M.Sc. (Computer Science)- Fourth Semester
Subject Name- “Artificial Intelligence and Expert System”
Paper-II (Video Part-1)

By- Prof. Dileep Kumar Sahu

Assistant Professor

Department of Computer Application

Govt. Vishwanath Yadav Tamaskar Post Graduate Autonomous College, Durg (C.G.)

Email ID: dileepksahu20@gmail.com

Game Playing

Mini-Max Search

BY- PROF. DILEEP KUMAR SAHU

Games and Game Tree

- There might be some situations where more than one agent is searching for the solution in the same search space, and this situation usually occurs in game playing.(Multi-agent systems + competitive environment)
- In game theory any multiagent environment is a game as long as each agent has “significant” impact on others.
- So, **Searches in which two or more players with conflicting goals are trying to explore the same search space for the solution, are called adversarial searches, often known as Games**.
- Games are modeled as a Search problem and heuristic evaluation function, and these are the two main factors which help to model and solve games in AI.

Types of Games

	Deterministic	Chance Moves
Perfect Information	Chess, Checkers, GO	Backgammon, monopoly
Imperfect Information	Battleships, blind, tic-tac-toe	Bridge, poker, scrabble

Types of Games

- **Perfect information:** A game with the perfect information is that in which agents can look into the complete board.
  - Agents have all the information about the game, and they can see each other moves also.
  - Examples are Chess, Checkers, Go, etc.
- **Imperfect information:** If in a game agents do not have all information about the game and not aware with what's going on, such type of games are called the game with imperfect information, such as:
  - tic-tac-toe, Battleship, blind, Bridge, etc.
- **Deterministic games:** Deterministic games are those games which follow a strict pattern and set of rules for the games, and there is no randomness associated with them.
  - Examples are chess, Checkers, Go, tic-tac-toe, etc.
- **Non-deterministic games:** Non-deterministic are those games which have various unpredictable events and has a factor of chance or luck.
  - This factor of chance or luck is introduced by either dice or cards. These are random, and each action response is not fixed.
  - Such games are also called as stochastic games. Example: Backgammon, Monopoly, Poker, etc.

Mini-Max Search Algorithm in Game Theory

- **Minimax** is a decision rule used in artificial intelligence, decision theory, game theory, statistics, and philosophy for *mini*mizing the possible loss for a worst case (*max*imum loss) scenario.
- When dealing with gains, it is referred to as "maximin"—to maximize the minimum gain.
- Originally formulated for two-player zero-sum game theory, covering both the cases where players take alternate moves and those where they make simultaneous moves, it has also been extended to more complex games and to general decision-making in the presence of uncertainty.

Mini-Max Search

- Mini-max algorithm is a recursive or backtracking algorithm which is used in decision-making and game theory. It provides an optimal move for the player assuming that opponent is also playing optimally.
- Mini-Max algorithm uses recursion to search through the game-tree.
- Min-Max algorithm is mostly used for game playing in AI. Such as Chess, Checkers, tic-tac-toe, go, and various tow-players game. This Algorithm computes the minimax decision for the current state.
- In this algorithm two players play the game, one is called MAX and other is called MIN.
- Both the players fight it as the opponent player gets the minimum benefit while they get the maximum benefit.
- Both Players of the game are opponent of each other, where MAX will select the maximized value and MIN will select the minimized value.
- The minimax algorithm performs a depth-first search algorithm for the exploration of the complete game tree.
- The minimax algorithm proceeds all the way down to the terminal node of the tree, then backtrack the tree as the recursion.

Mini-Max Search Algorithm

```
function minimax(node, depth, maximizingPlayer) is
  if depth ==0 or node is a terminal node then
    return static evaluation of node
  if MaximizingPlayer then      // for Maximizer Player
      maxEva= -infinity
      for each child of node do
          eva= minimax(child, depth-1, false)
          maxEva= max(maxEva,eva)       //gives Maximum of the values
      return maxEva
  else                    // for Minimizer player
     minEva= +infinity
     for each child of node do
        eva= minimax(child, depth-1, true)
        minEva= min(minEva, eva)        //gives minimum of the values
     return minEva
```

- **Initial call:**
- **Minimax(node, 3, true)**

Working of Min-Max Algorithm:

- The working of the minimax algorithm can be easily described using an example. Below we have taken an example of game-tree which is representing the two-player game.
- In this example, there are two players one is called Maximizer and other is called Minimizer.
- Maximizer will try to get the Maximum possible score, and Minimizer will try to get the minimum possible score.
- This algorithm applies DFS, so in this game-tree, we have to go all the way through the leaves to reach the terminal nodes.
- At the terminal node, the terminal values are given so we will compare those value and backtrack the tree until the initial state occurs.

Working of Mini-Max Search

- Following are the main steps involved in solving the two-player game tree:

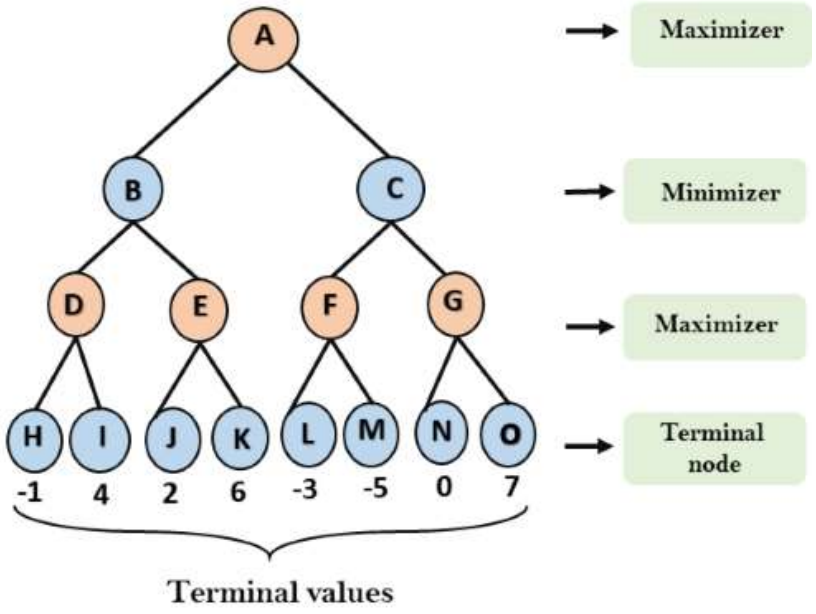
Step-1: In the first step, the algorithm generates the entire game-tree and apply the utility function to get the utility values for the terminal states. In the below tree diagram, let's take A is the initial state of the tree.

Suppose maximizer takes first turn which has worst-case :

initial value =- infinity,

and minimizer will take next turn which has worst-case:

initial value = +infinity.

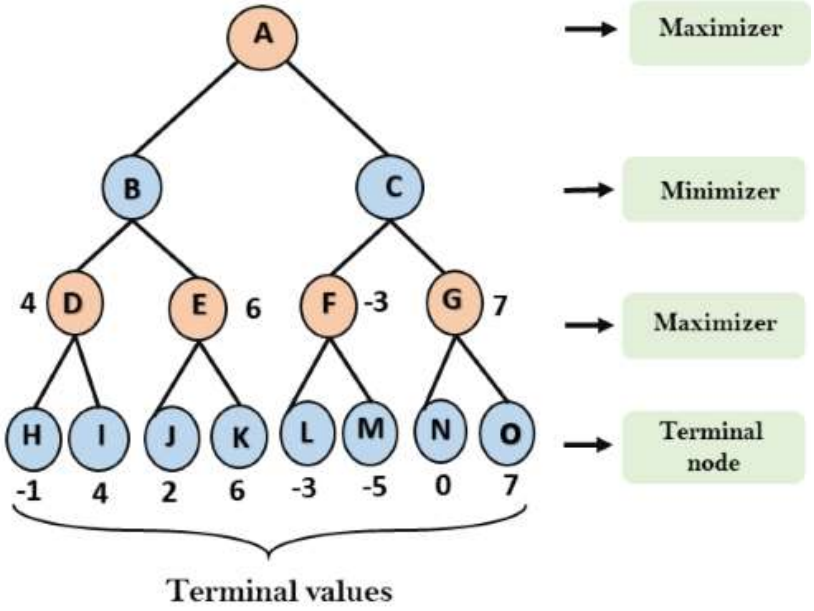


Working of Mini-Max Search

- **Step 2:** Now, first we find the utilities value for the Maximizer, its initial value is -∞, so we will compare each value in terminal state with initial value of Maximizer and determines the higher nodes values.

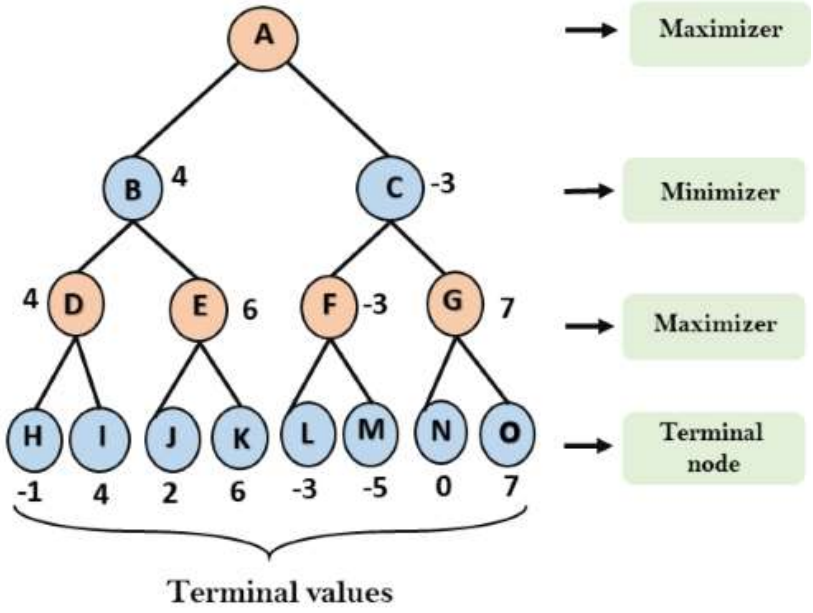
It will find the maximum among the all.

- 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



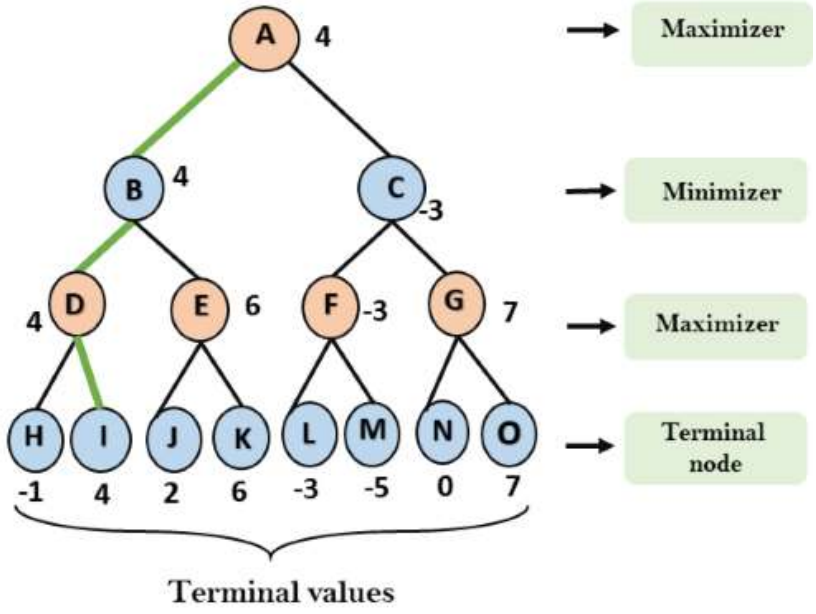
Working of Mini-Max Search

- **Step 3:** In the next step, it's a turn for minimizer, so it will compare all nodes value with +∞, and will find the 3rd layer node values.
- For node B= min(4,6) = 4
- For node C= min (-3, 7) = -3



Working of Mini-Max Search

- **Step 3:** Now it's a turn for Maximizer, and it will again choose the maximum of all nodes value and find the maximum value for the root node. In this game tree, there are only 4 layers, hence we reach immediately to the root node, but in real games, there will be more than 4 layers.
- For node A max(4, -3)= 4
- That was the complete workflow of the minimax two player game.



Properties of Mini-Max algorithm

- **Complete-** Min-Max algorithm is Complete. It will definitely find a solution (if exist), in the finite search tree.
- **Optimal-** Min-Max algorithm is optimal if both opponents are playing optimally.
- **Time complexity-** As it performs DFS for the game-tree, so the time complexity of Min-Max algorithm is **$O(b^m)$**, where b is branching factor of the game-tree, and m is the maximum depth of the tree.
- **Space Complexity-** Space complexity of Mini-max algorithm is also similar to DFS which is **O(bm)**.

Limitation of the minimax Algorithm

- The main drawback of the minimax algorithm is that it gets really slow for complex games such as Chess, go, etc.
- This type of games has a huge branching factor, and the player has lots of choices to decide.

References

- . Elaine Rich and Kevin Knight: Artificial Intelligence- Tata McGraw Hill.
- https://www.javatpoint.com/mini-max-algorithm-in-ai

Thank You

BY- PROF. DILEEP KUMAR SAHU