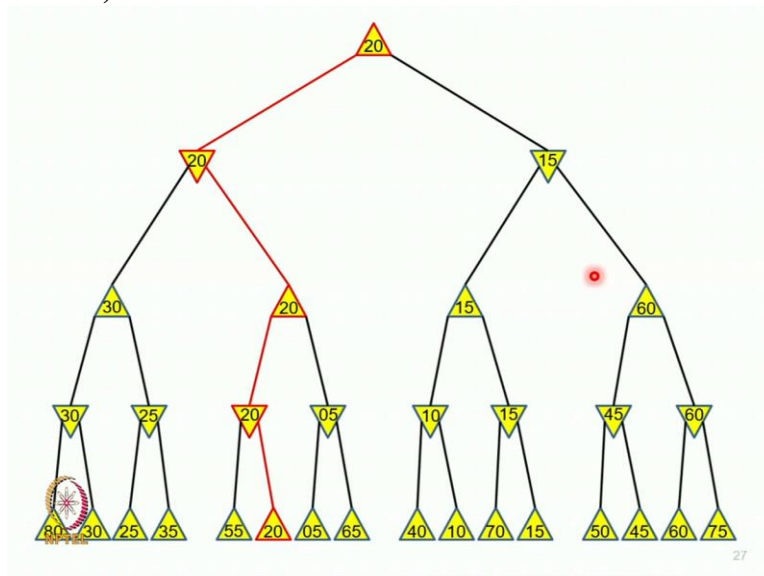


**Artificial Intelligence**  
**Prof. Mausam**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology Delhi**

**Lecture-30**  
**Adversarial Search: An Example of Mini max Search-Part-2**

Let us look at another example just to make sure that we understand this together, and this is my game tree.

(Refer Slide Time: 00:25)



I start in my adversary place. I will in that case of searching I am going to try left first. And at the end I get to these leaf nodes and these leaf nodes give me some utility function and I would like everyone to help me with filling up this. So the very first node I reach when I start doing these searches is? So, I start going left and finally I hit 80. So, I reach 80. This is the leaf so I back it up; I send this information to the parent.

So the min node, which is the parent of 80, learns that there is a value of 80 below me. So it says 80 then it goes to the other child. And the other child says 30 now as soon as the other child says 30 what becomes the parents value 30 because it is a min old, very good. So it becomes 30 and now it saves the information up to its parent which is a max node saying, Look in this sub tree I; the best you the parent can achieve is 30.

Now the parent says the max, in the first child, I can get 30 what about the 2nd child? So it sends this down and you know, you go back start backing it up so you get 25 then you get a 35 what is a new value of the min? 25 because it is a min 35 is worst it does not get changed and so, this 25 then gets sent up the previous value of the max was 30 the new value of the max is 30 because 25 is worst and this information goes up which is the min.

Everybody with me this is simple stuff, but notices what is this 30 says? So, let us look at this particular 30, this particular 30 says that if I get to this node, the best we together can achieve me and my adversary playing as best as we can is 30 because I will be playing this left move and my adversary will be playing the right move. Of course this goes on. So, the min node is 30 it says, I can go left and I can get 30 but I can also possibly go right and see what do I get then so it says 55, then the left sub tree gives me 20.

So I have 20. Similarly, I go to the right sub tree it gets 5; it sends the information that, it is 5. And then it goes down again. So here is a question for you. Think about 1 or 2 places where you are doing extra computation. I do not have I do not need an answer right now. We will come to that. But as you are looking at this diagram, see if you can find some places where we are doing unnecessary computation.

If you can figure it out, we can quickly discover the next algorithm we want to learn. But I will just keep going. So it gets to 5. It sends its neighbor its parent information that you know, I met 5, the 5 goes down again it gets 65. It is a min node. It remains 5. It remains 5 it sends this information up. It is a max node it remains 20, the information up that is a min node. So when 20 goes up, it becomes 20.

And so this information goes up to the max node, the root node. And so the root node says that if I take the left action, the best we can achieve is 20. Now, let me see what I can do with the right action. And so this process continues. So we get with the right action, this kind of information. So the right sub tree says, you know, I can best get 15 because it is a max node, the final solution is 20. And now, I have solved this mini max tree. And in fact, I have found the optimal play.

What is the optimal play? I will take left. Adversary will take right. I will take left adversary will take right this is the optimal play. So in this way, if we can search till the end of the tree, and if we know exactly what the utility function at the end, then we can back it up at every point doing max min as appropriate to give me not only the value of each play, but also the optimal play. There is a question or the answer to the question that I asked earlier. Yes, question.

In this case, are we assuming that always max finishes the game? No, we are not assuming that. So at the leaf I have drawn triangles, but I should have ideally drawn squares. So this is just the end of the game. In this sub tree of leaf is at the same level, but it does not have to be. I should point out that leaves are neither max nor min it is important to understand. I drew them as max and I can double, I mean change the slides. But the end nodes, the leaf nodes are just the end of the game.

They are just sending out the utility function, they are neither doing a max nor doing the min, they are just telling you the final value. And secondly, they need not be at the same level, you know, this is the kind of things I could not fit on a slide, but you can imagine that for example, this 60 is a leaf then only the 60 will be backed up. The 2 children need not go to the same depth and moreover, since the number of steps does not have any negative reward or penalty or anything like that.

It does not matter how many steps it takes, the only thing that matters is whether it finishes and when it finishes, you know what value do we get?

**(Refer Slide Time: 06:50)**

## Minimax Strategy

- Why do we take the **min** value every other level of the tree?
- These nodes represent the **opponent's** choice of move.
- The computer assumes that the human will choose that move that is of **least value** to the computer.



So, I have already explained this is called the min max strategy. We take the min value for every other level and max value for every other level depending upon adversary of your computer. The nodes represents the min nodes represent the opponent choice, the max node represents my choice computers use, and the human will choose the move that is of the least value to the computer. Because we are playing more or less what is called a 0 sum game.

I do not know if you have heard this term zerosum game in game theory is a zerosum game basically means if I win, you lose. If I win 2 you lose 2. If I win 100, you lose 100. If I lose 10, you win 10. So basically, the sum of us is 0. And if it is a 0 sum game, only one of us can win is life a 0 sum game. Otherwise, you will never make you do projects in teams. In pairs, there are 3 kinds of multi agent settings which I will not talk too much about. But there are 3 kinds of settings.

One is called the adversarial setting, which is where we are where one wins, the other loses or if it is a team of players playing one win, then everybody else loses. Then they have what is called cooperative games. Cooperative games would be you know a set of robots which are trying to do a rescue operation together. They are not saying that you know, I should rescue a human and I will not let the other robot rescue a human.

Then we are all in the same thing, it is our collective effort together we need to win. Together we need to rescue all humans. This is called a cooperative multi agent setting. And then, in life, we as people are the adversarial or cooperative. It is a mix of both Viswajith says, Viswajith knows that sometimes you can be cooperative and sometimes we cannot be that political parties are usually not cooperative. But Indian cricket team maybe depends. So we are neither cooperative nor adversarial.

We as people are what are called self interested agents. We are interested in us and us only. And anything we do, we do to optimize our utility function. We are selfish. This is my belief, but more people agree with me then. I also believe that if I give it up to the beggar I do not do it for their happiness. I do it for my happiness, which comes out of the satisfaction of helping a poor person. If I do anything for my family, this is a very sad way of thinking about life.

I apologize. But we are self interested agents. If I do something for my family, it is for my long term happiness. You know, I need to make sure my wife remains happy so that I remain happy. You may not be able to sympathize with me now. So, we are what is called self interested agents in a self interested agent. Even in multi agent setting, what happens is suddenly people come together in the context of a sub goal, because this helps both agents.

But sometimes we may end up being adversarial to each other because our goals do not align. So people study the art of negotiation, people study, teaming people study cooperation and forming teams and acting against teams, etc. in the context of this sub area of AI called multi agent systems but we are not going to study this in the class today, so I thought I will just give you not today in the course. So I thought I will give you at least a feeling of what a multi agent system.

Sort of is, but we are going to study the adversarial 2 player agent system and we have already studied the min max algorithm now question for you. Is min max algorithm closer to depth first search breadth first search or iterative deepening search how many people think it is closer to breadth first search? You have to raise one hand at least. Again nobody thinks so. How many people think it is closer to iterative deepening search one person? One that is it one and a half,

how many people think it is similar to depth first search? Almost everybody very good so at least you are falling something in the class I am happy. So yes, it is like a depth first search analog.

(Refer Slide Time: 11:35)

## Minimax algorithm Adversarial analogue of DFS

```
function MINIMAX-DECISION(state) returns an action
   $v \leftarrow \text{MAX-VALUE}(\textit{state})$ 
  return the action in  $\text{SUCCESSORS}(\textit{state})$  with value  $v$ 

function MAX-VALUE(state) returns a utility value
  if  $\text{TERMINAL-TEST}(\textit{state})$  then return  $\text{UTILITY}(\textit{state})$ 
   $v \leftarrow -\infty$ 
  for  $a, s$  in  $\text{SUCCESSORS}(\textit{state})$  do
     $v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(s))$ 
  return  $v$ 

function MIN-VALUE(state) returns a utility value
  if  $\text{TERMINAL-TEST}(\textit{state})$  then return  $\text{UTILITY}(\textit{state})$ 
   $v \leftarrow \infty$ 
  for  $a, s$  in  $\text{SUCCESSORS}(\textit{state})$  do
     $v \leftarrow \text{MIN}(v, \text{MAX-VALUE}(s))$ 
  return  $v$ 
```



But in the adversarial setting why? Because there is no memory frontier that we are maintaining we are going down one sub tree evaluating it completely and then backtracking and going to the other sub trees. So it is like that first search. But of course we are not iteratively increasing our depth. So it is not iterative searching. So this is the Vennila version. So as you can do more things and you can also create a iterative deepening but we will not go there for now.

So, it is like depth first search and for depth first search, you know, you can ask the following questions and by the way, you can read the scored it basically says that I have to take a mini max decision and the mini max decision would return the action which is the max value and in the max value, I will look at all successors and I will find the minimum valued successes and send that information out.

So, I will find the max minimum valued successor and for min, I will find the min maximum valued successor. So, this would be my sort of game that for if I max for all my children, I will ask them to do the min and then there whatever min they have I will pick the max and recursively. So, this is a mini max.

(Refer Slide Time: 12:52)

## Properties of Minimax

- Complete?
  - Yes (if tree is finite)
- Optimal?
  - Yes (against an optimal opponent)
  - No (does not exploit opponent weakness against suboptimal opponent)
- Time complexity?
  - $O(b^m)$
- Space complexity?
  - $O(bm)$  (depth-first exploration)



And now you can ask the question is it complete optimal time complexity and space complexity in terms of BDM as we have been studying? reminder B is branching factor M is the maximum depth of the tree there is no such there is actually no D there is no goal. So, is it complete when is depth first search complete? When tree is finite so, if tree is finite, it is complete is it optimal, you will finally get to the optimal value, but this will be optimal against an optimal adversary.

This is very important, because it is possible that I have an adversary fall which I know a weakness and if I know a weakness, I know that in at a point where they can take many actions, they may not take the best action. And if I know that, then I can exploit that weakness and I can get even more score, but that we are the mini max algorithm does not give you it does not exploit opponent weakness against suboptimal opponent.

And whenever you create a game tree, a game playing engine? Like in an assignment, for example, you are allowed to do some opponent learning, you are allowed to think about, are this opponent is this opponent is making some kind of mistakes? This is a very aggressive opponent is this a very conservative opponent? Should my strategy change against such an open and you can do this.



But this algorithm tries to come up with the optimal flip play for you and the adversely. And time complexity is going to be  $b$  to the power  $m$  because I am going to search the whole tree till


the very end. And space complexity is going to be  $b^m$  similar to depth research because I am sort of doing that depth first exploration. Now, let us think about whether mini max is a good algorithm.

(Refer Slide Time: 14:53)

### Good Enough?

- Chess:
  - branching factor  $b \approx 35$
  - game length  $m \approx 100$
  - search space  $b^m \approx 35^{100} \approx 10^{154}$
- The Universe:
  - number of atoms  $\approx 10^{78}$
  - age  $\approx 10^{18}$  seconds
  - $10^8$  moves/sec  $\times 10^{78} \times 10^{18} = 10^{104}$

  Exact solution completely infeasible



So let us say we take chess branching factor of chess is about 35. A typical game is about 100 length. That means the search space is about 35 to the 100, which basically means about 10 to the 154. The universe has 10 to the power 78 atoms and the universe's age is 10 to the power 18 seconds. And I do not know where this comes from, but let us say the universe is able to evaluate when 10 to the power 18 moves per second or 10 to the power 18 expansions per second.

Even then, it will not be able to explore more than a size of 10 to the power 104. So, exact solution is completely infeasible. Not only can you and I but even the universe cannot figure it out. Right from Big Bang up to this point, even if it was figuring out are using all its atoms to learn to play chess, and each atom was a different game move I mean game configuration, it would have still not completed the game all possible games.

So clearly, for all practical purposes mini max is not a helpful algorithm, because we will not be able to play it till the very end. So we need to come up with tricks of the trade so that we can search to limited depth and still figure out a good solution. So, we are going to do 2 main ideas. One idea is not searching things, which are provably suboptimal and second idea is cutting off the search and doing some machine learning to help with utility function estimation. Those are



the 2 things we are going to do. In today's class, we will do one of them and this is where I would want everybody's attention. Because if we get this is going to be awesome.