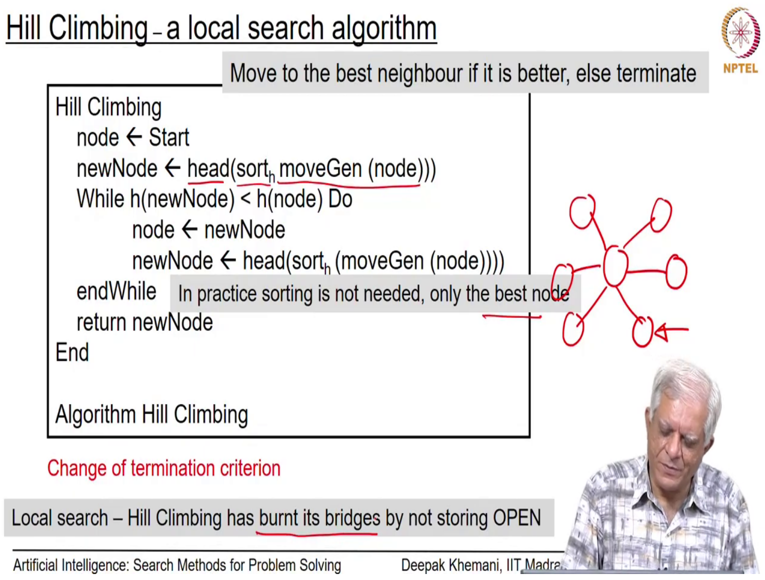
Artificial Intelligence: Search Methods for Problem Solving

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

Chapter – 03
A First Course in Artificial Intelligence
Lecture – 24
Heuristic Functions and the Search Landscape

(Refer Slide Time: 00:15)



Hill Climbing - a local search algorithm

Move to the best neighbour if it is better, else terminate

```
Hill Climbing
   node ← Start
   newNode ← head(sort_h moveGen (node)))
   While h(newNode) < h(node) Do
          node ← newNode
          newNode ← head(sort_h (moveGen (node))))
   endWhile
   return newNode
End

Algorithm Hill Climbing
```

In practice sorting is not needed, only the best node

Change of termination criterion

Local search – Hill Climbing has burnt its bridges by not storing OPEN

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

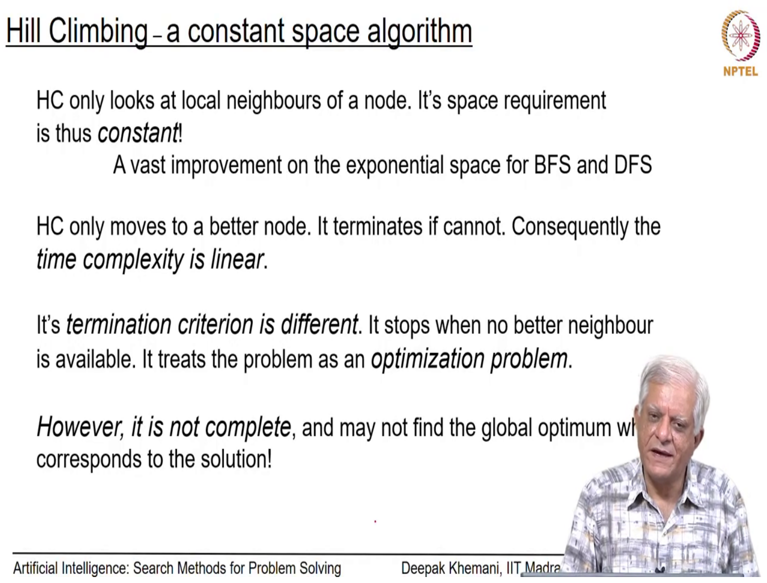
So, welcome back last time we saw this algorithm called hill climbing which is constant space algorithm which explores the search space in a heuristic fashion. And what this algorithm does is if you recall, it start it is at some node this thing it generates the children of these this node. Then amongst the children it identifies the best one. So, let us say this is the best one, what it does is it just moves to that node and forgets about the other nodes ok. And then it generates the children of this node.

And it will move to the best node. So, it just repeats this process throwing away everything else that it has generated in the past. So, because of this the space requirement is constant and the only space, it needs is the neighbors of a current node that is that are generated. So, in this algorithm that we have written here, we have essentially said that you apply moveGen to a node then, you sort it then you take the head. And if this new node is better than the old node, then you move to it otherwise you terminate essentially.

So, the strategy in a nutshell is move to the best neighbor, if it is better else terminate essentially. Now, we have written all this sort in this algorithm because, we adopted this algorithm from the best first algorithm. In practice we do not need to sort it all you need is the best node and that can be found in linear time simply by scanning all the neighbors essentially.

Also, observe that the termination criteria has changed, we are not saying that terminate when you found the goal node. We are saying that terminate when there is no better node essentially. And the most important thing is that it has burnt its bridges by not storing the previous open node essentially. As a result of which the algorithm may not take you to the desired goal node as we will see in some examples. And then we will work towards improving upon this process.

(Refer Slide Time: 03:03)



Hill Climbing - a constant space algorithm

HC only looks at local neighbours of a node. It's space requirement is thus *constant*!

A vast improvement on the exponential space for BFS and DFS

HC only moves to a better node. It terminates if cannot. Consequently the *time complexity is linear*.

It's *termination criterion is different*. It stops when no better neighbour is available. It treats the problem as an *optimization problem*.

However, it is not complete, and may not find the global optimum wh corresponds to the solution!

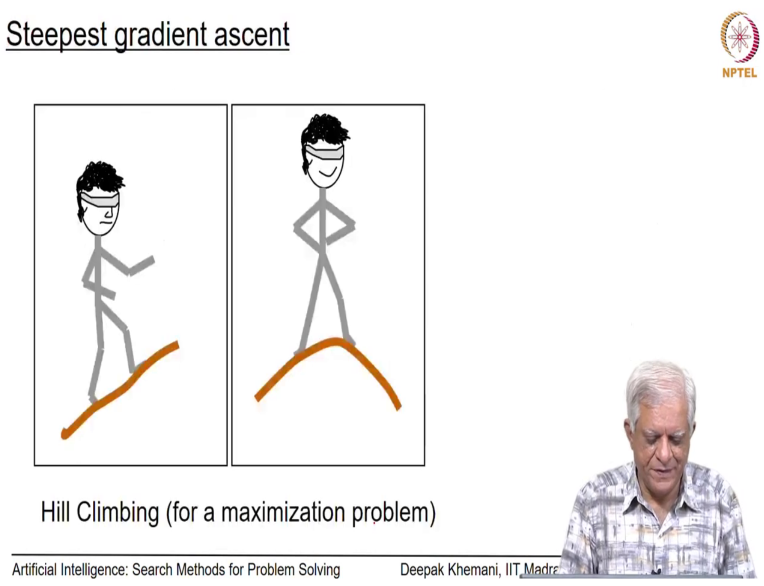
Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Madra

So, its a constant space algorithm as we have just said, it only looks at the local neighbors of a node and therefore, its space requirement is constant which is the vast improvement on the earlier algorithms which required exponential amount of space.

Since it only moves to a better node and it terminates if it cannot consequently the time complexity is linear because, it can just move some number of steps in one direction and terminate essentially. As we have said the termination criteria is different it stops, when no better node is available.

And it treats the problem as thus as a optimization problem, it says look for the best value of the heuristic function essentially. However, because of these local nature of its search there is a trade off, when the trade off there it is not complete and we may not end up on the global optimum, which will correspond to the solution that we were looking for in the original space essentially.

(Refer Slide Time: 04:09)



Steepest gradient ascent

Hill Climbing (for a maximization problem)

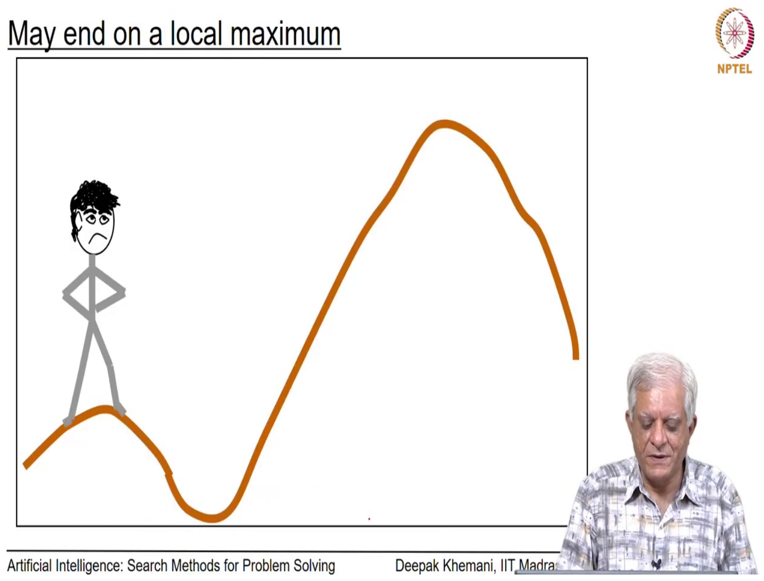
Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Madra

So, this was just a figure that we saw essentially what it does is that it follows the gradient. And it is also known as the steepest gradient ascent or steepest gradient descent, as the case may be as we also mentioned optimization handles both minimization and maximization.

If you have a minimization problem which is the case that you would have, if you were minimizing the heuristic value. Because, the heuristic value of the goal node is going to be 0, we can convert this into a maximization problem by taking the negative of the heuristic value or by subtracting it from some constant number.

So, a minimization and maximization are just two sides of the same coin. And in this diagram we have shown a maximization problem, because of the name that the algorithm which is hill climbing. So, imagine you are climbing a hill blindfolded and then you reach what you think is the top because all your surrounding neighbors are worse than you.

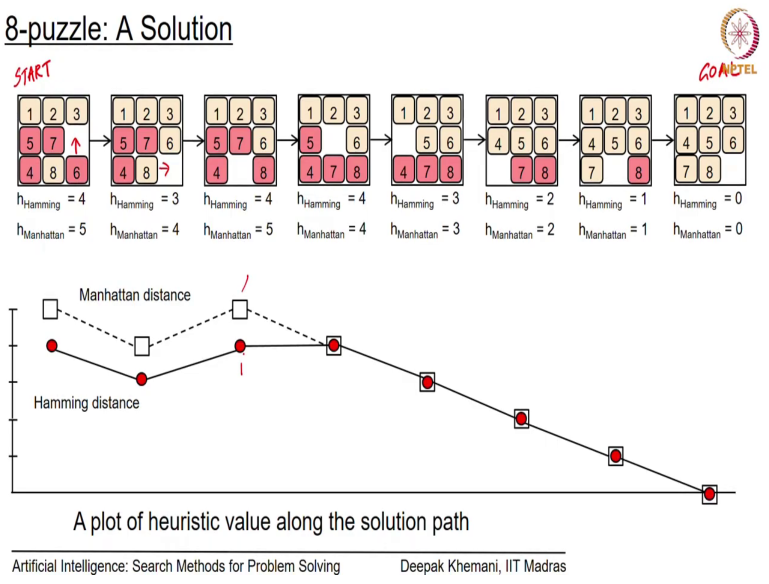
(Refer Slide Time: 05:17)


May end on a local maximum

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

But, when you open your eyes you find that its not really the top its a local maxima and algorithm will get stuck at this point essentially.

(Refer Slide Time: 05:27)


8-puzzle: A Solution
START
GOAL
$h_{Hamming} = 4$ $h_{Manhattan} = 5$
$h_{Hamming} = 3$ $h_{Manhattan} = 4$
$h_{Hamming} = 4$ $h_{Manhattan} = 5$
$h_{Hamming} = 4$ $h_{Manhattan} = 4$
$h_{Hamming} = 3$ $h_{Manhattan} = 3$
$h_{Hamming} = 2$ $h_{Manhattan} = 2$
$h_{Hamming} = 1$ $h_{Manhattan} = 1$
$h_{Hamming} = 0$ $h_{Manhattan} = 0$
Manhattan distance
Hamming distance
A plot of heuristic value along the solution path
Artificial Intelligence: Search Methods for Problem Solving
Deepak Khemani, IIT Madras

Here is an example that we have seen earlier from the 8 puzzle solution. So, the leftmost is the start state that we have for this particular example. And the right most is the goal state and you can see that what you we are showing here is a solution as a human would solve it. The heuristic function that we have discussed earlier was either of the 2 hamming distance, which simply counted the number of tiles out of place all the Manhattan distance which computed the length of the Manhattan distances to for each tile to its destination location.

Now, in the start state you can see the tiles which are labeled 5 7 4 and 6 are out of place. So, the hamming distance is 4 and if you look at the Manhattan distance carefully you will see that it is 5. What this sequence of states shows is a solution as solved by human being and what we have drawn below that is the plot of the heuristic value as the solution progresses.

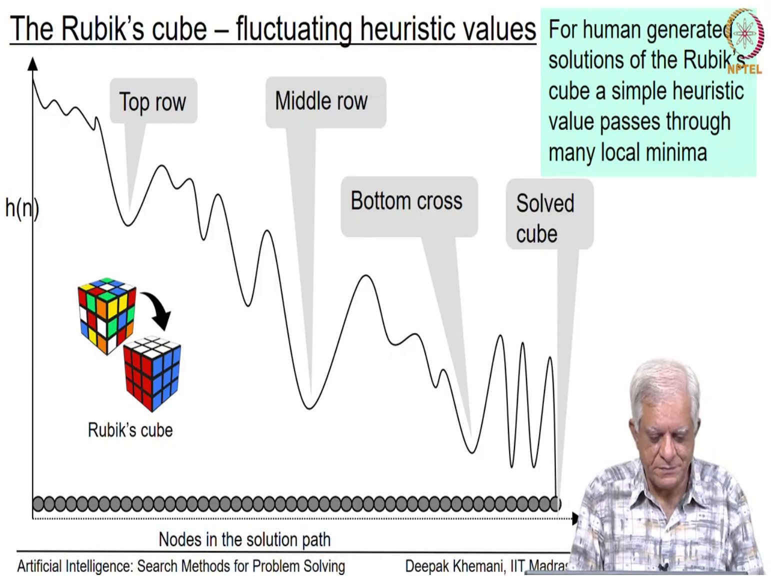
So, you can see that the red circles correspond to the hamming distance and the squares correspond to the Manhattan distance. And the hamming distance starts with a value of 4 and once you make one move which is that you move the tile 6 up. Then, its got into its final destination. So, there are only 3 misplaced tiles.

So, the hamming distance has now reduced to 3, but next when you move in the solution tile 8 to the right you see that it has gone from its correct position to its wrong position. So, it is misplaced now. So, the heuristic value has gone up and as you follow the solution, you can see that the heuristic value initially decreased and then increased and then eventually decreased essentially.

From the states four onwards you can see hamming distance and Manhattan distance is same because, essentially there are 4 3 2 1 respectively tiles out of place and each of them is exactly one move away from their destination. So, the number of tiles out of place is turns out to be equal in this example to the Manhattan distance.

And we can see there are after going over this local maxima, irrespective of which heuristic function we use. Eventually we go down to a heuristic value of 0 which corresponds to the solution. But hill climbing would have got stuck either here or in fact, here for both the heuristic values and it would not have moved any further essentially.

(Refer Slide Time: 08:23)



The Rubik's cube – fluctuating heuristic values

For human generated solutions of the Rubik's cube a simple heuristic value passes through many local minima

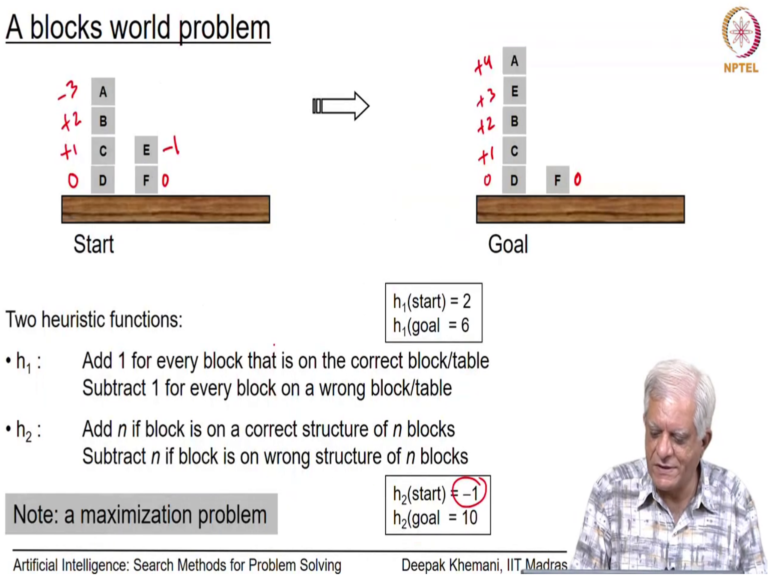
Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Madras

So, in general if you look at problems for example, the Rubik's cube, which you may be familiar with where you start off with a jumbled of cube, when the goal is to have all faces having one color each. And if you know how to solve it and nowadays most children figure out or learn how to solve it.

And if you were to plot the heuristic value in a similar fashion, you would see that its got many local optima on the way in this case many local minima. So, when you do the top row then you get one minima, but then to get the middle row you have to disrupt the top row and that kind of a thing.

So, this process if you look at it from the goal reasoning point of view that you have broken up of your problem of solving the Rubik's cube to sub goals, which say that first do the top row, then do the middle row, then do the bottom cross, then you know the bottom corners and so on. Then moving from one sub goal to other you find that you have to disrupt what you have achieved in the previous, previous sub goals. And this is called as non serializable subgoal problem which we will come back to later when we look at planning again.

(Refer Slide Time: 09:37)



A blocks world problem

Two heuristic functions:

- h_1 : Add 1 for every block that is on the correct block/table
  Subtract 1 for every block on a wrong block/table
- h_2 : Add *n* if block is on a correct structure of *n* blocks
  Subtract *n* if block is on wrong structure of *n* blocks

h_1(start) = 2
h_1(goal = 6

h_2(start) = -1
h_2(goal = 10

Note: a maximization problem

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

So, talking about planning here is a small planning problem and its a blocks world problem, what you see on the left hand side is the start state. So, you have an infinite table you have label blocks all of the same size. And one block can be stacked on top of another block and the left hand side is the start position and the right hand side is a goal position. And we want to look at how search based planning algorithm, would use a heuristic function to try and solve this problem.

So, we can think of two heuristic functions in this case, just the just like we did in the case of the 8 puzzle. The first function is that for every block that is on the correct block or on the table, you add 1 for that and subtract 1 for every block which is on the long block essentially. So, the other heuristic function is that you add n if a block is on a correct structure of n blocks essentially.

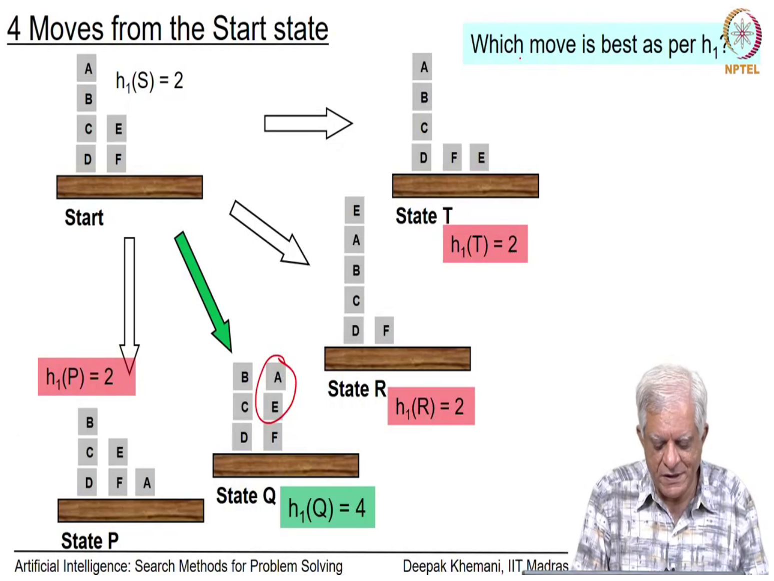
So, let us look at the first one first, in the first heuristic function we are just simply trying to see whether everything is on a correct block. And as you can see there are 6 blocks and in the goal state everything is in the correct position. So, your heuristic value is 6 essentially. In the start state you can see that B is on the correct block, C is on the correct block, D is on the correct block, but A is on the wrong block and E is on the wrong block. And F is on the correct block essentially.

So, eventually it comes down to us value of 2 essentially, for the second heuristic function if you look at the values. Then you can see that you would add 0 because, the size of the tower is 0 plus 1 plus 2 minus 3 0 minus 1. And eventually you will come down with a value of minus 1 which is shown here essentially. And if you compute in a similar fashion the value of the goal node, then you would have 0 here plus 1 plus 2 plus 3 plus 4 and 0 for this.

This is a heuristic function that we have used we could have added 1 for the bottom most block, but that would not have changed things very much and you can see that the value of this is ten essentially. Also observe that the way that we are counting the heuristic value, we are saying that we count the number of blocks in the correct position.

And because that is maximum in the goal position the problem has become a maximization problem, we could have also counted the number of blocks in the wrong position, but then you know that could just be obtained by taking a negation of this. So, as we said maximization and minimization are two sides of the same problem.

(Refer Slide Time: 12:50)


4 Moves from the Start state

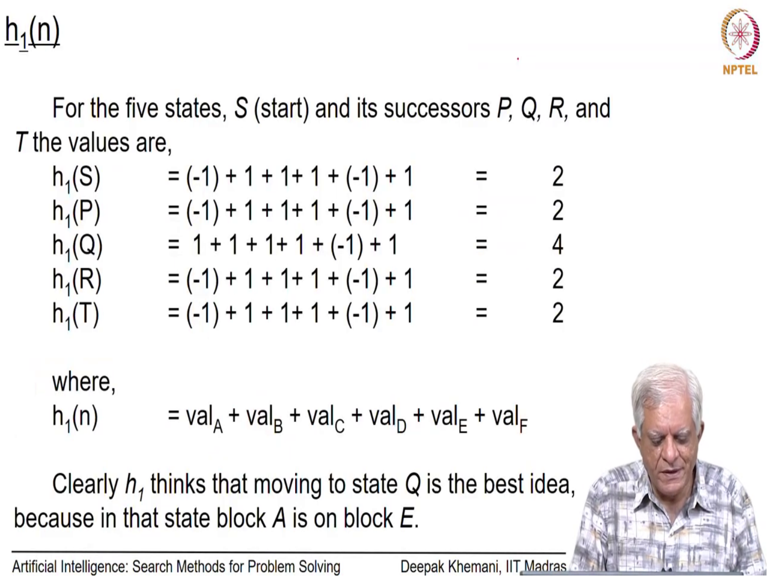
Which move is best as per h_1?

So, let us see how the 2 heuristic functions perform and we assume that we are doing hill climbing. So, from the start state we can imagine 4 moves that you pick up A and put it on the table or you pick up A and put it on top of E or you pick up E and put it on top of A or you pick up E and put it on the table essentially.

So, this four states are shown here and also the heuristic values as per the first heuristic function. So, you can see that the start state has a heuristic value of 2 and 2 of the neighbors which is state T and state R, they also have heuristic value of 2.

As well as state P which also has a heuristic value of 2 and the state Q has a higher heuristic value which is that of 4 and that higher value has come, because of the fact that this A is on E which is what it should be in the goal state if you remember the goal state essentially ok. So, the goals in the goal state A was supposed to be on E essentially.

(Refer Slide Time: 14:03)



$h_1(n)$

For the five states, *S* (start) and its successors *P*, *Q*, *R*, and *T* the values are,

$h_1(S)$	$= (-1) + 1 + 1 + 1 + (-1) + 1$	=	2
$h_1(P)$	$= (-1) + 1 + 1 + 1 + (-1) + 1$	=	2
$h_1(Q)$	$= 1 + 1 + 1 + 1 + (-1) + 1$	=	4
$h_1(R)$	$= (-1) + 1 + 1 + 1 + (-1) + 1$	=	2
$h_1(T)$	$= (-1) + 1 + 1 + 1 + (-1) + 1$	=	2

where,

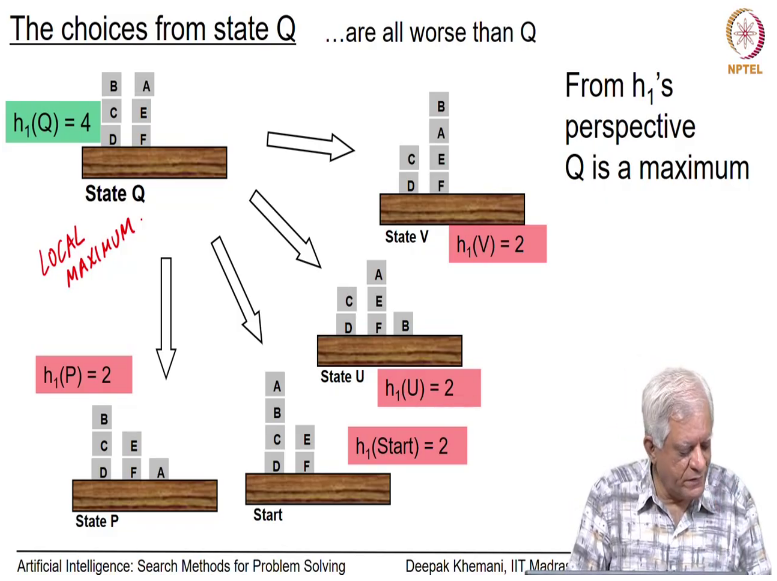
$h_1(n) = val_A + val_B + val_C + val_D + val_E + val_F$

Clearly h_1 thinks that moving to state *Q* is the best idea, because in that state block *A* is on block *E*.

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

So, this algorithm will move to state Q and this simply shows the competition that we have done. We have added one for everything which is on the correct spot and minus 1 for everything, which is on the wrong spot and this simply gives us the value that we have just computed.

(Refer Slide Time: 14:16)



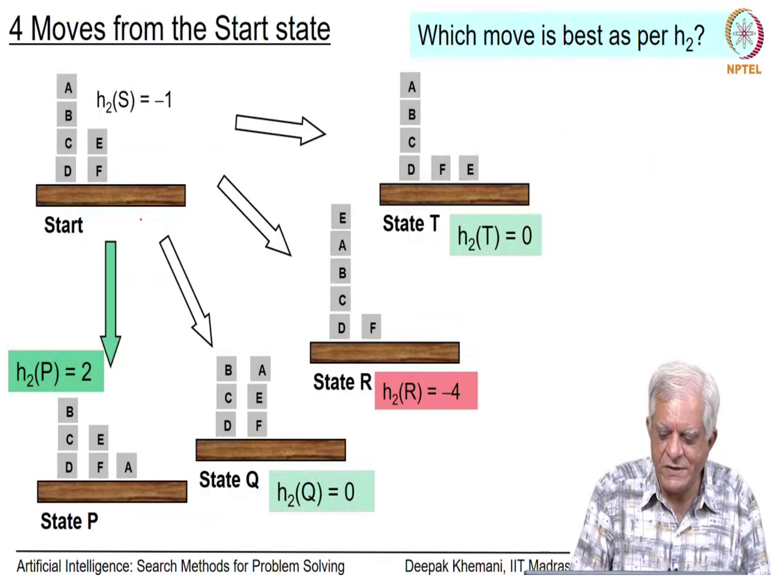
The choices from state Q ...are all worse than Q

From h_1's perspective Q is a maximum

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

Now, if you are at state Q hill climbing look is looking at 4 neighbors, state Q has now value of 4. And all the four neighbors as you can see and you can compute, for yourself and check have a heuristic value of 2 essentially; which means that state Q has become a local maximum and our algorithm will stop at this state not move any further.

(Refer Slide Time: 14:50)



4 Moves from the Start state

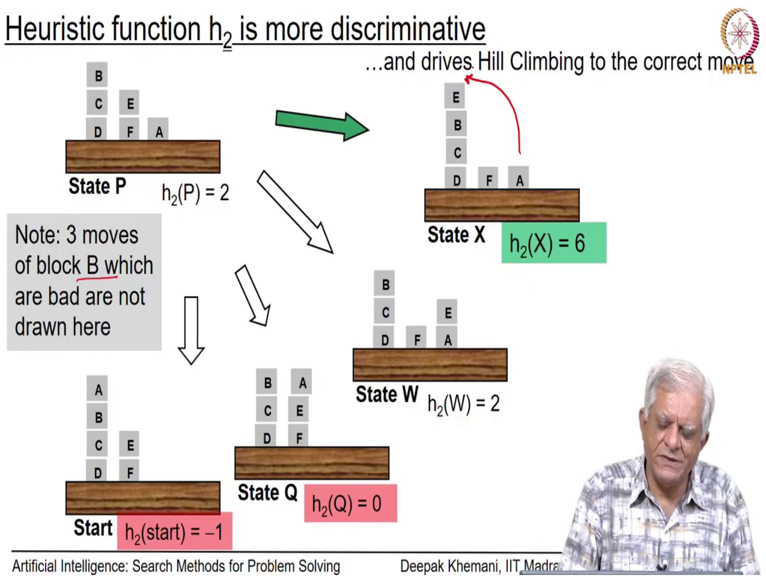
Which move is best as per h_2?

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

Now, let us see how the second heuristic function which was a little bit more uniform as you can see because, it looked right down to the table and to see if everything was on the correct block or not. So, again we have the same four moves from the start state to the states P Q R and T.

And you can see that now the function is a little bit more discriminative the values are fluctuating a little bit more, state P has a value of 2 the start state has a value of minus 1 and remember that the goal state has a value of plus 10 state P has a value of 2s. It turns out to be the best state as depicted by the green arrow state Q has a value 0 state R has a value minus 4 and state T has a value of 0.

(Refer Slide Time: 15:39)


Heuristic function h_2 is more discriminative
...and drives Hill Climbing to the correct move
State P $h_2(P) = 2$
State X $h_2(X) = 6$
Note: 3 moves of block B which are bad are not drawn here
State W $h_2(W) = 2$
State Q $h_2(Q) = 0$
Start $h_2(start) = -1$
Artificial Intelligence: Search Methods for Problem Solving
Deepak Khemani, IIT Madras

So, our algorithm will move to state P. Now, from state P it sees many moves, now as you can see there are 3 towers here and or 3 stacks and from each of those 3 stacks you could pick up the topmost block and put it in three different places either on the 2 other stacks or on the table.

So, there would be 3 2 3 which is 9 moves we are not drawn all the 9 moves because, we have assumed that the 3 moves for block B which are going to be bad anyway because, they are disrupting a well formed stock we have not drawn here. So, because you know they will anyway have a worse value.

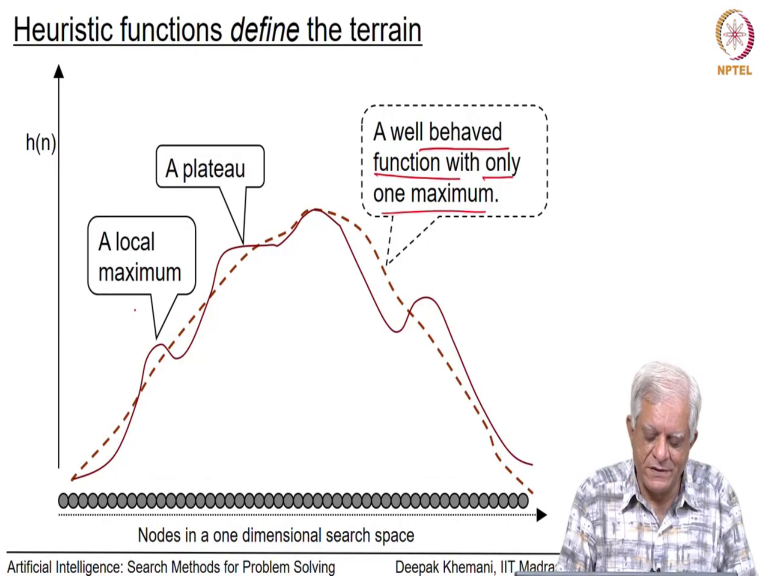
But from the state P which has a heuristic value of 2 if you look at the remaining states then one of them of course, is a start state which is minus 1 we do not want to go back, there state

Q. We have seen it has a value of 0, but there are states W and X which are new states state W has a value of 2 and state X is a value of plus 6 essentially.

So, clearly the algorithm will move to state X and as you can imagine all that remains to be done, from this state is to pick up this block A and put it on top of block E. And then you would have solved the goal and indeed in the next round this algorithm would have found a goal state.

So, we saw two versions of the heuristic function; one version drove the algorithm to the goal state this is the second heuristic function. And the first heuristic function got stuck at the local maxima.

(Refer Slide Time: 17:26)


Heuristic functions *define* the terrain

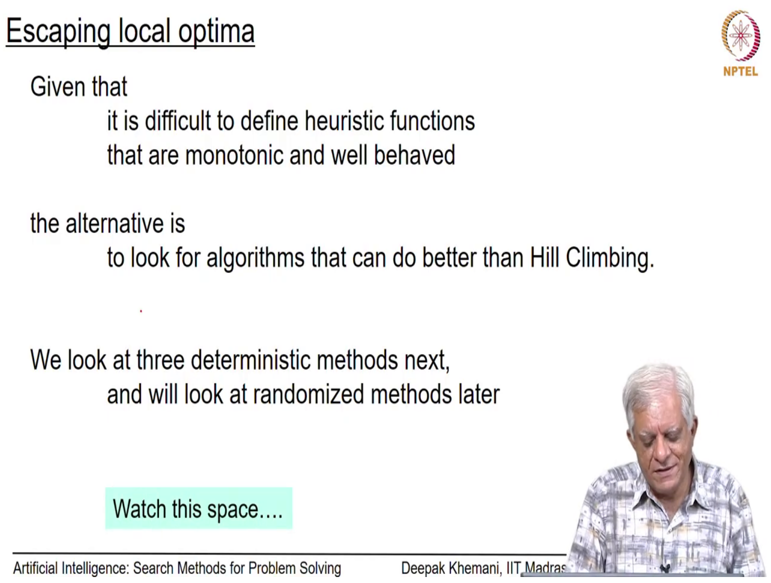
Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Madras

So, from this you can kind of generalize and make an observation, that the surface that you are traversing using the steepest gradient ascent or descent as the case may be is defined by the heuristic function itself. So, in our case in the blocks world example the second heuristic function, defined well behaved surface or function which had only one maximum essentially.

Now obviously, if such there is such a surface then hill climbing will work well, but on the other hand other heuristic functions, they may have things like local maxima or they may have a plateau. And then the hill climbing algorithm would get stuck essentially. So, that is a drawback of hill climbing this is that its not always possible to derive perfect heuristic functions.

If we are perfect heuristic function we would hardly need to do search you know so, but in the real world that is not the case. So, we often end up doing search and we have to look for better and better ways of doing this search.

(Refer Slide Time: 18:32)


Escaping local optima

Given that
 it is difficult to define heuristic functions
 that are monotonic and well behaved

the alternative is
 to look for algorithms that can do better than Hill Climbing.

We look at three deterministic methods next,
 and will look at randomized methods later

Watch this space....

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

So, what is the tasks in ahead of (Refer Time: 18:36) us ahead of us now? That we have looked at this locals search algorithm called hill climbing which has many good features primarily the fact that it requires constant amount of space that is a biggest redeeming feature of this thing. But its problem is that it can get stuck in a local optima local maxima or the local minima as the case may be. So, what we need to do next is to look for better ways.

So, assuming and you have to take my word for it here is that it is difficult to define heuristic functions that are monotonic and well behaved in the sense that, they will have only one local one local maxima or minima, which will also be the global maximum or minimum. Its assuming that its difficult to design such functions and we saw the Rubik's cube, example just try to imagine a function which will give you a monotonic surface.

The alternative is to look for algorithms that can do better than hill climbing. So, if we cannot change the heuristic function can we change the algorithm yeah and that is a question that we will be asking next. We start with three deterministic methods next because so, far ours algorithms have been deterministic. And later on we will move on to randomize methods ah, which are in fact, very popular in the optimization community. So, there are methods like simulated annealing or there are methods like genetic algorithms or there are methods like swarm optimization or ant colony optimization.

But we will come to them after we have done with the deterministic method, which is what we will study in the next class. And we will also look at the different formulation of the search problem, that instead of doing search in the state space; we will look at doing search in what is called as a solution space or a plan space. So, well start with that and look at some deterministic methods for optimization in the next class.