Artificial Intelligence: Search Methods for Problem Solving.

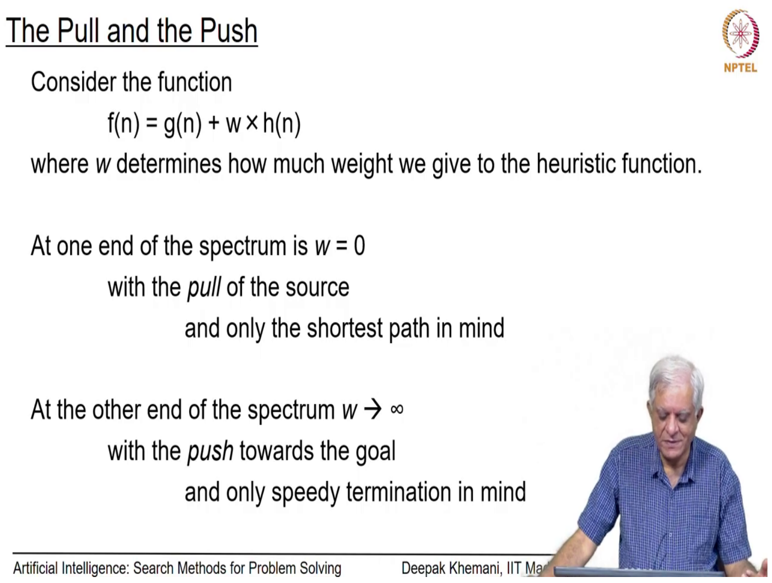
Prof. Deepak Khemani
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Chapter – 05
A First Course in Artificial Intelligence
Lecture – 41
B&B – A* - wA* - Best First

So, here we are again, we just finished looking at A star and before that we have looked at various algorithms which look search for paths in the state space or in the plan space. Let us start by trying to generalize the different algorithms that we have seen.

So, the algorithm that we have seen is branch and bound A star and best first search and what we are going to see now is w A star which falls under the generalization that we are I am going to talk about and we will try to look at these four algorithms under a unifying framework.

(Refer Slide Time: 00:59)



The Pull and the Push

Consider the function

$f(n) = g(n) + w \times h(n)$

where *w* determines how much weight we give to the heuristic function.

At one end of the spectrum is $w = 0$
with the *pull* of the source
and only the shortest path in mind

At the other end of the spectrum $w \rightarrow \infty$
with the *push* towards the goal
and only speedy termination in mind

Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Ma

So, the framework that I am talking about is to do with two aspects of search one is the pull that the source has on the candidates and this pull is realized in, but particularly in the algorithm branch and bound which tries to stick to as close to the source as possible. The other is a push and the push is towards the goal state and this is something which is experienced only in best first search.

And as we saw insta algorithm combines the two aspects and we showed that consequently it finds a shortest path, but let us try to look at all these algorithms in one framework and in fact, generalize it. So, we consider the function f of n is equal to g of n plus w into h of n where w is a weight that will determine how much importance we give to the heuristic function.

Now, g of n represents the pull factor where it says that we want to minimize the cost of coming from start node to the node n and h of n represents the push factor which says that let us get on to towards the goal as early as possible and let us choose a node which is as close to goal as possible or it appears to be as close as to the goal as possible because the heuristic function is only an estimate and w will control what how much importance we give to the heuristic function.

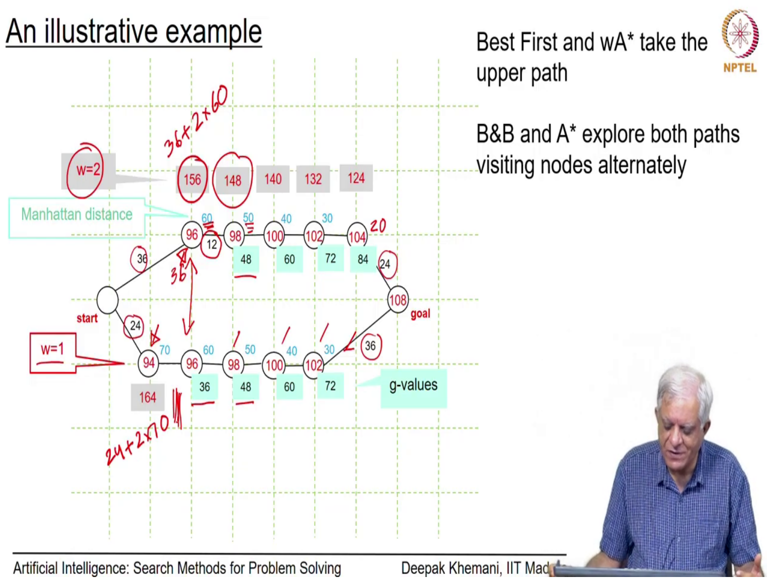
Now, at one extreme you can say that w is equal to 0 that means we just do not give any importance to the heuristic function and we only give importance to the pull to the source.

And we have only in mind that we have to find the shortest path essentially. And you know that if you put w is equal to 0 f of n becomes equal to g of n and that is a algorithm branch and bound or dijkstra algorithm that we had considered earlier. It only is worried about staying as close to the source as possible. At the other extreme we can say that w tends to infinity.

When we say w tends to infinity we are actually trying to say that the weight of g of n is 0 or negligible but in this one framework we can just express it by saying that w tends to infinity. If we had 2 weights; one for g of n and 1 for h of n we could have said that the weight for g of n is 0, but equivalently we can say that the weight for h of n is tending to infinity; that means, its much much more important than g of n then we have only the push towards the goal that is being active.

And what is in the task is to find the goal as soon as possible the task is for that we have speedy termination and we saw that if we put w equal to 1 here we would get A star search, but we can choose different values of w and we will choose one today just to see what is the effect of that.

(Refer Slide Time: 04:23)


An illustrative example

Best First and wA* take the upper path

B&B and A* explore both paths visiting nodes alternately

Now, look at this very small example where we have two goals two paths from start state to the goal state, the start state is the left most node and the goal state is the rightmost node.

And every horizontal edge has a cost of 12 I have not written them for the rest of the edges, but we will assume that they are all of course, 12. The diagonal edges are either 24 or 36 as you can see here and if you just think a little bit about this you will see that this is essentially the h cos have been chosen so, that they are about 120 percent of the Manhattan distance between those nodes essentially ok.

This is because we want to eventually have a admissible heuristic function or heuristic function which is going to be the Manhattan distance will underestimate the cost to the goal essentially ok. So, let us first consider what best first does that is the algorithm that we had started looking at it and the best first is what uses the Manhattan distance.

So, for the first two nodes that are generated which are this one and this one best first has to choose 1 and clearly it will choose the one with the lowest heuristic distance which is 60 because the other one is 70 as you can see and it will expand that.

Once it expands that it will generate its child which is of course, 50 and clearly you can see that this process will go on it will go from 60 to 50 to 40 to 30 to 20 which is what is the heuristic value of this node to 0 which is the heuristic value of a goal. So, the best first search algorithm will simply shoot off in the path which is on the top and will not even consider the path which is at the bottom.

I have carefully chosen these two paths so, that they are actually of equal length. So, if you inspect them also you can see that there is a kind of a symmetry about these paths the actual paths path cost is boths is same for the both, but our algorithm best first search only goes on one path and ignores the other that is because its starting point in this top path is closer to the goal and it simply prefers that.

Now, if you look at branch and bound. Branch and bound only considers g values; g values are in the shaded nodes as we can see the first node will have g value of 24 in the bottom graph bottom paths and 36 in the in the top graph because that is the edge cost. So, it will choose the bottom node which has a value of 24 and come to the second node which has a value of 36.

Now, this value of 36 is going to be equivalent to the value of 36 here on the top node and now it has to choose between them. So, since we assume that the choice is made arbitrarily, it will choose one of them and generate its child the child will be longer in both cases it will have a cost of 48 as we can see here.

So, it will go back to the other node with the value 36, then it will look at both the nodes with value 48 then it will look at both the nodes with value 60 and so, on and so, forth and explore boths the paths before it terminates with the value to the goal value by picking up the goal. The behavior of A star is similar in nature the values that you see inside the circles.

These are f values as computed by A star which as you can now realize happened when the weight is one essentially. So, f of n is equal to g of n plus h of n. Now, the first two nodes have a value of 94 and 96 as you can see 94 is obtained by 70 plus 24 70 is the heuristic distance and 24 is the edge cost.

Likewise, 96 is also computed in a similar fashion 60 is the heuristic distance and 36 is the edge cost. So, it will obviously, choose 94 exactly like what branch and bound does and generate its next child which is 96. So, now, we can see that both these nodes have the same f value. So, it can choose either one of them and not only that as it moves forward their children also have the same value.

So, the f values they increase as you go towards the goal and this increase happens because your estimates are becoming more and more accurate, the underestimating heuristic function has less to do because you have already covered part of the path. So, as you go from left to right in this graph which is which means as you go towards the goal the heuristic values increase.

And in this particular example the sorry the heuristic values decrease the f values increase and in this particular example they again increase in sync. So, from 96 at the next level they go to 98 and at the next level they go to 100, then they go to 102 and eventually find the paths to the goal which is of a cost of 108 essentially.

So, the behavior of A star in this particular graph is identical to the behavior of branch and bound. It visits the name nodes in the same order and it may find either the top path or the bottom path, but you can work that out that its more likely to find the bottom path because it will find that path to the goal first with a f value of 102 when it expands it will put the goal on open with a value of 1 0 8.

It will expand one the node with 1 0 4 also, but it would not have found a better path. So, the parent of 108 will remain 102. So, A star will find a path from this direction now what about w A star that is the algorithm that we want to look at now and we look at the value where weight is equal to 2.

And the values shown in this gray boxes are the values where f of n is equal to g of n plus 2 into h of n. So, for the first node for example, on the top the value is 156 and this comes because we take 60 in it is 36 plus 2 into 60 and that comes to 156 and for this node it is 24 plus 2 into 70 and that comes to 164.

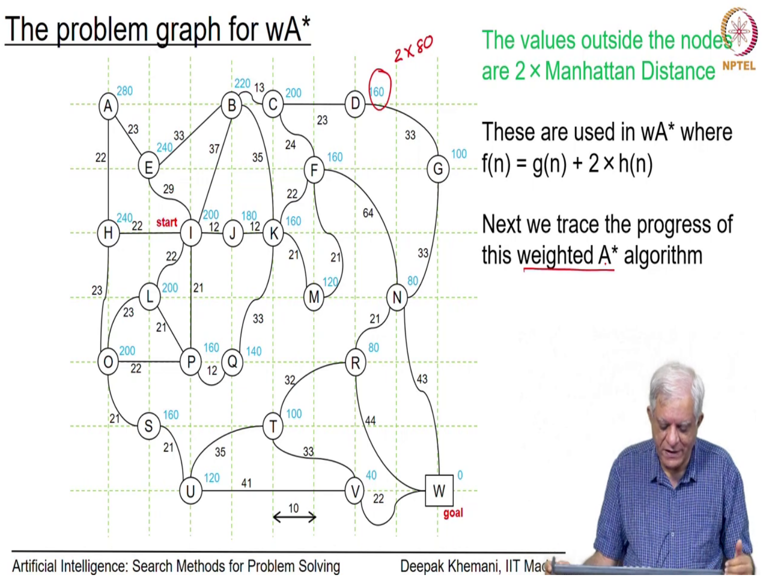
Now, you can see that 156 is the lesser of say 2 and when it generates its next child it gets a value of 148 which is again lesser than the node of 164. So, on this graph wA star with w equal to 2 behaves exactly like best first search it shoots off on the on the top path and continues till it reaches the goal and terminates and the algorithm never proceeds beyond the node on the bottom path essentially ok.

So, you can see that here is a small example graph where there are 2 paths to the goal both paths are of equal cost, but it says that their topology is a little bit different the top path is closer to the nodes are closer to the goal the bottom paths the nodes are closer to the source and both best first and is w A star they take the upper path and without looking back they just go ahead towards the goal whereas, branch and bound and A star they look at both the paths alternately.

So, they look at one node in the top paths one node in the bottom paths then one node in the top path and so, on they explore the graph much more diligently before saying that we have found a path to the goal and the path as we have shown in the case of A star and also in branch and bound which is Dijkstra algorithm is the optimal path.

So, now let us look at a situation of the problem that we have been looking at so, far the same example that we saw for best first and A star we now look at it from the perspective of w A star where w is equal to 2.

(Refer Slide Time: 13:22)



The problem graph for wA*

The values outside the nodes are 2 × Manhattan Distance

These are used in wA* where f(n) = g(n) + 2 × h(n)

Next we trace the progress of this weighted A* algorithm

Artificial Intelligence: Search Methods for Problem Solving

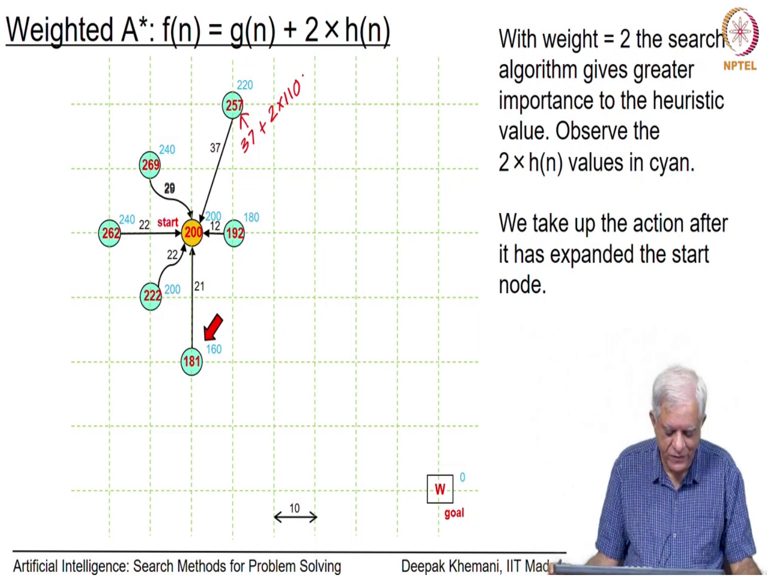
Deepak Khemani, IIT Mad

So, we can see that the values that I have been written on the next to the nodes for example, this 150 is 160 is 2 into 80.

So, the 80 as you can come is the Manhattan distance and the value that the node will consider the algorithm will consider is 160 which is twice the Manhattan distance because the weight is 2 (Refer Time: 14:00) essentially.

So, what I have done is I have pre labeled the entire graph with 2 into Manhattan distance values so, that its easy for us to simulate hand simulate the algorithm essentially ok. So, we now trace the progress of weighted A star over this graph ok.

(Refer Slide Time: 14:22)


Weighted A*: f(n) = g(n) + 2 × h(n)

With weight = 2 the search algorithm gives greater importance to the heuristic value. Observe the 2 × h(n) values in cyan.

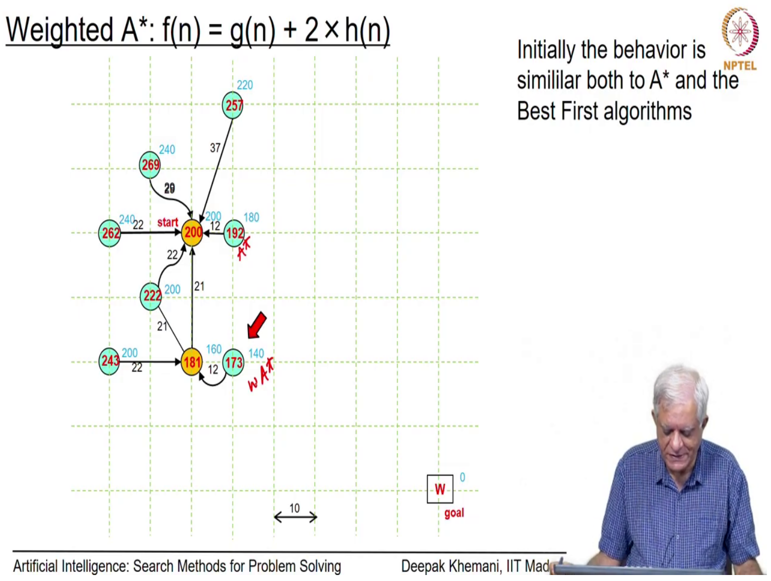
We take up the action after it has expanded the start node.

Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Mad

So, as before we start off by expanding the start node you can see that the start node has a value of 200 which is the value that the weighted A star gives to h. So, h is actually 100 if you remember and this 200 comes because we do 2 into 100 essentially and likewise this value of 257 for example, has been got by 37 plus 2 into 110 and likewise for the other nodes.

So, as before the orange nodes are the ones in the closed we take up the action after the first node s has been expanded and the cyan nodes are the ones on the open and as before we will keep marking the best node on cyan by this red block yellow that we have and w A star is going to expand the nodes.

(Refer Slide Time: 15:39)



Weighted A*: f(n) = g(n) + 2 × h(n)

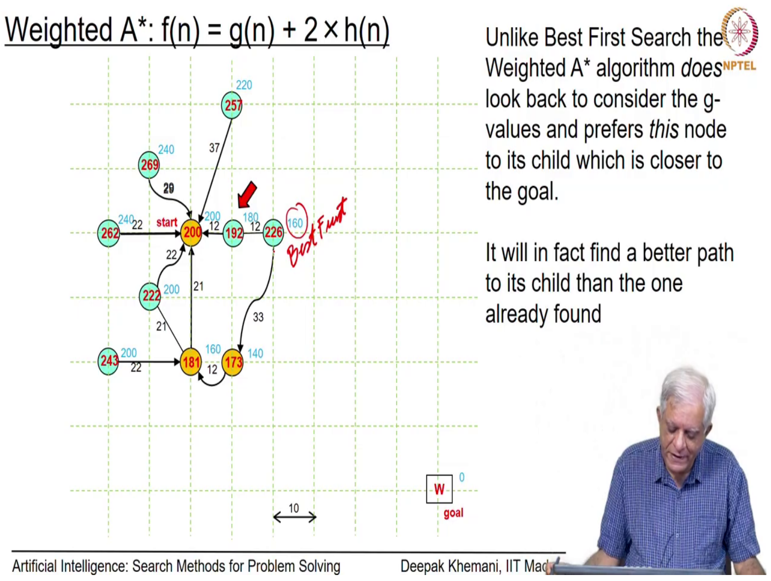
Initially the behavior is simililar both to A* and the Best First algorithms

Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Mad

So, initially of course, we have seen that all the algorithms that we have seen I have chosen this particular node and expanded that node. So, the behavior is similar to that, but from here onwards the behavior of the different algorithms varies essentially.

So, best first search and A star had both seen this node with a value of 173. Now, A star had seen this node with a value of 192 which was 90 plus 12 which was 102. So, A star would have picked this one, but w A star picks this one and also the best first search picks this one.

(Refer Slide Time: 16:22)



Weighted A*: f(n) = g(n) + 2 × h(n)

Unlike Best First Search the Weighted A* algorithm *does* look back to consider the g-values and prefers *this* node to its child which is closer to the goal.

It will in fact find a better path to its child than the one already found

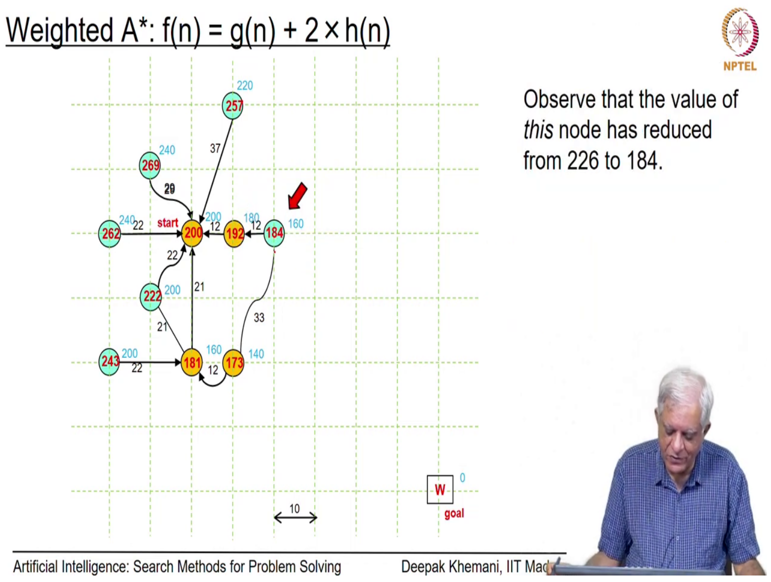
Artificial Intelligence: Search Methods for Problem Solving   Deepak Khemani, IIT Mad

And if you have gone through the example of branch and bound which I asked you to do you would see that even branch and bound would pick this node essentially. At this point the behavior of the best first and w A star changes best first would have chosen this value because it is closer.

And if you remember best first in fact, from here it spread off towards the goal without looking at this node with a value 192 which w A star is still looking at and the reason for that is a w A star also looks back like A star it takes into account the cost of reaching the node as well.

And you can see that the paths that best first search has found so, far is a expensive path it is 21 plus 12 plus 33 whereas, A star has had found the shorter path which is of two edges of length 12 and w A star also does that essentially. So, best first would have expanded this node with a heuristic value of 80 or 2 into heuristic value being 160.

(Refer Slide Time: 17:26)

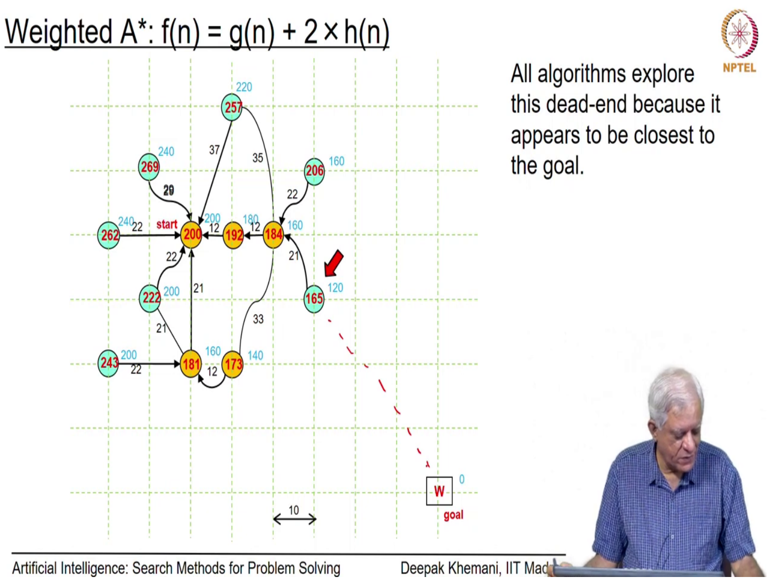

Weighted A*: f(n) = g(n) + 2 × h(n)

Observe that the value of *this* node has reduced from 226 to 184.

Artificial Intelligence: Search Methods for Problem Solving   Deepak Khemani, IIT Mad

But A star looks at another node which is on open and then it finds us cheaper paths to this node which has now come down to 184 the heuristic value does not change of course, the f value changes the f value is defined differently it was 226 before w A star expanded the node with a value 192 and it becomes 184 after that essentially.

(Refer Slide Time: 17:49)


Weighted A*: f(n) = g(n) + 2 × h(n)

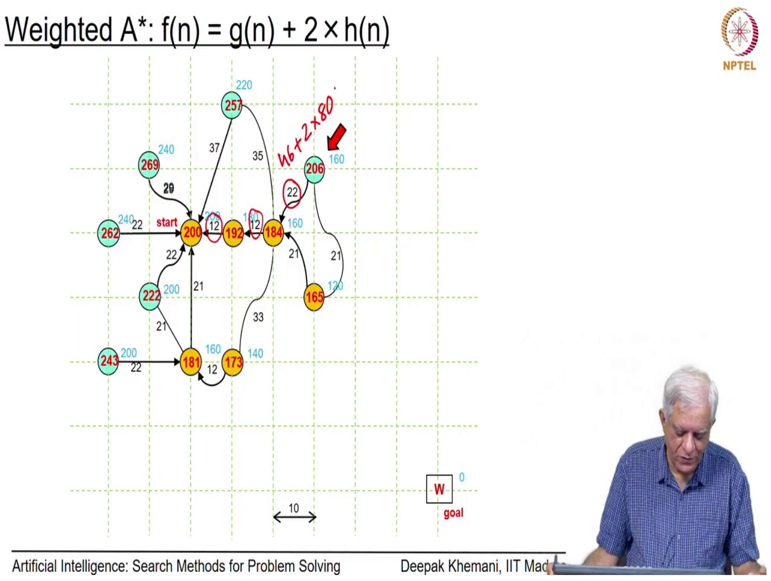
All algorithms explore this dead-end because it appears to be closest to the goal.

Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Mad

Then of course, it expands this particular node and as we have seen this particular node which is going to be the next node to be picked by w A star is a favorite of all algorithms all heuristic algorithms because it appears to be closest to the goal.

And as you can see that because this distance is short or even the Manhattan distance is small all algorithms have a tendency to pick this node and we had seen that this was a kind of a dead end because there was no pass going from this node except to the node that we are visiting now which is has a value of 206.

(Refer Slide Time: 18:13)



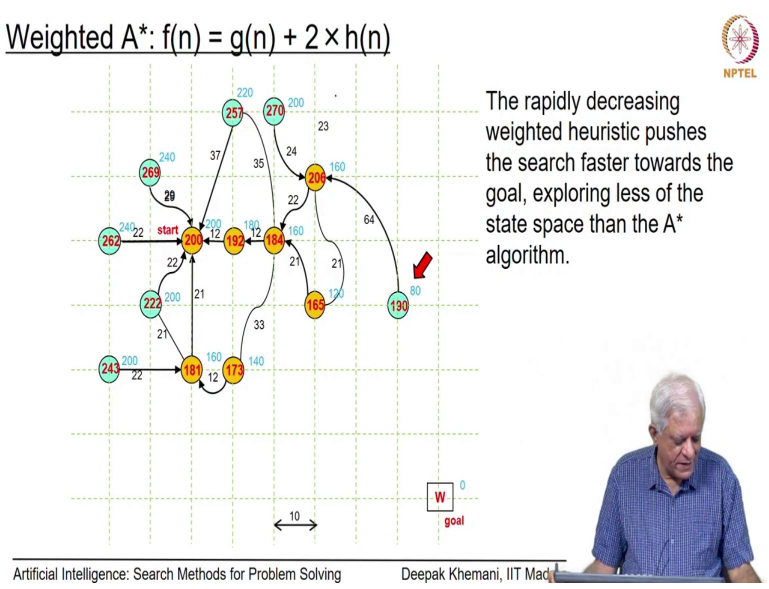
Weighted A*: f(n) = g(n) + 2 × h(n)

Artificial Intelligence: Search Methods for Problem Solving

Deepak Khemani, IIT Mad

So, again remember that this 206 has come by adding up the cost. So, what are the cost here? 12 here 12 here 22 here. So, g of n is 46 plus the heuristic value must 80. So, 2 into 80 and that gives us a value of 206 and that is a smallest of all the cyan nodes or in other words smallest of all the nodes on open.

(Refer Slide Time: 18:51)


Weighted A*: f(n) = g(n) + 2 × h(n)

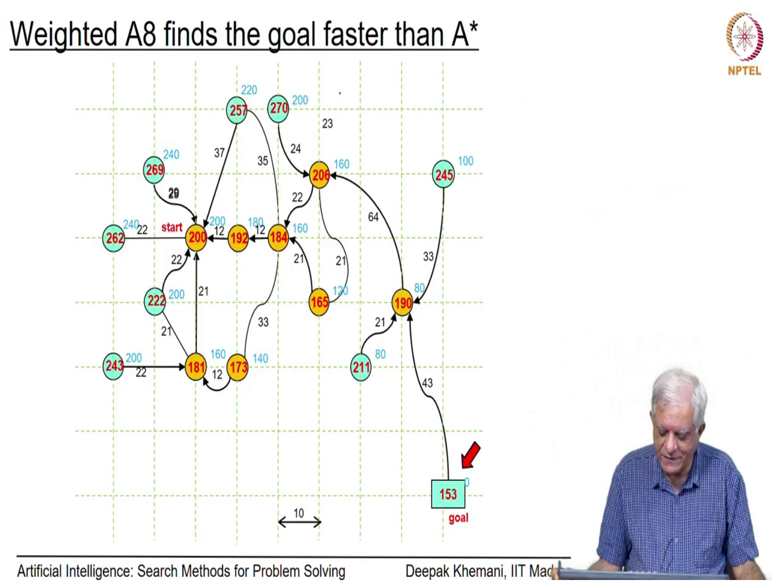
The rapidly decreasing weighted heuristic pushes the search faster towards the goal, exploring less of the state space than the A* algorithm.

Artificial Intelligence: Search Methods for Problem Solving

Deepak Khemani, IIT Mad

And then the w A star algorithm which has a greater push factor it has a greater effect of heuristic value.

(Refer Slide Time: 19:05)

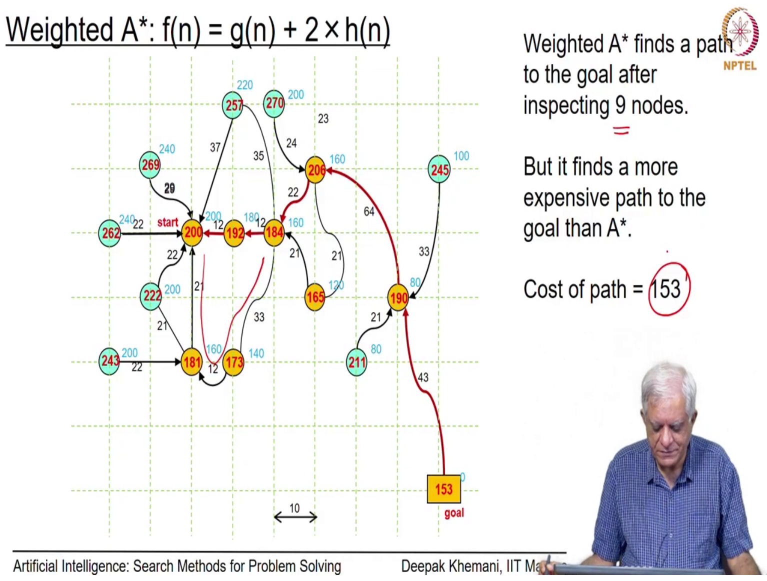

Weighted A8 finds the goal faster than A*

Artificial Intelligence: Search Methods for Problem Solving

Deepak Khemani, IIT Mad

Now heads of rapidly towards the goal and along the same path in which best first search also went and finds a goal this thing without exploring the path which either branch and bound or w A or A star would have done.

(Refer Slide Time: 19:16)



Weighted A*: f(n) = g(n) + 2 × h(n)

Weighted A* finds a path to the goal after inspecting 9 nodes.

But it finds a more expensive path to the goal than A*.

Cost of path = 153

Artificial Intelligence: Search Methods for Problem Solving

Deepak Khemani, IIT Ma

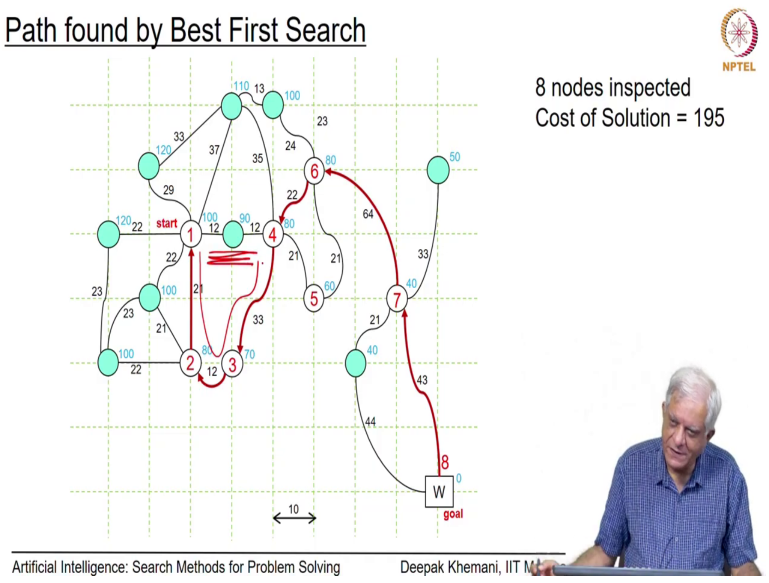
What happens as a consequence is that it has found a different path the path is different from both from A star and also from best first. So, if you remember A star had found the path of course, 148 we will do a quick comparison next.

Best first had found the longer path best first had perform the path which was going like this which was; obviously, much longer which had had cost of 195 or something whereas, w A star finds a path in between it is a cost of 158. It has seen 9 nodes 9 nodes have been inspected and put into clothes closed which is less than what A star did ah, but it has found a more expensive paths than A star.

So, one thing to observe here is that even though this algorithm has a flavor of A star in the sense that we use both g of n and h of n because of the excessive weight given to s h of n the heuristic function has become something which is not admissible. Remember that we had said that A star is admissible as long as the heuristic function underestimates the distance to the goal.

Now, clearly when we multiply the Manhattan distance by 2 it is not going to underestimate the distance so, that property of heuristic function being admissible is lost and as a consequence we can see that w A star has found the path which is more expensive than the optimal path, but it has found it faster.

(Refer Slide Time: 20:52)



Path found by Best First Search

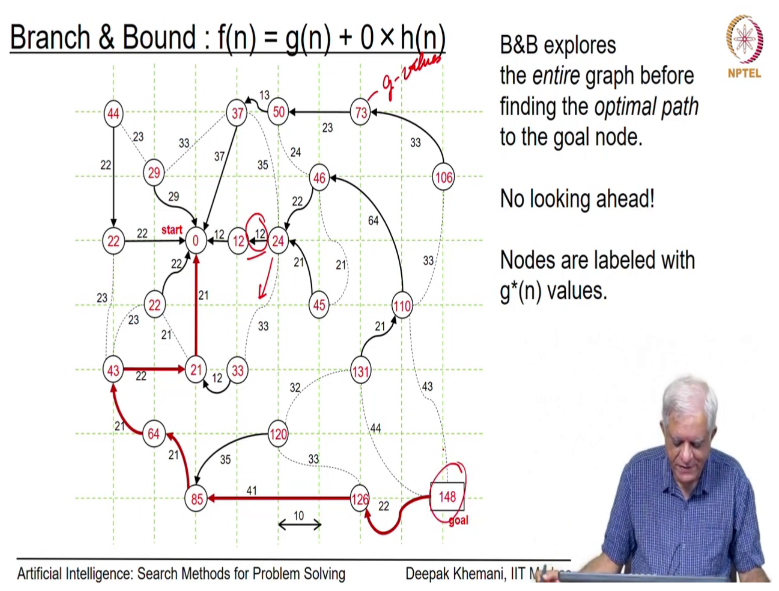
8 nodes inspected
Cost of Solution = 195

Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT M

So, this is again just a just to show what is the path it has found sorry this is a path that was found by best first search algorithm. We have seen this was the first algorithm we considered and we saw that it found this longer path like this it had only inspected eight nodes and the numbering is given here 1 2 3 4.

The order in which it inspects those node and it found the path of cost 195 which is much more than even what w A star has done and the reason is that w A star managed to find this particular segment which is much cheaper than this segment of coming down and then going up again essentially.

(Refer Slide Time: 21:31)



Branch & Bound : $f(n) = g(n) + 0 \times h(n)$

B&B explores the *entire* graph before finding the *optimal path* to the goal node.

No looking ahead!

Nodes are labeled with $g^*(n)$ values.

Artificial Intelligence: Search Methods for Problem Solving

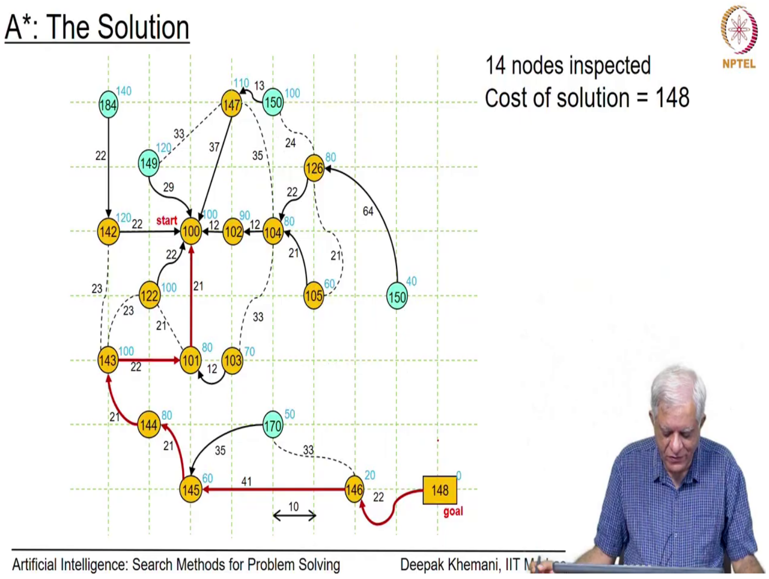
Deepak Khemani, IIT M

I had asked you to look at branch and bound on the same graph and I hope you did that is at some point. Branch and bound extends the path with the lowest g values. The g values for this particular graph have been written inside the nodes here and the g values have been computed as the best value for that path because there are there may be multiple paths to a node.

For example, for this node there are two paths one is a either coming from here or one is coming from here, but the g value and the parent pointer it reflects the best path after the algorithm is terminated. Remember that branch and bound and dijkstras algorithm they can reassign parents and update cos of node which are on open.

So, this is a this is what branch and bound does it in this particular example it explores the entire graph and that is in this case its entire graph because the goal node is at one corner and all other nodes are closer than goal nodes, but even in a larger graph where is the goal nodes were not in the corner branch and bound would explore a larger amount of space then either A star or best first or w A star it does not look ahead at all essentially. Now, goal it has found is optimal; obviously, we have seen that Dijkstra algorithm always does this.

(Refer Slide Time: 23:06)



A*: The Solution

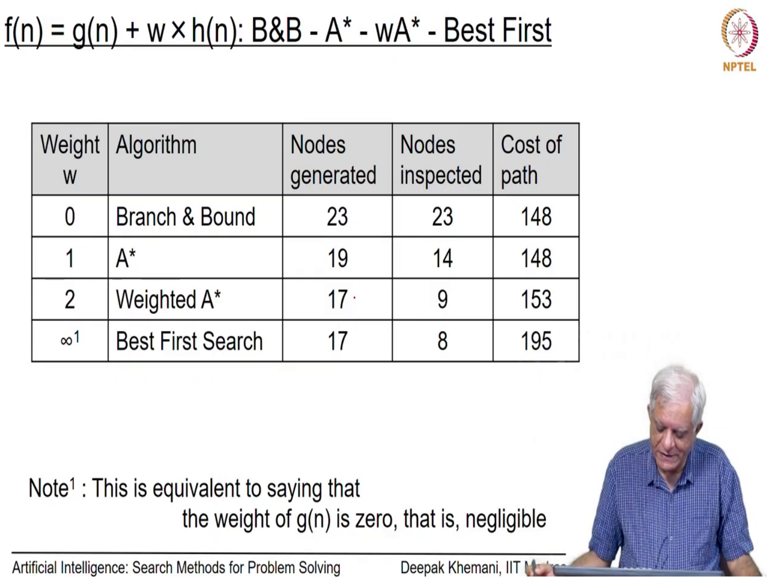
14 nodes inspected
Cost of solution = 148

Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT M

And if you compare this with A star which we had also seen earlier the same path is found by A star as branch and bound except that A star has only seen a sub graph of the graph which is depicted here.

The nodes in orange are the closed nodes which is much less than what transient bound does but which is much more than what best first and w A star do it has inspected 14 nodes, but it has found the optimal path and the reason for that is that its heuristic function was admissible.

(Refer Slide Time: 23:41)



$f(n) = g(n) + w \times h(n)$: B&B - A* - wA* - Best First

Weight w	Algorithm	Nodes generated	Nodes inspected	Cost of path
0	Branch & Bound	23	23	148
1	A*	19	14	148
2	Weighted A*	17	9	153
∞[1]	Best First Search	17	8	195

Note[1] : This is equivalent to saying that the weight of g(n) is zero, that is, negligible

Artificial Intelligence: Search Methods for Problem Solving   Deepak Khemani, IIT M

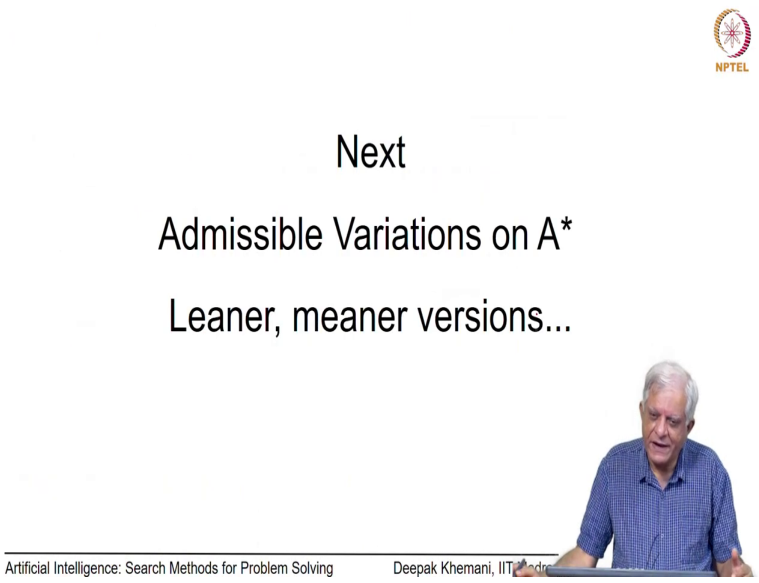
So, here is a comparison of the four algorithms you can see that as weight increases in this table weights from 0 to 1 to 2 to tending to infinity and as I said when you write weight is infinity it is equivalent to saying that the weight for g of n is 0 but just another way of saying this thing.

The number of nodes that are generated which means which are in open and closed decreases as the heuristic as the as the importance to the heuristic function increases it starts with 23 and comes down to 19 and then 70.

What is more important is that the number of nodes actually picked by the algorithm that also decreases as the weight increases. As we have seen in this example branch and bound peaks inspects all the 23 nodes, but A star inspected only 14 of them found a cost of 148 which is as good as the path found by branch and bound. So, its clearly an improvement.

Weighted A star saw 9 nodes pick 9 nodes, but it found a more expensive path of 153 and branch and bound saw only 8 nodes and found a much more expensive path and that is because we saw that at one point it was coming down and going up and ignored one node which w A star did not ignore and in the next session we will look at variations of A star which are admissible.

(Refer Slide Time: 25:01)



Next

Admissible Variations on A*

Leaner, meaner versions...

Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Madras

We saw that w A star for example, did not find the optimal path now we are going to focus on variations which will find the optimal path, but do some kind of saving and in particular we will see that the savings that we are interested is in spaces essentially ok. So, we will do that in the next session.