M.Sc. (Computer Science)- Fourth Semester
Subject Name- “Artificial Intelligence and Expert System”
Paper-II (Video Part-2)

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

Alpha Beta Pruning

BY- PROF. DILEEP KUMAR SAHU

Alpha-Beta Pruning

- Alpha-beta pruning is a modified version of the minimax algorithm. It is an optimization technique for the minimax algorithm.
- As we have seen in the minimax search algorithm that the number of game states it has to examine are exponential in depth of the tree.
- Since we cannot eliminate the exponent, but we can cut it to half.
- Hence there is a technique by which without checking each node of the game tree we can compute the correct minimax decision, and this technique is called **pruning**.
- This involves two threshold parameter Alpha and beta for future expansion, so it is called **alpha-beta pruning**.
- It is also called as **Alpha-Beta Algorithm**.
- Alpha-beta pruning can be applied at any depth of a tree, and sometimes it not only prune the tree leaves but also entire sub-tree.

Alpha-Beta Pruning

- The two-parameter can be defined as:
  - **Alpha:** The best (highest-value) choice we have found so far at any point along the path of Maximizer. The initial value of alpha is **-∞**.
  - **Beta:** The best (lowest-value) choice we have found so far at any point along the path of Minimizer. The initial value of beta is **+∞**.
- The Alpha-beta pruning to a standard minimax algorithm returns the same move as the standard algorithm does, but it removes all the nodes which are not really affecting the final decision but making algorithm slow.
- Hence by pruning these nodes, it makes the algorithm fast.

Alpha-Beta Pruning

- Condition for Alpha-beta pruning:

  $\alpha >= \beta$

NOTE:

- The Max player will only update the value of alpha.
- The Min player will only update the value of beta.
- While backtracking the tree, the node values will be passed to upper nodes instead of values of alpha and beta.
- We will only pass the alpha, beta values to the child nodes.

Pseudo-code for Alpha-beta Pruning

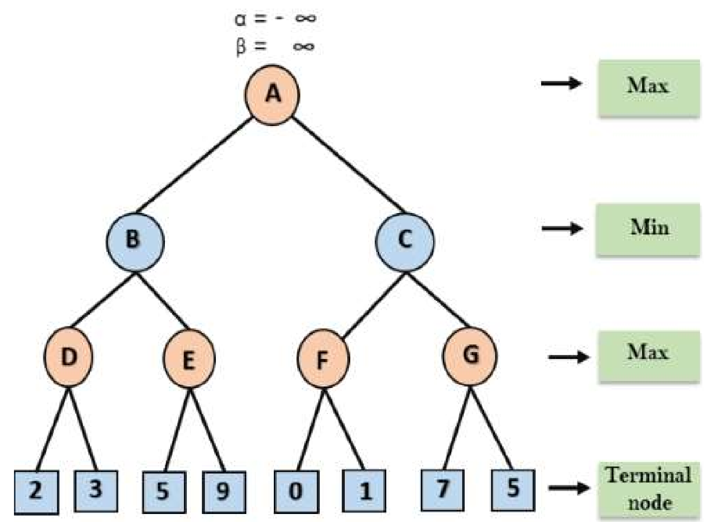
```
function minimax(node, depth, alpha, beta, maximizingPlayer) is
if depth ==0 or node is a terminal node then
return static evaluation of node

if MaximizingPlayer then      // for Maximizer Player
  maxEva= -infinity
  for each child of node do
      eva= minimax(child, depth-1, alpha, beta, False)
      maxEva= max(maxEva, eva)
      alpha= max(alpha, maxEva)
      If beta<=alpha
        break
  return maxEva

else                     // for Minimizer player
    minEva= +infinity
       for each child of node do
           eva= minimax(child, depth-1, alpha, beta, true)
           minEva= min(minEva, eva)
           beta= min(beta, eva)
            if beta<=alpha
              break
         return minEva
```

Working of Alpha-Beta Pruning

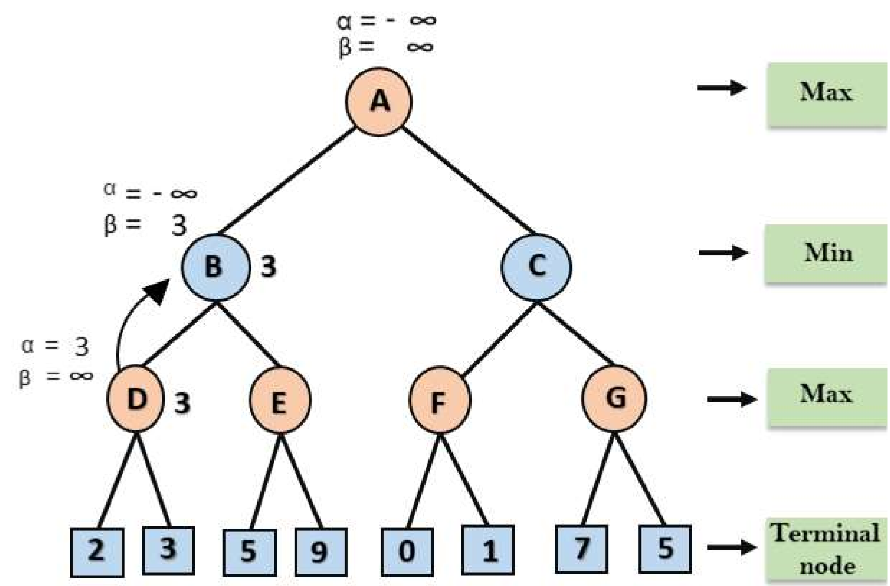
- Let's take an example of two-player search tree to understand the working of Alpha-beta pruning:
- **Step 1:** At the first step the, Max player will start first move from node A where α= -∞ and β= +∞, these value of alpha and beta passed down to node B where again α= -∞ and β= +∞, and Node B passes the same value to its child D



Working of Alpha-Beta Pruning

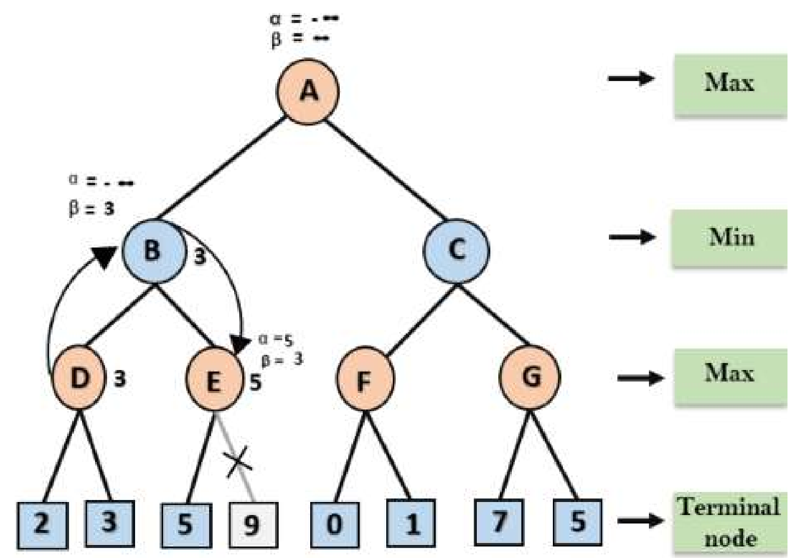
- **Step 2:** At Node D, the value of α will be calculated as its turn for Max. The value of α is compared with firstly 2 and then 3, and the max (2, 3) = 3 will be the value of α at node D and node value will also 3.

- **Step 3:** Now algorithm backtrack to node B, where the value of β will change as this is a turn of Min, Now β= +∞, will compare with the available subsequent nodes value, i.e. min (∞, 3) = 3, hence at node B now α= -∞, and β= 3



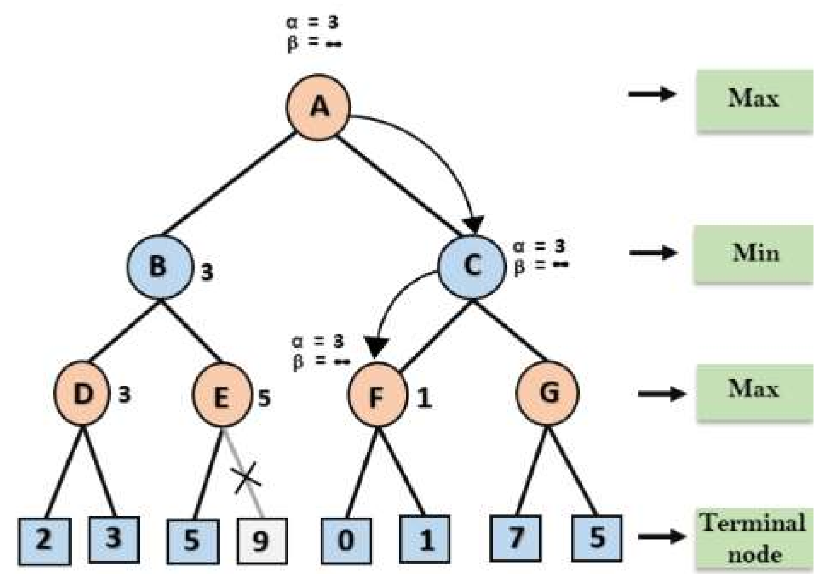
Working of Alpha-Beta Pruning

- In the next step, algorithm traverse the next successor of Node B which is node E, and the values of α= -∞, and β= 3 will also be passed.
- **Step 4:** At node E, Max will take its turn, and the value of alpha will change. The current value of alpha will be compared with 5, so max (-∞, 5) = 5, hence at node E α= 5 and β= 3, where α>=β, so the right successor of E will be pruned, and algorithm will not traverse it, and the value at node E will be 5.



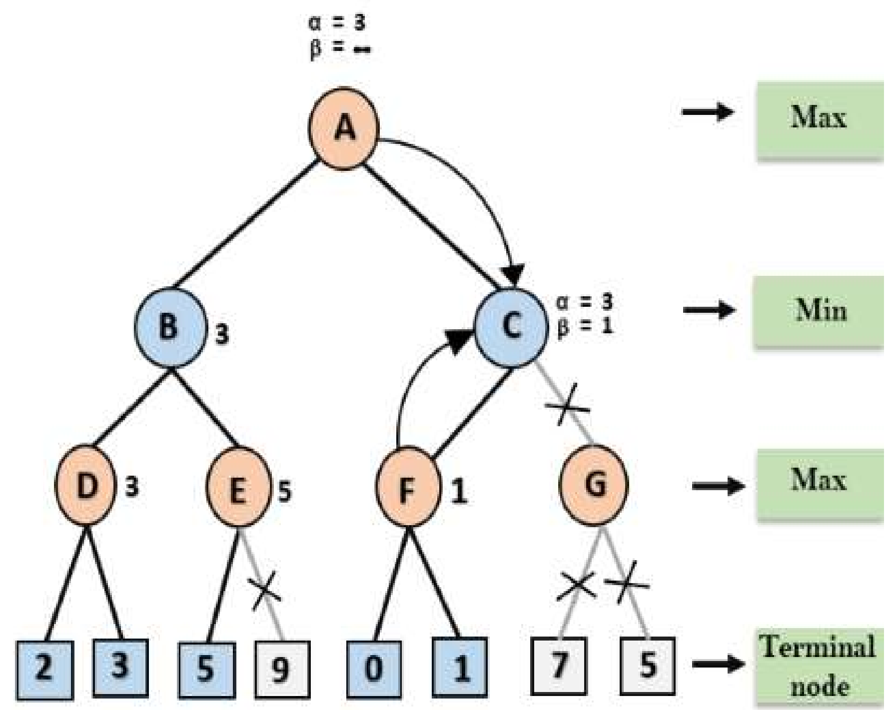
Working of Alpha-Beta Pruning

- **Step 5:** At next step, algorithm again backtrack the tree, from node B to node A. At node A, the value of alpha will be changed the maximum available value is 3 as max (-∞, 3)= 3, and β= +∞, these two values now passes to right successor of A which is Node C.
- At node C, α=3 and β= +∞, and the same values will be passed on to node F.
- **Step 6:** At node F, again the value of α will be compared with left child which is 0, and max(3,0)= 3, and then compared with right child which is 1, and max(3,1)= 3 still α remains 3, but the node value of F will become 1.



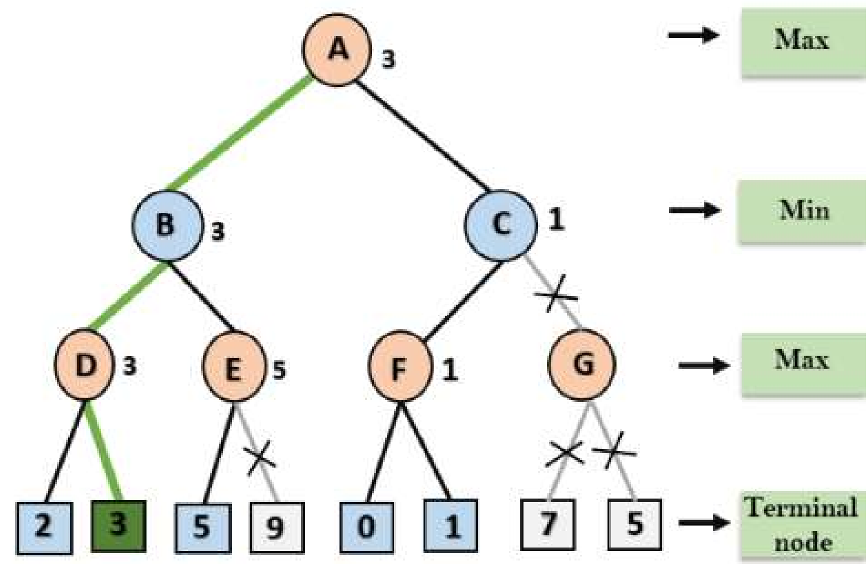
Working of Alpha-Beta Pruning

- **Step 7:** Node F returns the node value 1 to node C, at C α= 3 and β= +∞, here the value of beta will be changed, it will compare with 1 so min (∞, 1) = 1.
- Now at C, α=3 and β= 1, and again it satisfies the condition α>=β, so the next child of C which is G will be pruned, and the algorithm will not compute the entire sub-tree G.



Working of Alpha-Beta Pruning

- **Step 8:** C now returns the value of 1 to A here the best value for A is max (3, 1) = 3.
- Following is the final game tree which is the showing the nodes which are computed and nodes which has never computed.
- Hence the optimal value for the maximizer is 3 for this example.



Move Ordering in Alpha-Beta pruning

- The effectiveness of alpha-beta pruning is highly dependent on the order in which each node is examined.
- Move order is an important aspect of alpha-beta pruning.

It can be of two types:

- **Worst ordering:** In some cases, alpha-beta pruning algorithm does not prune any of the leaves of the tree, and works exactly as minimax algorithm.
  - In this case, it also consumes more time because of alpha-beta factors, such a move of pruning is called worst ordering.
  - In this case, the best move occurs on the right side of the tree. The time complexity for such an order is $O(b^m)$.
- **Ideal ordering:** The ideal ordering for alpha-beta pruning occurs when lots of pruning happens in the tree, and best moves occur at the left side of the tree.
  - We apply DFS hence it first search left of the tree and go deep twice as minimax algorithm in the same amount of time.
  - Complexity in ideal ordering is $O(b^{m/2})$.

Rules to find good ordering

Following are some rules to find good ordering in alpha-beta pruning:

- Occur the best move from the lowest node.
- Order the nodes in the tree such that the best nodes are checked first.
- Use domain knowledge while finding the best move.
  - Ex: for Chess, try order: captures first, then threats, then forward moves, backward moves.
- We can keep the states, as there is a possibility that states may repeat.

References

- https://www.javatpoint.com/ai-alpha-beta-pruning
- . Elaine Rich and Kevin Knight: Artificial Intelligence- Tata McGraw Hill.

Thank You