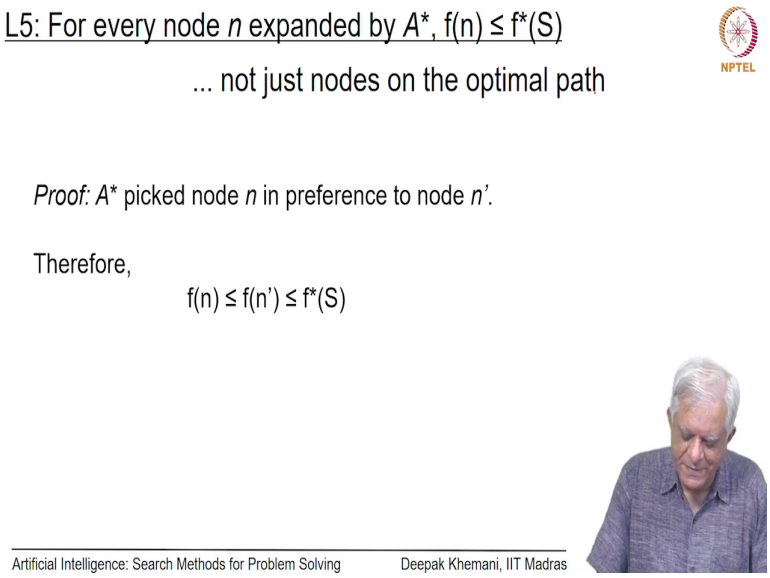
Artificial Intelligence: Search Methods for Problem Solving
Prof. Deepak Khemani
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Chapter – 05
A First Course in Artificial Intelligence
Lecture – 40
Finding Optimal Paths: Higher, Faster

(Refer Slide Time: 00:14)

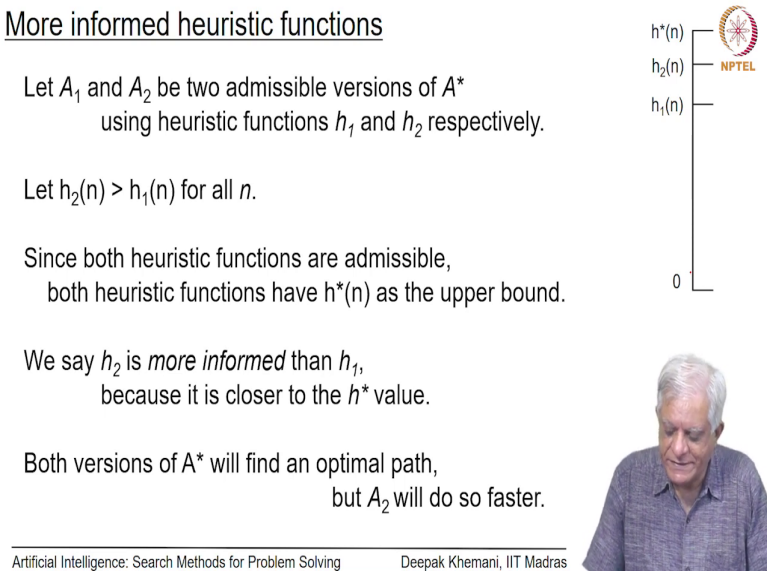


The next lemma that we want to prove is that for every node that A star picks or every node that is expanded by A star, its value f of n is going to be less than the value of the optimal cost path. So, it is not just for the nodes which are on optimal path, but A star only picks those nodes whose estimated cost of the final solution. Remember that f of n stands for a path from start to goal passing through the node n.

So, it will only pick such nodes, such that the estimated cost through those nodes is less than the optimal cost. In other words, it gives us an insight that it will not go off in the wrong direction too much essentially. What is the proof for this? The proof is very simple that A star has picked this node n when it could have jolly well picked the node n prime which we have shown always exist on the optimal path.

So, clearly the value of n can at most be equal to the value of n prime, because we have not stated that if two values are equal what is the criteria for choosing a node. So, f of n must be less than or equal to f of n prime. And in under such conditions f of n can be picked by the algorithm A star. But we have already shown in lemma 2 that f of n prime is less than f star of s which is the optimal cost. So, therefore, the estimated cost f of n for any node that a star picks must be less than the optimal cost.

(Refer Slide Time: 01:50)



Now, let us talk about different kinds of heuristic functions. So, once condition we have stated is that the heuristic function must always underestimate a distance to the goal. But different heuristic functions may have different levels of estimation.

So, let us discuss the impact of how search behavior is influenced by the value returned by the heuristic function. So, let us say that there are two heuristic functions. And they are being used by two versions of A 1 and A 2 versions of A star which we will call A 1 and A 2 and they use these two heuristic functions h 1 and h 2 respectively.

So, as shown in the diagram h 1 and h 2 must be underestimating functions if the algorithm is to be admissible. So, both h 1 and h 2 for any node n are less than h star of n which is the highest value that in any admissible search will allow. But we want to look at what is the effect if h 2 is always consistently greater than h 1.

Observe that the lowest possible value of a heuristic function is 0, and you can think of branch and bound and Dijkstra's algorithm as being instances of A star where the heuristic function returns always 0 which means only it looks at the path from start to end and assumes that going from end to goal is 0 cost. And then its behavior would be like Dijkstra's algorithm, or like branch and bound it would have no sense of direction.

If it has a heuristic function we want to see how does this sense of direction get influenced. So, clearly of function with a h of n, and we have shown that in lemma 5 that given the heuristic function any node that A star picks must have its estimated cost from start to goal passing through that node to be less than the optimal cost. So, what happens when h 2 is always larger than h 1?

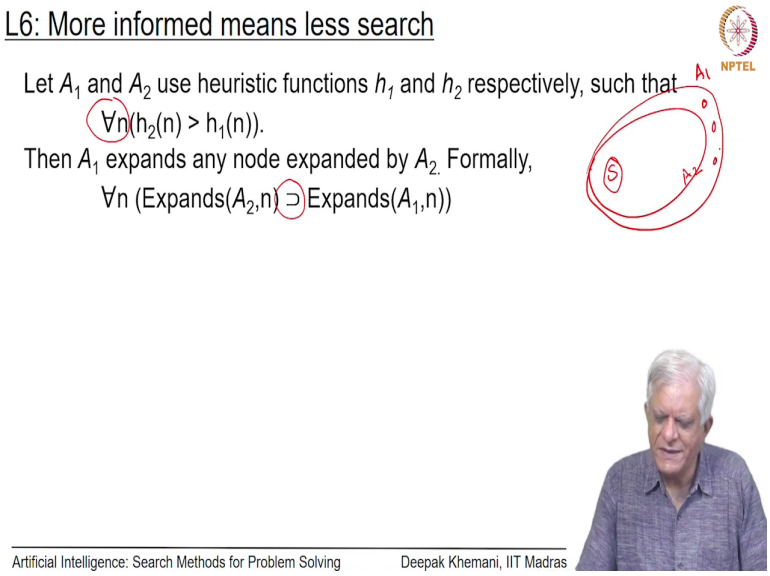
So, let h 2 of n be greater than h 1 of n for all n, this is just the property of the heuristic function that this is the case. Since both functions are admissible, both functions will have h star as the upper bound they cannot be larger than h star if they have to be admissible.

But we say that h 2 is more informed than h 1 because it is closer to the h star value. So, the higher the heuristic value, we say the more informed it is. Again clearly we can see that if h of n is 0 as is the case for branch and bound or Dijkstra's algorithm it does not use that estimated distance information at all.

What we are saying now is that as the h value increases if a function h 2 is always more than returns a value which is always more than a function h 1, then we say h 2 is more informed than h 1. Of course, both have to be less than the h star values essentially. So, both versions will find an optimal paths to the goal because they are less than they are under underestimating functions, but we want to show that A 2 will do so faster which means A 2 will search less of the space as possible.

Remember that I keep saying that if we had a perfect heuristic function, then you know the search would directly go off towards the optimal path. But in practice we do not have perfect heuristic functions and we can try to work with better and better functions. And what we are saying here is that if you have a more informed heuristic function, then the search would have to do less amount of work, and it will terminate faster. So, let us do a formal proof of this.

(Refer Slide Time: 05:53)



We want to show and this will be lemma 6 that more informed means less search. If the heuristic function is more informed, then the algorithm using that heuristic function will have to do less search. So, as we have said let A 1 and A 2 use heuristic functions h 1 and h 2 respectively such that for all n h 2 of n is greater than h 1 of n. So, I hope you know familiar with this notation which is comes from logic. It says it for every node n just a shorter way of writing that.

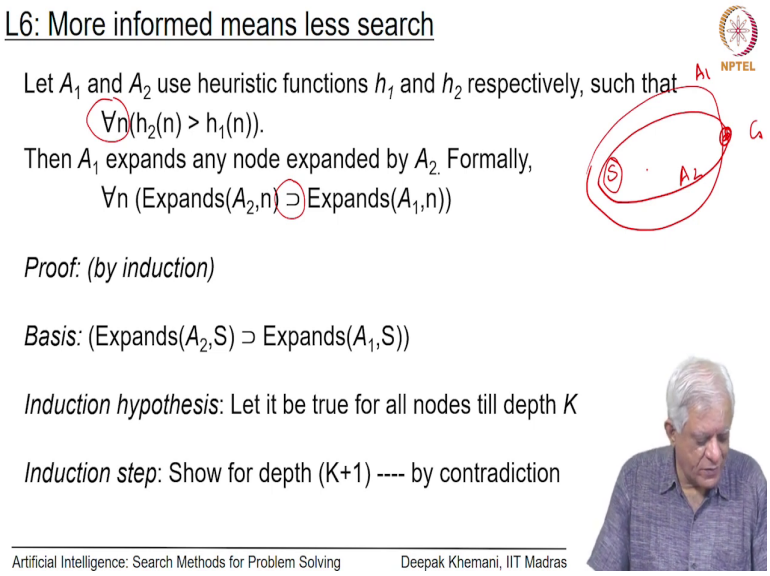
And for every node n h 2 of n is greater than h 1 of n. So, this means h 2 is more informed than h 1. Then we want to show that if A 2 expands any node, then A 1 will also expand it. What are we saying here is that the set of nodes inspected by A 2 would be contained in the set of nodes inspected by A 1.

What does that mean? That if for example, this is the start node, and this is the set inspected by A 2 till any point of time. The set inspected by A 1 would be a superset of the set inspected by A 2.

In other words, we can write it by saying that if A 2 looks at any node n, then A 1 also must have looked at any nodes n. So, this I like this sign to mean implication the logical implication that for all n if A 2 expands n, then A 1 also must expand n this is what we want to show.

This is want we want to prove that any node which will be inspected by A 2 which is a more informed heuristic function would necessarily have to be inspected by A 1. And the corollary of this is that there may be nodes inspected by A 1 which are not inspected by A 2. So, nodes for example which are outside the region of this, which means that A 2 is doing less amount of search than A 1. I should actually draw this graph a little bit like this that we are anyway interested in one particular node essentially.

(Refer Slide Time: 08:16)



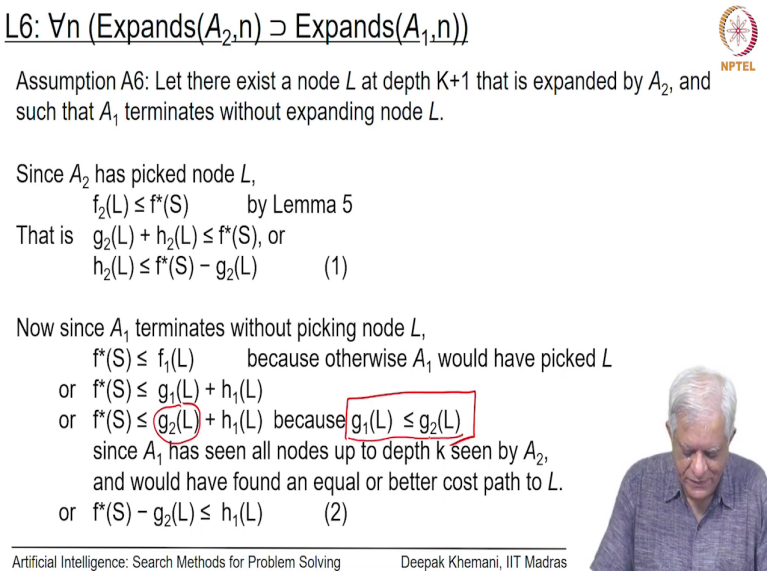
So, obviously, as we go towards the goal node both will explore some nodes, but there are nodes which A 2 A 1 inspects which A 2 is not inspected. We will do this proof by induction. And the basis is the base case is that both algorithms start with the start node S. So, this statement that we want to prove is true for the start node S that A 2 has seen S implies A 1 has seen S. And this is true by definition because both algorithms start with this node S.

The induction hypothesis is going to be that let this statement be true for all nodes up to some depth K essentially. So, we are assuming that this statement is true for all nodes up to some depth K and the induction step K will show that show that this is true for nodes which are at depth K plus 1.

So, if it is true at start which is at depth 0; and then if it is true at depth K and if you can show that it is; that it is true at depth K implies that it is to a depths K plus 1, then by the principle of mathematical induction we can show that the statement is true for all n essentially.

So, what are we doing here now? We have assumed that the base cases that both in inspected node S start node, then the hypothesis is that let it be true that they this statement is true for depth K which means that up to depth K any node inspected by A 2 is also inspected by A 1. And we want to show that it is also true for depth K plus 1. If it is true for depth K, it must be true for depth K plus 1. And we want to show this by contradiction.

(Refer Slide Time: 10:35)



So, we make an assumption that let there exists a node which we will call as L at depth K plus 1 that is expanded by A 2 such that A 1 terminates without expanding node L. Now, the first observation to make is that L should exist on the open of A 2.

Why? Because the property is true of for depth up to depth K that A 2 generated this node L which is at depth K plus 1, and therefore, A 1 also should have generated that node because the parent of this node L would have been inspected by both by the induction hypothesis.

A 2 has generated node L A 2 must, so that A 1 also must have generated node l. So, both have generated L. L is on the open of both, but we are saying that A 2 has expanded node L, but A 1 has terminated without expanding node L essentially. So, this is the assumption. And we will show that this is not possible. That if A 2 has expanded this node L which is at depth K plus 1, A 1 must necessarily have expanded this node L if it is to be consistent with what all we have stated so far.

So, let us look at this proof a has A 2 has picked this node L which means its f value must be less than equal to the optimal cost as stated by lemma 5. And which means that its g plus h value g 2 of L plus h 2 of L is less than equal to f star of S. And we can just rearrange this inequality to say that the heuristic value h 2 of L must be less than equal to the optimal cost f star of S minus the estimated cost of g 2 of L essentially.

We have simply got this by starting with the statement which we got from lemma 5 that the estimated cost must be less than the optimal cost. And therefore, we get this inequality which talks about h of L and relates it to f star of S and g 2 of L; g 2 of L is the cost of the path found by this algorithm which is A 2 from the start node up to the node L essentially.

Now, we have said that assumption is that A 2 has picked this node L, but we are saying A 1 terminates without picking node L which means A 1 has not picked node L. If it has not picked node L, then f star of S must be less than or equal to f 1 of L remember A 1 is using the heuristic function h 1. So, the worst case or the extreme case, we can think of is when

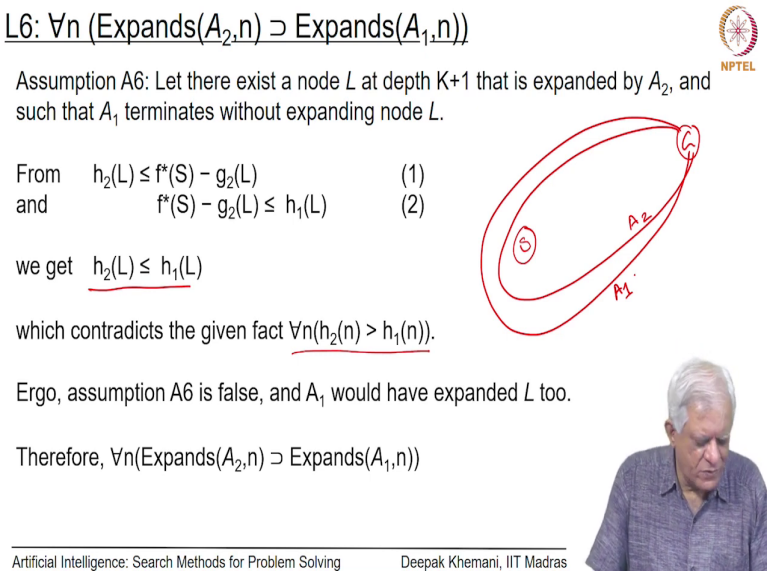
both are equal essentially. So, if both are equal, it is possible that the that it would have picked another path to the optimal another node which leads to the optimal path.

In other words because it has not picked node L, it has terminated without picking node L, the f value of that node can only be greater than the optimal value. It cannot be less than the optimal value; it could be equal in some cases, but it cannot be greater than that. So, again we express this in a different form. And we express it we replace f of f 1 of L by g 1 of L plus h 1 of L which is by definition. But now we say that we can write instead of g 1 of L we can write g 2 of L here.

And this is because we know that g 1 of L will always be less than or equal to g 2 of L. So, we can replace g 1 of L with a larger quantity, and the inequality would still hold. Why is this statement true that g 1 of L is always less than or equal to g 2 of L? Simply because of the induction hypothesis that since A 1 has seen all the nodes up to depth K seen by A 2.

It would have found the path that A 2 found, but it might have even found the shorter path forward you know. So, therefore, g 1 of L should be less than or equal to g 2 of l. So, now, we have this inequality which now we can rearrange to write it as saying that f star of S minus g 2 of L is h 1 of l.

(Refer Slide Time: 15:41)



Remember we have g 2 in both the cases. So, we can now take up inequalities 1 and 2, combine them, and this is what we get. In inequality 1 said that S 2 of L is less than equal to f star of S minus g 2 of L, and h 1 of L is greater than or equal to the same quantity which means by transitivity we get that h 2 of L is less than or equal to h 1 of L. All this is based on the assumption that there was a node at depth K plus 1 called L which was picked by A 2, but which was ignored by A 1.

And we are trying to show that could not be possible. And part of that proof is the statement that h 2 of L must have been less than or equal to h 1 of L, and because L 1 has A 1 has not picked L and A 2 has picked L. But now this contradicts the fact that for every node, we have said h 2 of n is strictly greater than h 1 of n. We said that h 2 of n is more informed than h 1 of n.

Now, clearly these two statements are not consistent with each other. This second statement says that our statement of more informedness says that h 2 is always larger than h 1 of n. Whereas what we have shown that if A 2 had picked L and A 1 had not picked L, it could only be under the conditions that h 2 of L was less than or equal to h 1 of L. So, these two statements are inconsistent with each other, and therefore, we can say that the assumption a six that we have made is false that A 1 would have expanded L 2.

Now, the principle of mathematical induction says that if we have shown that the statement is true for the base clause which is true in this case trivially because both nodes both algorithms expand the node S.

And then if you assume the induction hypothesis which says that this statement is true for all nodes up to K 1 which means that all nodes seen by A 2 are also seen by A 1 up to depth K 1. And then we show that it must also be true for all nodes at depth K plus 1 because the L was some arbitrary node that we chose at depth K plus 1.

And then we said that it is not possible that A 2 inspects L and A 1 does not inspect L which means that if A 2 has inspected L, A 1 also must inspect L. And then by the principle of mathematical induction, we can say that this property is true for all nodes, that means, that if A 2 has inspected any node n, A 1 should have also inspected that node n.

Which means from the diagram that I was trying to draw earlier that if this is the goal node and let us say this is a search space searched by A 2, then the search space explored by A 1 would be a superset of the search space explored by A 2; or in other words, the search space explored by A 2 is contained in the search space explored by A 1 at the point of termination. And since A 1 has seen more nodes than A 2, it would have taken longer time, and therefore, A 2 terminates faster.

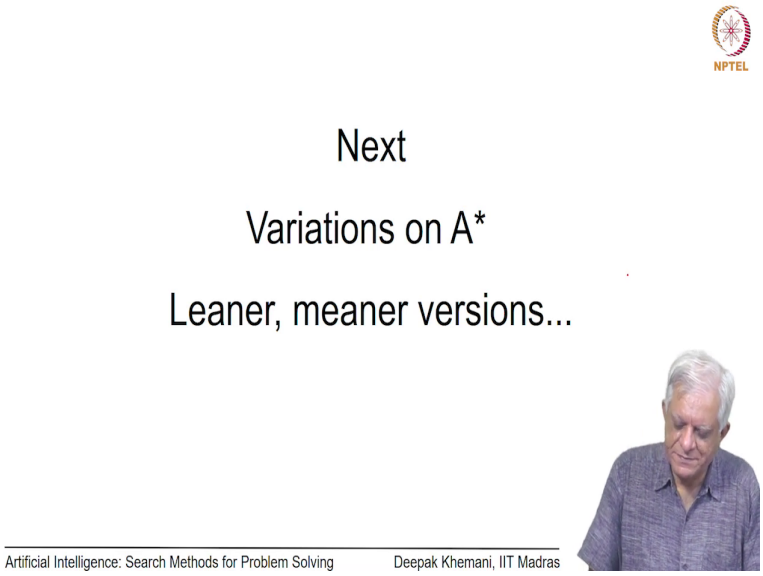
So, the lesson that we have learnt here the moral of the story is that the heuristic function but must underestimate the cost to the goal if it is to be admissible, but given that the higher the

heuristic function the better it is from efficiency point of view, because the higher the heuristic function the less the search space that the algorithm would have to explore.

So, this kind of these completes the basic properties of A star that we have shown. There is another property that we will discuss later which is to tied up with this property we stated for Dijkstra's algorithm, but we know that A star does not achieve that property which is that any node that it picks on the way also not necessarily the goal node, any node n. Dijkstra's algorithm would have found the shortest path to n, and we saw that that is because it is only working with known cause, but A star could have found path which is not optimal which later it might find the optimal path and revise the course.

So, we will see that under certain condition the A star also behaves like Dijkstra's in the sense that it will only pick a node and when it has found an optimal cost to the node n, but that requires a further restriction on the heuristic function. This is something that we will do a little bit later.

(Refer Slide Time: 20:39)



First what we will do is we will look at some variations on A star because having said all this A star remember is a variation of best first search it is used by it uses a heuristic function to guide the search. And like best first search the complexity of A star will depend on how good the heuristic function is. If the heuristic function we have seen that the better the higher the heuristic value, the lesser will be the space generated by A star explored by A star.

Nevertheless empirically it has been seen that the behavior of A star tends to be exponential in nature because very often we do not find very good heuristic functions. Both the space and the time complexity tend to be exponential.

And what we will do next is to look at a couple of variations which require which either find the solution faster may be at the expense of admissibility or we will find look at variations which will require less space than A star, they say space complexity would be lower.

They would find the optimal solution, but at the expense of time complexity. A little bit like what we did when we moved from best first search to DFID that we could find the optimal solution, but at a slightly more cost involving time. We will do that in the next session essentially ok.