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

Lecture – 19
Informed Search: Iterative Deepening A*and Depth First Branch & Bound (Part – 5)

Alright now A star algorithm is like the any algorithm of best first algorithm and therefore the number of nodes that you have to keep in memory is fairly high. In the worst case there are explanation. And we have discussed in the last class that exponential space is not something we really like we want to avoid it and remember we went to iterative deepening search in order to deal with the memory problem.

So the next order of business is can we come up with the iterative deepening version of A star algorithm? So what is the intuition? The intuition is that the A star algorithm expands all nodes with a lower F value before it gets to the higher F value that is the high level intuition right. Just like breadth first search would expand all nodes of a certain depth before going to the next depth. And just like uniform cost search will expand all nodes of a certain G before it goes to the next node.

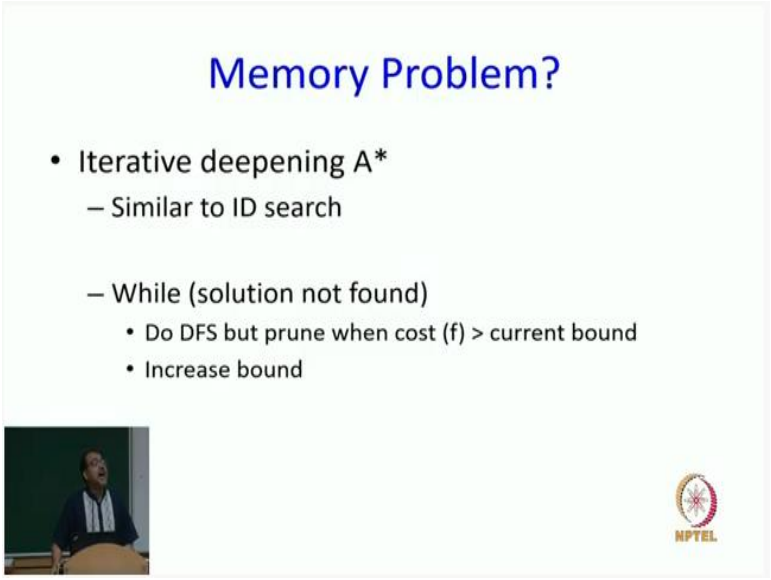
So there all somehow incorporating the intuition that I have to exhaust all nodes up to a certain bound but that bound depends on what am I working on. It could be depth it could be costs. So far it could be total cost and so on and so forth. So that gives you a clue. So first of all if you want to solve the memory problem will ever algorithm look like breadth first search or depth first search come on depth first search.

Because iterative deepening search is essentially depth first search but with a cut off bound. But iterative deepening search had a cut off bound on max depth and that is why we called it depth limited search. But max depth worked there because breadth search goes in depth fashion. But now max depth is not going to work for us. So what is going to work for us? Max.

"Professor - student conversation starts" What is going to work for us? Somebody raised their hands. You have an answer please raise. What is your name? Shreya, max F value right. Because

I am expanding nodes intuitively in F order let me expand all nodes up to a certain max F value and then increase the bound "**Professor - student conversation ends"**.

(Refer Slide Time: 03:10)

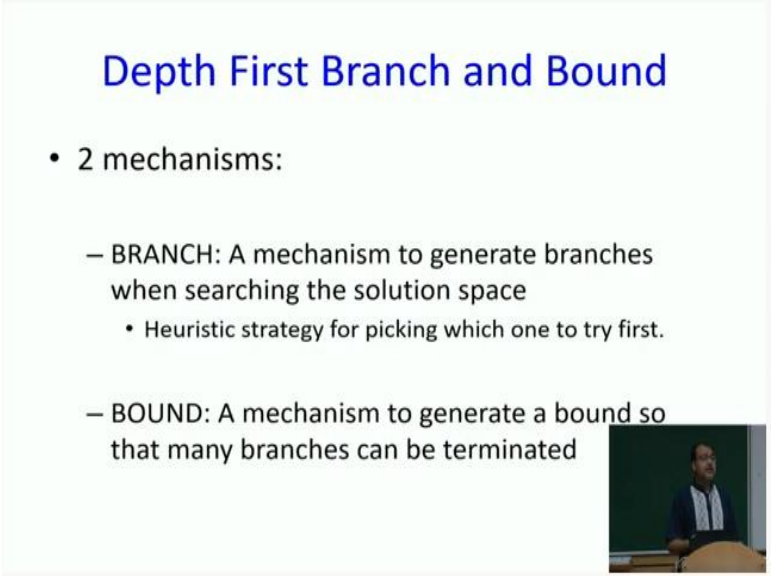


And that is exactly what is called the iterative deepening A star algorithm. Basically says just do depth first search. But as long as my F increases as a certain bound stop backtrack. If I find goal great if I do not find goal increase my F bound and that is called iterative deepening A star. So for real problems that we encounter we often do not implement A star. We often implement iterative deepening A star **"Professor - student conversation starts"**

Questions on this okay. Yes. So it will have much better space. Yes, yeah. So the question is what are strategies to increase the bound? I am keeping this open sometimes those strategies are determined by the problem of it. I mean the least you can do is increase it with the least cost right. So that should mostly work. But you can do probably better if you want right. So that will be problem dependent "**Professor - student conversation ends"**.

Okay now ID A star is 1 algorithm but I want to do another algorithm. This is not often. This is not typically done in an AI class but I like this algorithm and this becomes important as a basis for game playing later and so I would like to do this algorithm called depth first branch and bound. It is possible that you did some other course somewhere where you did a branch and bound algorithm. It is the same thing okay.

(Refer Slide Time: 04:47)

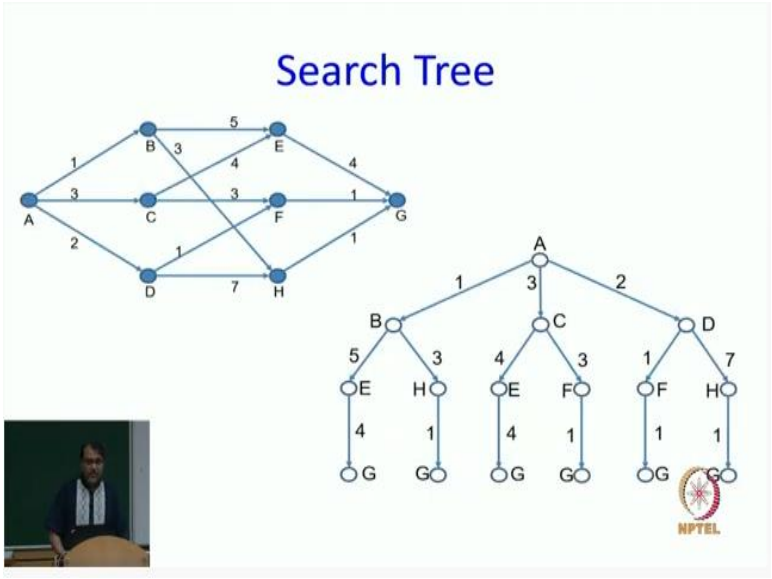


So branch and bound algorithm is also depth first search. But it has 2 mechanisms one is called the branching mechanism, one is called the bounding mechanism. A branching mechanism says which of my children are better? Let me go there. So you can think of it as depth first search with some preference on which node to visit first which node to visit later. It is okay. This is not a fundamental difference.

You can add this branching mechanism in most other depth first search algorithms. It is a heuristic strategy. So branching mechanism in depth first branch and bound is sort of like a heuristic strategy which child do I go first? However, the bounding mechanism is the strength of branch and bound and bounding mechanism says that I will be able to terminate branches if they exceed a certain bound.

So let us see if we can understand this and we will understand this in the context of this particular example. So this is an example of a graph.

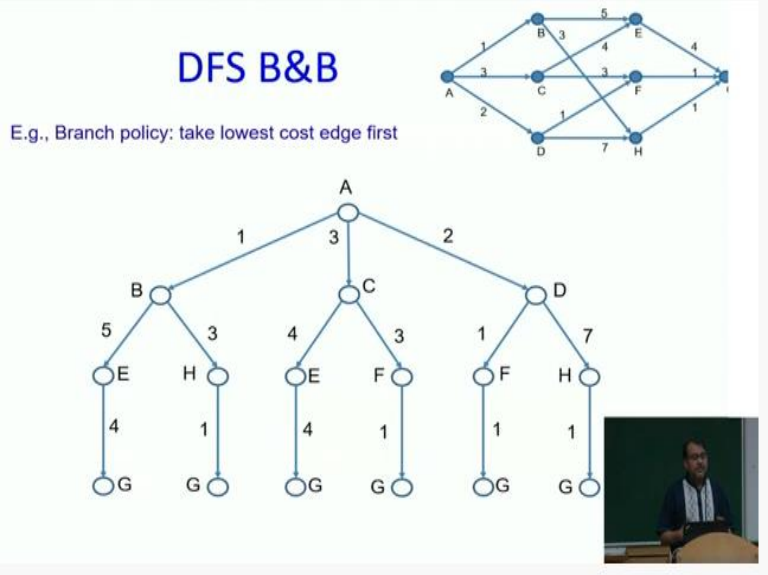
(Refer Slide Time: 05:52)



Let us say we are in the tree search paradigm for now. We are going to go from A to G. We want to find the optimal solution. Can you see the optimal solution? What is the optimal solution? B What is the next node? B sorry A it starts with A of course what am I doing? A, B, H, G, A, D, F, G correct. Alpha, delta, friend and goal A, D, F, G 2+1+1 4. This is what we are looking for okay.

And of course we can add IDA star if you want. But for now let us do depth first search branch and bound as an alternative mechanism. So this would be the search tree that we would be expanding. We would go from A to BCD, from B to EX, from C to EF, from D to FH and so on. And let us talk about the branching mechanism here. So let us take you can have any branching mechanism.

(Refer Slide Time: 07:09)



Let us take a simple branching mechanism of lowest cost edge first. Now this is just an example. We can take a different branching mechanism. But in this example we are going to go to the lowest cost edge first. And let us say we have the heuristic function as the lowest cost edge coming out of this node. So heuristic of B would be 3 because we will at least have to pay 3 to reach the goal. Heuristic of C would be 3, heuristic of D would be 1 okay.

So now let us start exploring it using depth first search with the branching strategy. Now initially I do not know what is the cost of the solution. So I can say that the cost of the solution can be infinite. My bound my upper bound is just infinite. I do not know. It can be really bad. I will maintain this global upper bound. This global upper bound initially is infinite because I have not found a single path to a goal.

However, let us start executing. So let us say I use my branching mechanism. So from A I go to you have to do this with me. If you do not want to do this, then that means you are not following. So you have to help me. So my branching strategy is go to the lowest take the lowest cost edge. So from A I go to B, from B I go to H, from H I go to yes I found a path to the goal. Now this may not happen initially I may have to backtrack etcetera.

But they will come a point where I will find a path to the goal. Is this optimal path? No reason for it to be optimal. Why? Because of a branching strategies heuristic. But as soon as I find a

path to the goal a path to a goal I can look at the cost. The cost here is 5 and I can update my upper bound. Now initially I taught my cost to the goal is less than equal to infinity. As soon as they find a solution my cost to the goal the best cost is <=5. My upper bound has become 5.

This may be the optimal, we do not know or we may find a better solution. But we know that the better solution will always be <5 if it is better. If it is as good it maybe 5. If it is worse, it will may be >5. So I have reduce my upper bound to 5. Now what am I going to do is that as I go to my nodes I am going to also maintain a lower bound okay. So once I backtrack to B what can I say? Well I cannot say much right now. My cost to B so far is 1 right.

And my best cost later is 3. So my I cannot prune away anything from B at this point. But as soon as I take this edge my cost has become 6+ the heuristic of E which is 4. So I know that whatever those solution. I am going to get from E it will at least be length 10 and if it is at least lengths 10 do I need to explore it any further? No because I may find goal, I may not find goal, but if I find goad it will at least be 10 and I have already found a path of length 5. I do not care for what happens in the future and I backtrack.

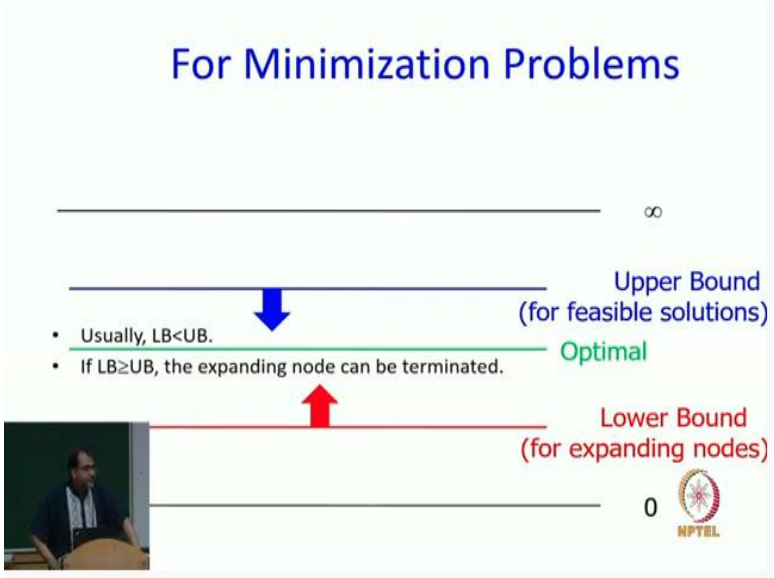
As I backtrack I come back to A and I want to assess D next. Now can I prove that I will not find a better solution through D? What is my lower bound here? It is 2+1 because my heuristic is only the lowest cost going forward. So my bound is only 3. I cannot prove anything. So I have to go to D. So I go to D. Now from here I my branching strategy says check F. The bound for F is 3+1 4. So I cannot say anything. I have to keep going.

And when I go to G, I have found a new path to the goal and in fact I have found a better path to the goal and because I found a better path I am going to reduce my upper bound. So I have now made my upper bound 4. Now as I come back when I try to go from D to H my cost becomes 9+1 and now I can prune it so I can get it off H. What is even more interesting is I can get rid of full sub trees.

So if you think about C what can I say is my lower bound of going through C 3+3 6 and 6 is already greater than 4. So there is no way I am going to get the best solution through C. So I

prune it. I do not even visit anything beyond C and therefore I have reduced some of the competition that I have to do. And this algorithm is called depth for search branch and bound.

(Refer Slide Time: 13:01)



Intuitively what is happening is I maintain a global upper bound and I maintained the lower bound for my expanding nodes whatever I am expanding so far. The upper bound has come from a feasible solution, a solution that I have already found. It not be the optimal solution and lower bound is I am looking for a better solution and usually lower bound is going to be less than upper bound.

But whenever lower bound is >= upper bound I can stop I do not need to go further. I have pruned anything else that happens in my sub tree I can backtrack. Everybody okay with the algorithm. Now comes a difficult question. What about depth first search branch and bound versus IDA star. Now first of all are they optimal. Let us say I am talking only about heuristics which are admissible.

(Refer Slide Time: 14:01)

So if my heuristics are admissible, are they optimal? See whenever I am pruning anything? Am I going to miss the solution at all? No I am not never going to miss a solution if I am pruning something. So both of them are optimal if they finish right. It is possible the depth first search never finishes and we will talk about that in a minute. However, and both of them have what about memory properties? Memory is not explanation necessarily right.

Because I am doing depth first search in both situations my memory never becomes exponential. Look you guys need to be able to do this analysis yourself. I have a cost bound and IDA star. I done everything to depth first search pruning nodes and at some point I stop. My memory never becomes exponential because I am only maintaining the path so far. Similarly, in depth first search branch and bound I am only doing depth first search.

So my memory is going to be no worse than the memory of depth first search. So I am never give an exponential memory regime. So this is good. Because I do not like exponential memory. However, I claim that one algorithm is systematic and one algorithm is not systematic. Which algorithm is systematic? Branch and bound because it is doing depth first search whereas IDA star has a bound. So whenever it increases its bound it needs to restart so it will visit the same node twice right.

So IDA star is not systematic. At the same time, it never expands a where f is > optimal cost. IDA star is guaranteed to never expand a node beyond the bound and whenever I hit the optimal cost bound that is when I will find the goal. But I will never expand any node where f > optimal cost. Is the same property 2 for depth for search branch and bound? No it is not. It is systematic but it may expand a lot of terrible nodes.

Because my initial upper bound can be infinite can be terrible. So it initially it may expand a lot of just random nodes before it hits a solution. Yes, it should do. So if there is an infinite depth graph, which algorithm is better? IDA star is better if there is an infinite depth graph because it is possible that I go into some breadth hole again and never come back. Very good so this is when IDA star will be better **"Professor - student conversation starts".**

When would branch and bound be better? No that is not true. The suggestion is, what is your name? Parth says when you want only 1 solution, only 1 feasible solution we do not care for optimality. That is not true we are let us say looking for optimality in both the cases. But when would depth first search branch and bound be a better algorithm? Sorry very good what is your name? Divya says when I have a better estimate of an upper bound sure very good.

But initially I do not know anything. So initially unless I have some domain knowledge from where I am getting an estimate of the upper bound. Initially we can say without doing anything it is infinite. What else? It is a correct answer. But it requires additional domain knowledge. What else? Say that again when the paths to the goal have less deviation from the optimal. So if the cost of many different paths to the goal is not too different from the optimal then this would be better okay.

But this what are they implicitly saying? If fanon on the goal is very high okay that is a very specific property. You are not you are both saying various ways of the same thing. If I have many different paths to the goal and they are easy to find in terms of like there are no infinite depths and so on and so forth. Then breadth first search will branch and bound do really well "**Professor - student conversation ends"**.

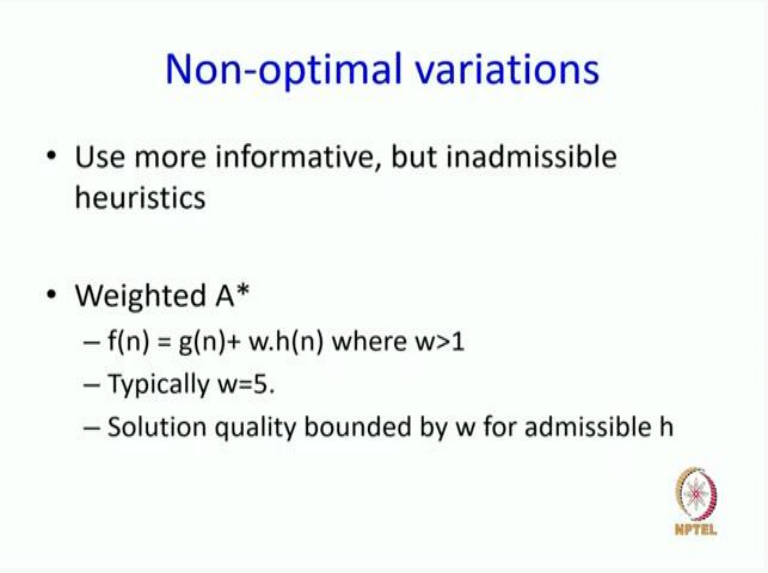
So in other words if it was easy to find a suboptimal solution, then branch and bound would be very good because you found 1 suboptimal solution that reduces your bound. Then you can prune a lot of nodes and once you start pruning you will search much faster. But if your first solution was very hard to find then life is tricky for branch and bound. But it is okay for IDA star. Because IDA star when it finds the goal that is when it stops whereas branch and bound finds many goals before it say I am done.

So it is a different way of looking at life. One of them is going in F order, one of them is going somewhere randomly with the branching strategy. Once it finds a goal it feels like oh, now I can do a much better search whereas IDA star has looked at all other possible ways to get to the goal. Sorry not possible ways to get to the goal. But a sort of proven that once I remove the first goal I will be optimal.

So I have looked at all possible estimates of different paths to the goal. But I am not reached the goal right. So this is a difference in their strategies. If the search tree is bound and depth branch and bound might do all right. If it is an infinite search tree IDA star might do much better. If it is easy to find a suboptimal solution branch and bound might do better. If it is difficult to construct a single solution IDA star might do better okay. Very good.

So I need to do only 1 algorithm further which is a very small simple algorithm. And then we will now move on to then move on to how to compute heuristics? So the one algorithm that we need to consider is called weighted A star.

(Refer Slide Time: 20:38)



So weighted A star says I am going back to the suggestion from this side, let us do alpha gn + beta hn. This fellow new weighted A star in their mind. So basically weighted A star says that instead of doing gn + hn, do gn + w times hn and w is called the weight in the weighted A star algorithm. And in fact do we want weight to be >1 or <1?

See what is the problem with A star algorithm? If you think about it from a practitioner perspective not in theoretician's perspective. From the practitioner's perspective it explores a lot of partial paths and stops exploring them only where it can prove that they will not lead to the optimal solution. But that ends up be doing a lot of extra exploration and you want to reduce it. If you remove gn from here, then it is a problem because greedy best first search is not even complete right.

So you want to emphasize the importance of a heuristic but you do not want to weigh them equally. You want to say heuristic is really important maybe more important. But gn is also important. So that is sort of what you are saying. So in practice what people do is they keep w to be >1 typically 5, 7, 10 things like that often 5. And if they can prove that if you are doing gn + w hn then the paths that you would find will be no worse than w times the optimal.

So if you use w = 5 and you find a path to the goal then it will be up within 5 times the optimal right. So you have a optimality bound there but in practice it finds excellent paths and it finds it

much faster. So it does not explore a lot of these side states and sort of goals more directly towards the goal. This algorithm is called the weighted A star algorithm okay.

So this is a good point to take a pause. We started out with the agenda that we want to develop an algorithm that can somehow add some intuition in it is full. I also said that intuition can be wrong so we cannot trust it completely. The algorithm their trust set completely is greedy best first search. It says heuristic order let us just go in the heuristic order and expand. But of course it you cannot trust it completely.

So we came up with the A star algorithm which says what happens in the future needs to be balanced with what has happened in the past. And if a lot of costs has already been paid in the past then this node should not look very good even if goal looks close and we devise the A star algorithm but of course it had bad memory properties in the worst case. And so we looked at memory efficient versions of A star.

So we talked about IDA star and we talked about depth first search branch and bound and then we said optimality is a gift. It is luxury. It is something that God has created but it is very hard to achieve. So let us let go of optimality. Yes, we want to find a path to the goal. We want to maintain completeness but optimality is too big a price to pay for computation. So let us do some suboptimal algorithms.

And we have so far done only 1 suboptimal algorithm and that algorithm is called weighted A star. And in this weighted A star we trust our intuition more and trust what has happened less but still they are balanced in some way balanced with the weight of w. And we even have an optimality bound here later we will do algorithms there we will not have any optimality bound whatsoever.

Now the key question that we need to answer. Is in general how do we find heuristics? Is there a general principle here or we just have to come take it out from our pocket somehow and say for this problem I can propose a heuristic which will be less than equal to the optimum and therefore it is a good heuristic we use in the A star algorithm? There is believe it or not a little bit of

understanding of how to create admissible heuristics which is what we are going to do in the next segment of the class. We can stop here. Thank you.