beginners
Prasoon Pathak – 202051143
Pushkaraj Kolekar – 202051149
Rashiya Dhruv Hareshbhai – 202051153
Suyash Jain – 202051189

Lab Assignment - 04

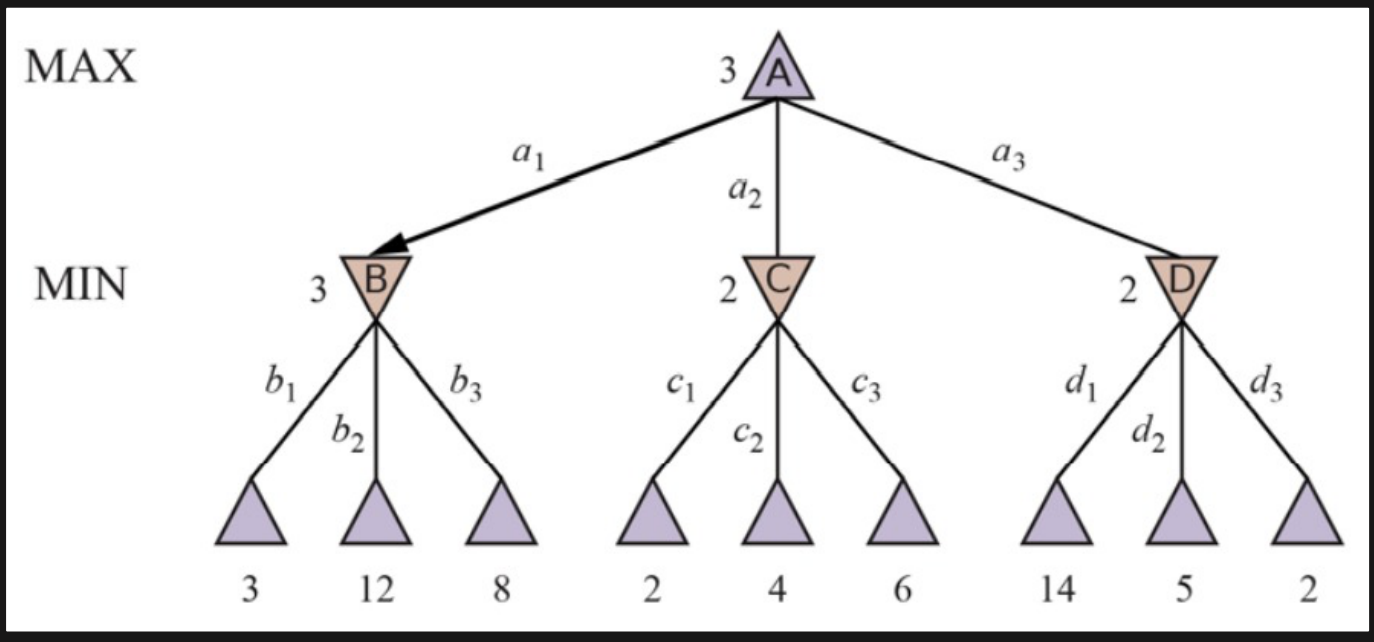
Learning Objectives: Game Playing Agent | Minimax | Alpha-Beta Pruning.

Systematic Adversarial Search can lead to savings in terms of pruning of sub-trees resulting in lesser node evaluations.

Minimax Search

- Minimax is a **recursive** decision-making algorithm used in AI for two-player games.
- It evaluates all possible moves of the players and returns the best move for the current player by considering the worst-case scenario for the opponent's next move.
- The algorithm minimizes the maximum loss (minimizes the worst outcome) for the current player.
- The game tree is explored until a terminal state is reached (win, loss, or draw) and the minimax value is assigned to the state, thus the algorithm uses **DFS** for the exploration of the game tree.
- The algorithm is used to find the best move for the current player and is guaranteed to find the optimal solution for games with perfect information.
- If **the maximum depth of the tree is m** and there are b legal moves at each point, i.e., the **branching factor is b** then,

Time Complexity = O(b^m)
Space Complexity = O(m)

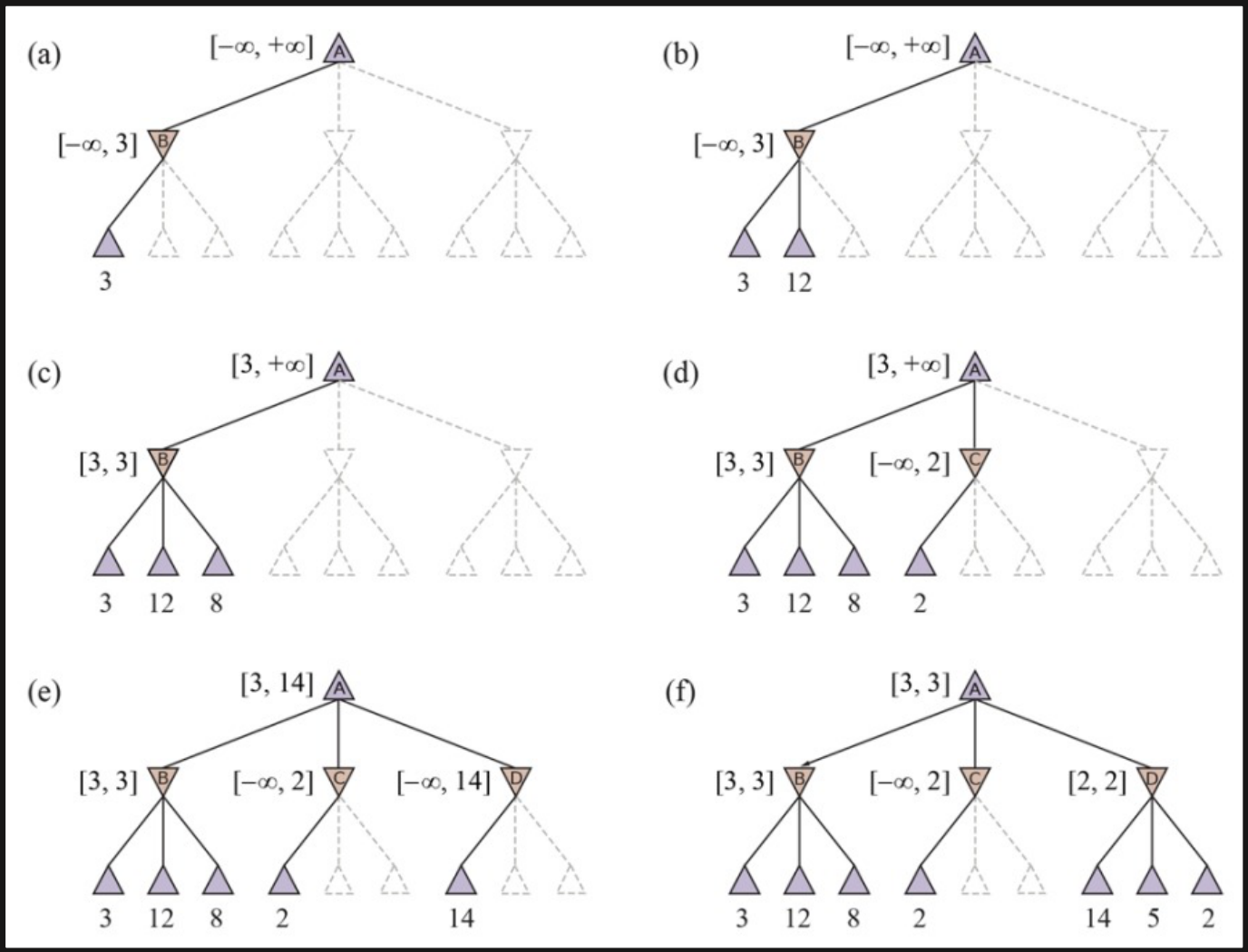


Disadvantages of Minimax Search

- The exponential complexity of minimax algorithm makes it impractical for complex games.
- **Example:** In case of chess the branching factor is about 35, and average game depth is about 80-**ply**, i.e., total moves and it isn't feasible to search **35^{80} States.**

<u>Alpha-Beta Pruning</u>

- No algorithm can completely eliminate the exponent, but we can sometimes cut it in half by **pruning**, i.e., eliminating unproductive sub-trees from the tree. This particular technique is called **Alpha-Beta Pruning.**
- It eliminates branches that can't possibly influence the final decision, thus, improving the efficiency of the search.
- This algorithm uses 2 parameters: alpha and beta (max and min) to keep track of the best move found so far and to prune branches that are no longer relevant.

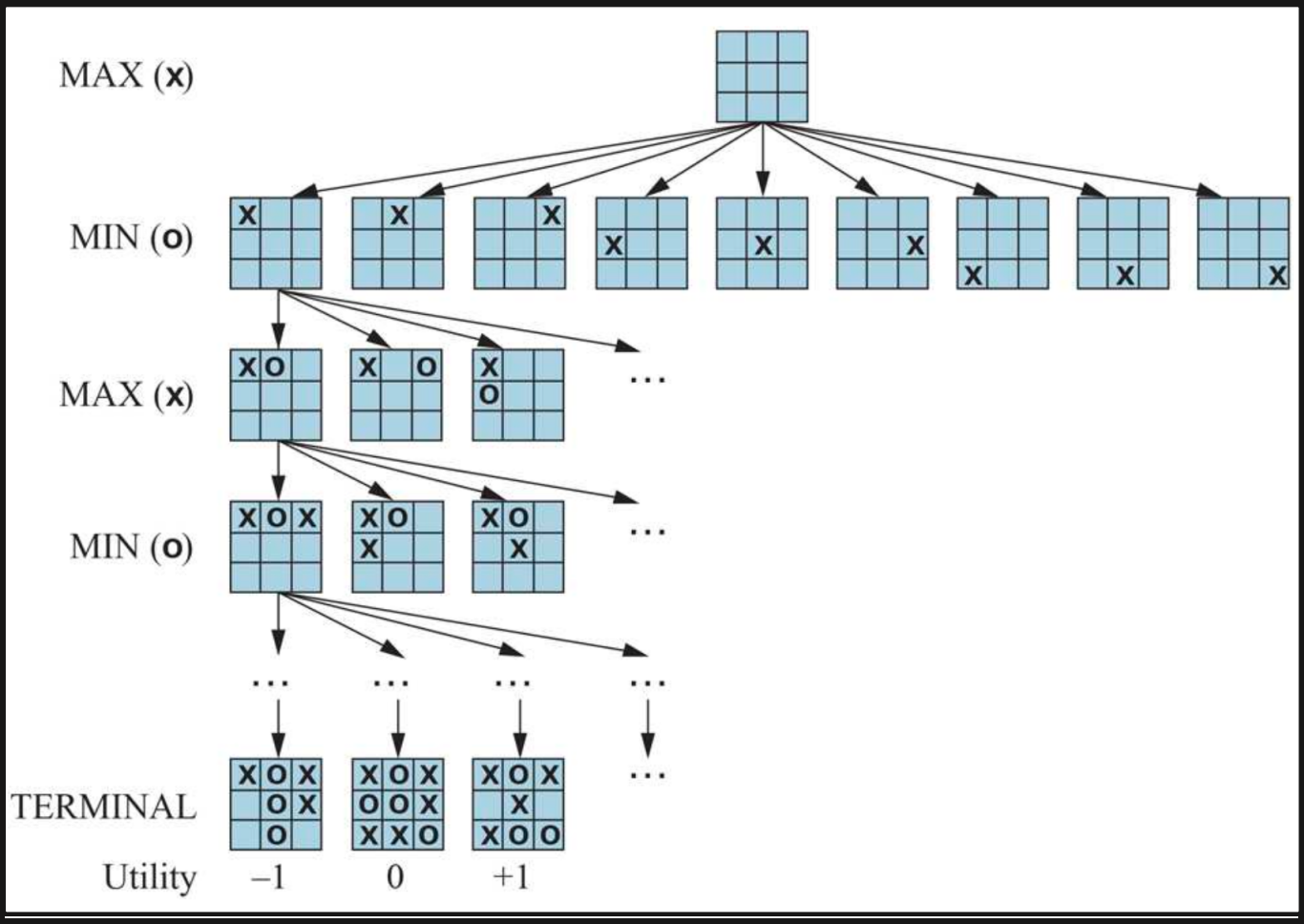


- The effectiveness of Alpha-Beta Pruning is highly dependent on the order in which the states are examined.
- If all the best moves are examined first, then the, Time Complexity of Alpha-Beta Pruning is reduced to $O(b^{m/2})$.
- This means that the branching factor becomes $b^{1/2}$ instead of b, therefore in case of Chess, branching factor becomes 6 instead of 35.
- For moderate b and random move ordering the Time Complexity is roughly $O(b^{3m/4})$.

- We can further reduce the search cost by keeping a transposition table of previously reached states.

Questions

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



But 9! (3,62,880) Conditions would also include games that would continue after a victory.
So, we would have to remove them !

Now, there are 8 lines of 3 squares (3 vertical, 3 horizontal and 2 diagonal) for winning the game.
Doing Some Calculations…

Win in 5 moves : 1440 possibilities.
Win in 6 moves : 5328 possibilities.
Win in 7 moves : 47952 possibilities.
Win in 8 moves : 72576 possibilities.
Win in 9 moves : 81792 possibilities.

Also, there will be total 16 possible patterns for the 5 X's and 4 O's (or vice versa) which have no 3 in the same row.

So, we are looking at 16 x 5! x 4! = 46080 possibilities for games ending in a **Draw** on the ninth move.
Thus in total we are looking at 81792+46080 = 127872 possibilities for games lasting as long as the ninth move. Therefore, 1440 + 5328 + 47952 + 72576 + 81792 + 46080 = 255168 possible games in total.

2. **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.**

- In case of Nim, with 3 piles, the minimax value of the position is determined by the minimax values of its child nodes which correspond to the possible moves that can be made from that position.
- Let us consider if player 2 wins, the value is 1 and if player 1 wins the value is -1 at the terminal nodes of the game tree where there are no moves to be made.
- As the algorithm works its way up the game tree, it updates the minimax values of each node by choosing the minimum value of its child nodes for player 1 and the maximum value of its child nodes for player 2.
- In this case, player 2 can always force player 1 to make a move, that leads to a terminal node with minimax value of -1 which means player 2 will always win !
- This is because player 2 can always mirror player 1's moves on a different pile.

3. **Implement MINIMAX and alpha-beta pruning agents. Report on number of evaluated nodes for Noughts and Crosses game tree.**
   https://colab.research.google.com/drive/1gYnZMkVj0YL0V1BoqmRp3r6kdEjjidU8?usp=sharing

4. **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.**

$$T(m) = b\ T(m-1) + O(1), \text{ if } m > 0$$
$$T(0) = O(1)$$

- The first term in the equation, b T(m-1), represents the time complexity of evaluating all of the child nodes of a given node at depth m.
- The second term, O(1), represents the time complexity of evaluating the node itself.
- By substituting T(m-1) with the equation for T(m), we can simplify the recurrence relation:

$$T(m) = b\ T(m-1) + O(1) = b\ (b\ T(m-2) + O(1)) + O(1) = b^2\ T(m-2) + b\ O(1) + O(1)$$

- Continuing this process, we can see that the time complexity of the alpha-beta pruning algorithm can be expressed as:

$$T(m) = b^m\ T(0) + b^{(m-1)}\ O(1) + b^{(m-2)}\ O(1) + \ldots + O(1) = O(b^m)$$

References

1. A first course in Artificial Intelligence, Deepak Khemani (Chapter 8)
2. Artificial Intelligence: a Modern Approach, Russell and Norvig (Fourth edition) (Chapter 5)
3. CITS3001, Algorithms, Agents and Artificial Intelligence, Semester 1, 2016, Lyndon While School of Computer Science & Software Eng. The University of Western Australia, 8. Game-playing AIMA, Ch. 5. https://teaching.csse.uwa.edu.au/units/CITS3001/Semester1/lectures/lectures/3001%20Game-playing.pdf
4. Game of Nim, Link: https://www.archimedes-lab.org/game_nim/play_nim_game.html