

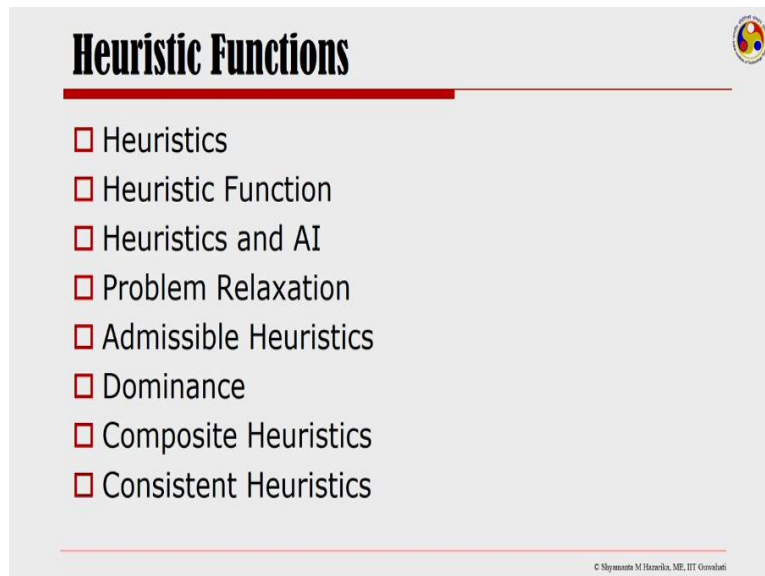
**Fundamentals of Artificial Intelligence**  
**Prof. Shyamanta M Hazarika**  
**Department of Mechanical Engineering**  
**Indian Institute of Technology - Guwahati**

**Module - 2**  
**Lecture - 4**  
**Heuristic Search**

Welcome to fundamentals of artificial intelligence. Today we look at heuristic functions. In the previous lecture, we had looked at uninformed search strategies. Uninformed search strategies do not use any problem domain information while looking for a solution in the problem space. Informed search on the other hand exploits problem domain information which we referred to as heuristics.

In this lecture, we will look at more closely what heuristics are, how heuristics could be formulated for a given problem. And we will list certain properties of heuristics, before we move on to informed search strategies in the next lecture.

**(Refer Slide Time: 01:29)**



**Heuristic Functions**

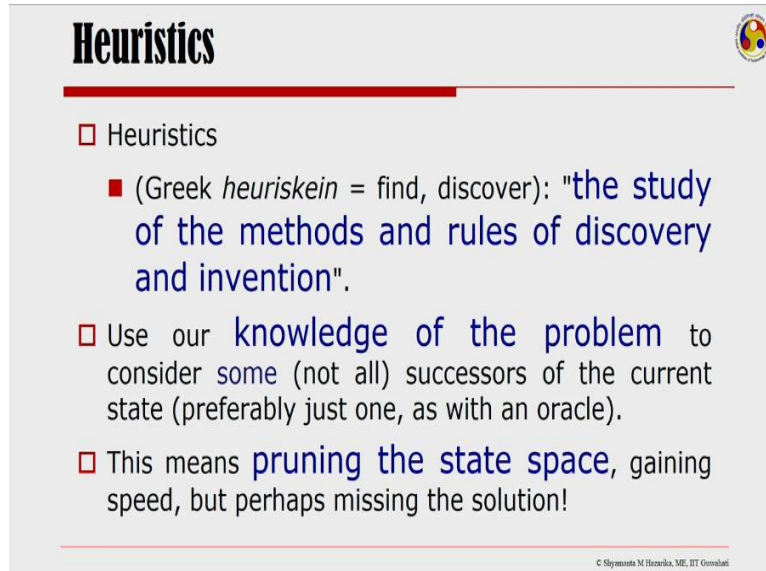
- Heuristics
- Heuristic Function
- Heuristics and AI
- Problem Relaxation
- Admissible Heuristics
- Dominance
- Composite Heuristics
- Consistent Heuristics

© Shyamanta M Hazarika, ME, IIT Guwahati

At the end of this lecture, it is expected that we would be able to understand or define heuristics; would be able to formulate heuristic functions; look at how heuristics and AI search are integrately related. We will look at the design of heuristic functions through a method called problem relaxation. And then, we will look at what are admissible heuristics. We will look at what do we mean by dominance between given heuristics.

We then gone to define something called composite heuristics and look at consistent heuristics. Heuristics is a Greek word which would mean find or discover. It usually is the study of methods and rules of discovery and invention.

(Refer Slide Time: 02:31)



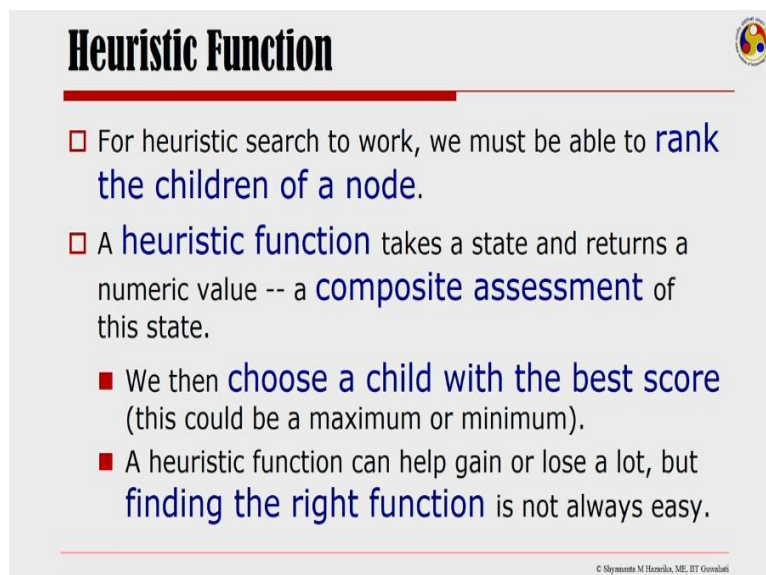
## Heuristics

- Heuristics
  - (Greek *heuriskein* = find, discover): "the study of the methods and rules of discovery and invention".
- Use our knowledge of the problem to consider some (not all) successors of the current state (preferably just one, as with an oracle).
- This means pruning the state space, gaining speed, but perhaps missing the solution!

© Shyamanta M Hazraika, ME, IIT Guwahati

Here we use our knowledge of the problem to consider some, not all the successors of a given state. This means, we will be able to prune the start state space gaining speed. But then the risk is perhaps we would also be missing the solution. Therefore, designing a heuristic and working with heuristics is of utmost importance.

(Refer Slide Time: 03:00)



## Heuristic Function

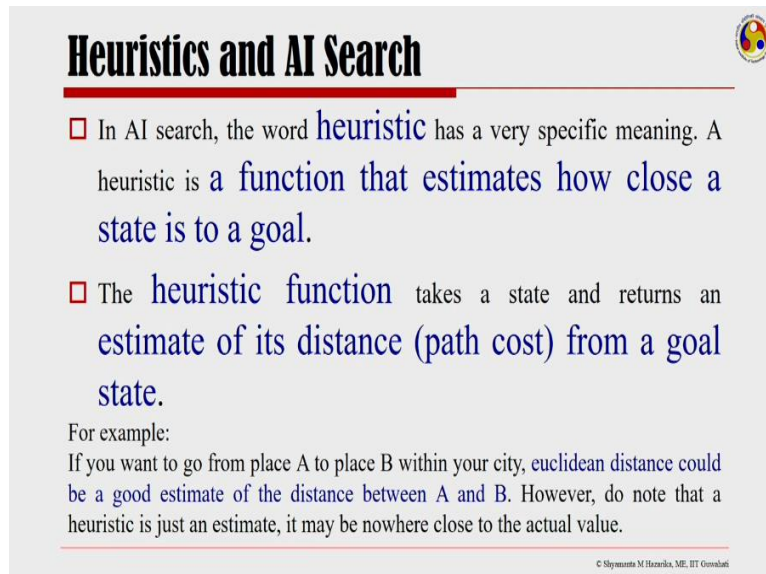
- For heuristic search to work, we must be able to rank the children of a node.
- A heuristic function takes a state and returns a numeric value -- a composite assessment of this state.
  - We then choose a child with the best score (this could be a maximum or minimum).
  - A heuristic function can help gain or lose a lot, but finding the right function is not always easy.

© Shyamanta M Hazraika, ME, IIT Guwahati

Heuristic functions are able to rank the children of a given node. And a heuristic function takes a state and returns a numeric value. It is somewhat like a composite assessment. We

choose a child with the best score and a heuristic function can help gain or lose a lot. But finding the right function is not always easy.

(Refer Slide Time: 03:33)



**Heuristics and AI Search**

- In AI search, the word **heuristic** has a very specific meaning. A heuristic is a function that estimates how close a state is to a goal.
- The **heuristic function** takes a state and returns an estimate of its distance (path cost) from a goal state.

For example:  
If you want to go from place A to place B within your city, euclidean distance could be a good estimate of the distance between A and B. However, do note that a heuristic is just an estimate, it may be nowhere close to the actual value.

© Sreyamshi M Hazarika, M.E., IIT Guwahati

Heuristics in AI search has a very specific meaning. It is a function that estimates how close a state is to a goal. In the next couple of slides, we will look at what we mean by that. The heuristic function takes a state and it returns an estimate of its distance, its cost of the path from the goal. Here it would be appropriate to think of an example from real life. Like, if you want to go from place A to place B within your city, euclidean distance could be a good estimate of the distance between A and B.

However, we need to note that the euclidean distance may not give the correct distance between A and B. The actual distance would be different. What the euclidean distance is giving us is an estimate of the distance between A and B. And it may be nowhere near close to the actual value.

(Refer Slide Time: 04:44)

## Playing 8-Puzzle

### Problem Solving as State Space Search

**s ; Start Node**

2	8	3
1	6	4
7		5

**g ; Goal Node**

1	2	3
8		4
7	6	5

© Shyamanta M Hazra, ME, IIT Guwahati

So, let us go back to playing the 8-puzzle game that we have been working on from the previous lecture and look at problem solving as state space search. On your left is the start node of an 8-puzzle tile configuration and on the right is the goal node. We want to search through the problem space for reaching from the start to the goal node.

(Refer Slide Time: 05:12)

## Problem Solving as State Space Search

Using a Heuristic Function

**g ; Goal Node**

1	2	3
8		4
7	6	5

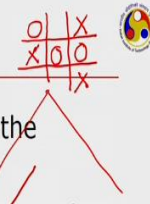
© Shyamanta M Hazra, ME, IIT Guwahati

We have seen this in the previous lecture, where we had looked at exploring all the nodes as it is expanded. Here we have the start node given to us and the goal node is this configuration on your right. So, what I wanted to show by repeating an exercise that I have done for uninformed search is that, when I am using some form of problem domain information which is roughly called an heuristic function, the whole idea of looking for a solution in the problem space has been directed.

Rather than doing expansion at every level, like I expand everything here, everything here, like I have done for what we have named as breadth-first search in our previously lecture. I have been very clearly able to go down a path which takes me to the solution in a directed manner. This is the motivation of using heuristic functions.

(Refer Slide Time: 06:24)

## Heuristics and AI Search



- The **principal gain** - often spectacular - is the **reduction of the state space**.
  - For example, the full tree for Tic-Tac-Toe has  $9!$  leaves. If we consider symmetries, the tree becomes six times smaller, but it is still quite large.
  - With a fairly simple heuristic function we can get the tree down to 40 states.
- Heuristics can also **help speed up exhaustive, blind search**, such as depth-first and breadth-first search.

© Shyamanta M Hazra, M.E., IIT Guwahati

The heuristic functions, the principal gain is often very spectacular. It is the reduction of the state space. The reduction of the state space is sometimes very, very surprising. Like, if you talk of a very simple game that you must have played in school like the tic-tac-toe, where you put circles and your opponent put it, puts an x. And then, and you again put a circle and your opponent puts an x.

And then, you put a circle again to stop your opponent from winning. Who then puts an x to stop you from winning. If you look at this small interesting game of tic-tac-toe, it has 9 factorial leaves. If we consider symmetries, the tree becomes 6 times smaller. But it is still quite larger in number. But, if you use a simple heuristic function which we will be talking of during our informed search techniques, we get the tree down to just 40 states.

So, just looking at 40 states, I can decide who is going to win or who is going to lose in this game of tic-tac-toe. And that is what heuristics are so important for. Heuristics can help speed up exhaustive blind search such as depth-first and breadth-first. In the previous slide, we were looking at something like a definitely breadth-first search, motivated by the heuristic

function of misplaced tiles. And we could directly go into the solution by not exploring every thing in the problem space.

(Refer Slide Time: 08:14)

**Problem Relaxation**

A standard approach of creating heuristics is problem relaxation.

We relax a problem by adding some new actions such that search is not required to find the solution cost in the relaxed problem.

Heuristics created as solutions to relaxed problems are usually admissible because adding new actions should only reduce the solution cost and not increase it.

40 ops  
X L U D

© Shyamanta M Hazarika, IIT Guwahati

The slide features two 3x3 grids. The left grid has a circled '4' above it and checkmarks in the top-left, middle-left, and bottom-left cells. The right grid has a circled '5' above it and checkmarks in the top-left, middle-left, and bottom-right cells. To the right of the grids, the text '40 ops' and a vertical list 'X L U D' are written in red. The words 'create' and 'relaxed problems' in the third paragraph are circled in red.

Problem relaxation is what decides how to create heuristics. A standard approach of creating heuristics is to relax a problem by adding some new actions. By relaxation we mean having actions possibly in the problem space which is not there in the original problem. But then, the idea of adding this actions as such, that to get to a solution, you do not need search anymore. The solution cost that I get can be directly computed without a search.

And that solution cost in the relaxed problem gives me the heuristic function. Heuristic created as relaxed problems are usually something called admissible. We will look at little while later what we mean by admissibility of heuristics. But first, let us look at what we mean by problem relaxation. Let us assume that we have the 8-puzzle game. The 8-puzzle game in terms of its original actions has only 4 operators which are just going right, left, up or down.

When I was counting the displaced tiles, I was not looking at this 4 operations. But I was looking at a relaxed problem of the whole idea. The idea being that I assume that I could take each tile and put it in its original position. In such a way, getting to the goal from the start was not about searching for in a graph, but trying to understand how many times I have to pick up things which are not in place and put them there. But before that, we want to understand what we mean by admissible heuristics.

(Refer Slide Time: 10:23)

## Admissible heuristics

- A heuristic is **admissible** if it **never overestimates** the cost of reaching a goal from any node.
  - According to this definition, even the null heuristic (which returns 0 for any node) is admissible!
- The only thing required for a heuristic to be admissible is that it **never returns a value greater than the actual path cost** to the nearest goal for any node.

© Shyamanta M Hazraika, M.E., IIT Guwahati

So, a heuristics is admissible if it never overestimates the cost of reaching a goal from any node. So, in that case, the null heuristic which returns 0 for any node is also admissible heuristics. The only thing required for a heuristic to be admissible is that it never returns a value greater than the actual cost and to the nearest goal from any node.

(Refer Slide Time: 10:54)

## Admissible heuristics

- A heuristic  $h(n)$  is **admissible** if for every node  $n$ ,  
$$h(n) \leq h^*(n)$$
where  $h^*(n)$  is the **true cost** to reach the goal state from  $n$ .
- An admissible heuristic **never overestimates** the cost to reach the goal, i.e., it is **optimistic**

© Shyamanta M Hazraika, M.E., IIT Guwahati

Let us say  $h_n$  is an admissible heuristic.  $h_n$  is admissible if and only if  $h_n$  is less than equal to  $h^*n$  where  $h^*$  is the true cost to reach the goal from the current  $n$ . An admissible heuristic never overestimates the cost to reach the goal. That is, it is optimistic.

(Refer Slide Time: 11:24)

## Problem Relaxation

7	2	4
5		6
8	3	1

	1	2
3	4	5
6	7	8

Start State                      Goal State

Here is an instance of the 8 puzzle problem. We are given the state on the left side and we want to transform it into the state on the right. The only action we have is sliding a tile up, down, left or right into the empty space. All actions are equivalent so let's assign each action the same cost 1. The cost of a solution is then the number of actions taken and we want to find the cheapest solution.

© Sreyas M Hazrika, ME, IIT Guwahati

So, let us look at problem relaxation and try to understand how heuristics are created. Again, falling back onto the 8-puzzle game; here we have the start state given to you on your left. And on the right is the goal that I want to arrive at. Now, one needs to understand that in an 8-puzzle game, all we can do is move tiles from 1 place to another if and only if there is a blank. So, the only possibility in this game right now is to move 5 this side, move 6 here, move 2 down there or move 3 up.

Those are the only possibilities. But in a relaxed game, I would be able to do more than that. So, we can transform it into any state on the right. The only action is about not sliding them like this; left or right into the empty spaces. So, in a relaxed problem definition, what we would be more interesting to do is to find out how many tiles do I need to pick up basically to give it to this side. So, in this problem, in a relaxed state, if I could be able to pick them up;



(Refer Slide Time: 12:44)

## Problem Relaxation

Start State                      Goal State

Here is an instance of the 8 puzzle problem. We are given the state on the left side and we want to transform it into the state on the right. The only action we have is sliding a tile up, down, left or right into the empty space. All actions are equivalent so let's assign each action the same cost 1. The cost of a solution is then the number of actions taken and we want to find the cheapest solution.

© Shyamanta M Hazra, ME, IIT Guwahati

So, I would love to have 7 picked up and put here. That is 1 operation. I would love to pick up 2 and put there. So, that would be another operation. 1 + 1, 2 operations. I would love to pick up 4, put it here. That would be 3 operations. I would love to pick up 5, put it there; 4 operations. Pickup 6, put it in its place; 5 operations. Pickup 8, put it in its place. Pickup 3, put it in its place. And pick up 1 and put it in its place.

So, I would have 8 operations required to move from this to this in a relaxed problem, where the relaxation is about picking them up. So basically, I was counting how many tiles were misplaced, so that I could pick them up and place it back. I could look at the same thing in a slightly different way. I could assume that these tiles that I have are actually in different 2D planes. In that case, I should be able to individually move them in x and y directions. So, like, let me give you what I mean.

(Refer Slide Time: 14:15)

## Problem Relaxation

Start State

Goal State

Here is an instance of the 8 puzzle problem. We are given the state on the left side and we want to transform it into the state on the right. The only action we have is sliding a tile up, down, left or right into the empty space. All actions are equivalent so let's assign each action the same cost 1. The cost of a solution is then the number of actions taken and we want to find the cheapest solution.

© Shyamanta M Hazarika, ME, IIT Guwahati

Like, if I want to bring 7 which is here, to here; so, I would be able to take 7 from here to here in number of 1 and 2 step. And take 7 from here to here in number of 1 step. So, I would be taking 3 steps and bringing 7 to its position. I would be able to take 2 hear and bring it here in 1 step. So, this is another form of heuristics that we will look at, where instead of counting the misplaced tiles, I am trying to count the distance that is there in terms of moving it along the x and y axis.

(Refer Slide Time: 14:57)

## Relaxed Problem

- A problem with fewer restrictions on the actions is called a **relaxed problem**
- The **cost of an optimal solution to a relaxed problem** is an **admissible heuristic** for the original problem ✓
- If the rules of the 8-puzzle are relaxed so that a tile can move **anywhere**, then  $h_1(n)$  gives the shortest solution.
- If the rules are relaxed so that a tile can **move to any adjacent square**, then  $h_2(n)$  gives the shortest solution

© Shyamanta M Hazarika, ME, IIT Guwahati

So, this is how we will look at a relaxed problem. A relaxed problem with fewer restrictions on the action and is possible. So, cost of an action solution to a relaxed problem is admissible heuristics for the original problem. This is what makes a problem relaxation important. If the rules of the 8-puzzle game as I was trying to explain are relaxed, that a tile can move

anywhere, then I give the shortest solution, because I pick up every tile and put it in its original position.

If the rules of the game are relaxed that a tile can move to an adjacent square, then that gives me again another shortest solution and to give me 2 heuristics.

(Refer Slide Time: 15:42)

### Number of Misplaced Tiles

✓

7	2	4
5		6
8	3	1

Start State

?

	1	2
3	4	5
6	7	8

Goal State

Suppose we could take out a tile and place it in its correct position in 1 move. Given any state, how many moves would it take to reach the goal state? It's precisely the number of misplaced tiles, also called the Hamming distance. In this relaxed version of 8-puzzle, we don't need to use search to find out the solution cost. We can calculate it directly so it can be used as a heuristic.

$h_1(n) = \text{number of misplaced tiles}$  ✓

© Shyamanta M Hazarika, ME, IIT Guwahati

So, let us look at the first heuristic called the number of misplaced tiles. Suppose we take out a tile and place it in its correct position, as I was trying to explain you in the previous slide. Given any state, question is, how many moves I would require to come to this state? It is precisely the number of misplaced tiles. This is also called the hamming distance. In this relaxed version of 8-puzzle, we do not need to search to find out a solution.

We can calculate it directly, so it can be used as a heuristic. This heuristic is called the heuristic of number of misplaced tiles. Let us call it as  $h_1$ .

(Refer Slide Time: 16:40)

## Total Manhattan Distance

5

7	2	4
5		6
8	3	1

Start State

9 ?

	1	2
3	4	5
6	7	8

Goal State

Suppose we could slide a tile into any block, as opposed to only the empty block, in 1 move (Imagine a 3D board, where each tile is in its own 2D plane). Given any state, how many moves would it take to reach the goal state? It'd be the sum of horizontal and vertical distances of each tile from its desired position, also called the **Manhattan distance**. We can calculate this value directly, without requiring search.

$h_2(n) = \text{total Manhattan distance}$

© Shyamanta M Hazarika, M.E., IIT Guwahati

The other heuristic that we were talking of is the heuristic where I assume in a relaxed state that each tile can move in its 2D plane. And these 2D planes are stacked 1 above another to make the whole problem. So, 7 can move from its current position and up there, come down to this level and then move here onto this level. For that matter, 3 can go from here via this to this.

So, suppose we can slide a tile into any block, as opposed to only the empty block and given any state; question is, how many moves would I need to come to the goal state? Now, if you look at it very closely, you could realize that it would be the total of how many steps each of the block requires to be moved. So, like 7 here requires 1, 2 and 3 moves; 3 requires 1 and 2 move to it reaching its position; so on and so forth.

We could calculate the total of these distance. This distance would then give me the total cost required to move from this given start to this given goal state. This distance that I have calculated, the sum of the vertical and horizontal distance of each tile from its desired position is also called the Manhattan distance. So, the second heuristic that we have now named is the total of the Manhattan distances for each of the tiles.

(Refer Slide Time: 18:34)

## Admissible heuristics

E.g., for the 8-puzzle:

- $h_1(n)$  = number of misplaced tiles ✓
- $h_2(n)$  = total Manhattan distance

2	3	4
5		8
6	7	1

Start State

	1	2
3	4	5
6	7	8

Goal State

- $h_1(S) = ?$  8 ✓
- $h_2(S) = ?$  3+1+2+2+2+3+3+2 = 18

© Shyamanta M Hazra, ME, IIT Guwahati

Let us try to understand it for the 8-puzzle problem more better. So, we have now 2 heuristics. 1, the number of misplaced tiles and number 2, the total Manhattan distance. Let us try to calculate each of them. Calculating each of them has been explained previously, but let us try to get that clear here. So, for the first heuristic I count how many tiles are misplaced.

So, if you see, 7 is not in its place, 2 is not in its place, 4 is not in its place; 6, 1, 3, 8, 5, none of them are in its place. So, the total number of misplaced tiles here are 1, 2, 3, 4, 5, 6, 7, 8. So, the number of misplaced tiles is 8. So, the heuristic value is 8. Expectation is that; now, remember what an heuristic value is. It is an estimate from the current state to the goal state. So, expectation is that by taking some 8 moves, I would be able to go from here to here.

Let us look at the second heuristic, the total Manhattan distance heuristic here. So, let us look at each of them one by one. Let us take number 1.

(Refer Slide Time: 19:57)

### Admissible heuristics

E.g., for the 8-puzzle:

- $h_1(n)$  = number of misplaced tiles ✓
- $h_2(n)$  = total Manhattan distance

7	2	4
5		6
8	3	1

	1	2
3	4	5
6	7	8

Start State                      Goal State

- $h_1(S) = ?$  8 ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓
- $h_2(S) = ?$  3+1+2+2+2+2+3+3+2 = 18

© Sreyamsha M Hazarika, ME, IIT Guwahati

Number 1 in the current state, if it has to reach number 1 in the goal state, we have to make a move of 1 step, 2 step and the third step, if I take that route. Or for that matter, 1 step, 2 step, 3 step, if I take that route. So basically, number 1 would need 3 steps to move to its goal position. Number 2 in the current state, would only need 1 move to reach to the goal position. So, here it is. So, only 1 move to reach to the goal position.

Number 3 is here and in the goal it is on the left middle. So, I should be able to move 3 here and go 1 step this side. So, I have 1, 2; 2 steps for 3. For 4, I have, let me clear this so that you can see it more clearly, what is happening for 4.

(Refer Slide Time: 21:09)

### Admissible heuristics

E.g., for the 8-puzzle:

- $h_1(n)$  = number of misplaced tiles ✓
- $h_2(n)$  = total Manhattan distance

7	2	4
5		6
8	3	1

	1	2
3	4	5
6	7	8

Start State                      Goal State

- $h_1(S) = ?$  8 ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓
- $h_2(S) = ?$  3+1+2+2+2+2+3+3+2 = 18

© Sreyamsha M Hazarika, ME, IIT Guwahati

So, for 4, here is the goal position. And here I am in the initial state. So, in order to move to the goal position, I would need 1 step coming down and 1 step going left. So, I would need 2 steps. Or, if I take the other route, I would be like 1 step going left and 1 step coming down. So, 2 is the number for number 4. Number 5 similarly, would be 1 step and 2 steps. Number 6 would be 1 step, 2 step, and a 3; so, 3 steps. Number 7, let us count number 7 here; 1 step, 2 step and 1 step more, so 3 steps. And number 8 is, if you see, has to be pushed to the extreme right.

(Refer Slide Time: 22:01)

### Admissible heuristics

E.g., for the 8-puzzle:

- $h_1(n)$  = number of misplaced tiles ✓
- $h_2(n)$  = total Manhattan distance

7	2	4
5		6
8	3	1

	1	2
3	4	5
6	7	8

Start State                      Goal State

- $h_1(S) = ?$  8 ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓
- $h_2(S) = ?$  3+1+2+2+2+3+3+2 = 18 ✓

© Sreyansha M Hazareka, ME, IIT Guwahati

So, 1 step, 2 step; so 2. So, the total number of moves that is required for the Manhattan distance heuristic is 18. So, the estimate that Manhattan does is that 18 steps possibly would be required if I have to move from the given start to go to the given goal. In fact, later on we will see that neither the misplaced tiles, nor the Manhattan distance has given us the accurate measure.

If you want to play this game, you later realize that this is more than what has been estimated by the misplaced tiles as well as the total Manhattan distance. So, now let us look at what we mean by admissible and what we mean by inadmissible heuristics.

(Refer Slide Time: 23:02)

## Admissible vs. Inadmissible Heuristics

- ❑ **Inadmissible heuristics**
  - ❑ pessimistic heuristics because they overestimate the cost
  - ❑ break optimality by trapping good plans on the fringe.
- ❑ **Admissible heuristics**
  - ❑ optimistic heuristics because they can only underestimate the cost
  - ❑ can only slow down search by assigning a lower cost to a bad plan so that it is explored first but sooner or later search will find the optimal solution.

© Shyamanta M Hazra, ME, IIT Guwahati

Inadmissible heuristics are pessimistic heuristics. They overestimate the cost. And on the other hand, admissible as I have been defining are optimistic heuristics, because they underestimate the cost. Later we will see that admissible heuristics can only slowdown search by assigning a lower cost to a bad plan but not very dangerous. Whereas, inadmissible heuristics break optimality by trapping good plans.

Now, let us recall from our previous lecture what we mean by an optimal solution. An optimal solution is, for the given solutions that I have; if I can get the minimum cost solution, that is an optimal solution.

(Refer Slide Time: 24:00)

## Inadmissibility breaks Optimality

Suppose node A has been expanded and nodes B and C are on the fringe.  
 $f(C) = 3 < f(B) = 5$   
Node C will be expanded next.

$f(\text{Goal}) = 4 < f(B)$   
Goal will be taken off the fringe!  
More expensive path A-C-Goal (cost 4).  
Instead of cheaper path A-B-Goal (cost 3).

This happened because our heuristic function overestimated the cost of path B-Goal.

© Shyamanta M Hazra, ME, IIT Guwahati



So, let us look now more closely on how inadmissible heuristics breaks optimality. Suppose node A has been expanded and I get 2 nodes, B and C on the fringe. Now, if you look at the cost of the nodes C and B and try to understand which node to expand; you will see that this cost is made up of 2 different costs. One, which is the exact cost of the current node that I arrive at from the start node.

And the other is an estimate of how much I need to spend to go to the goal. At A, because it is the start node, the actual cost of coming to this node is 0. Whereas, the heuristic estimate is given as 5. The branch that is coming to C, the cost is 1. Whereas the branch that goes to B, also has a cost 1. So, if I compute the total cost at C, it will be like the cost of coming to C which is 1 and the heuristic estimate of going from C to the goal which is 2;  $1 + 2$ , that is 3.

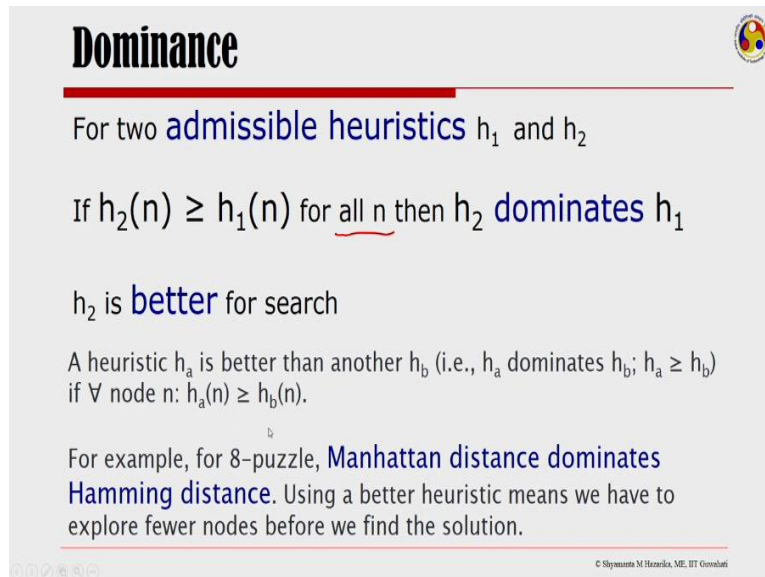
And if I look at B, the cost is of course 1, but the heuristic estimate that I make of going to the goal is 4; making the total cost of node b as 5. So, in this current scenario, node C will be expanded next. So, we expand node C. That is what will happen. So, when node C is expanded next, what will happen is that, I will arrive at goal G. And now, this goal will be taken of the fringe. Because, now I will see that I have arrived at a goal.

The total cost of the goal that I will get in this case is the total cost of coming by this path which is  $3 + 1$ , 4. If you now look at this problem very closely, you see that, because I have used an heuristic which give 4 here and 2 here, I got a path which the total cost was 4. Whereas, on the other side, I did have a path where the total cost was less than 4, which is 3. Question is, why did not I get this path and got this path?

This is precisely because of the following reason. We have estimated here that the cost to the goal would be 4, whereas the actual cost was only 2. We made an overestimate. Making an overestimate spoiled this path for us. And therefore, we were not able to get the optimal solution. A more expensive path which is of cost 4 was given to us instead of the cheaper path which is A B goal which is at the total cost of  $1 + 2$ , which is 3.

This happened only because our heuristic function that we have used overestimated the cost of the path B to goal. Any heuristic function that overestimates is a pessimistic heuristic function and we call it an inadmissible heuristic. An inadmissible heuristic therefore breaks optimality, does not lead to a path which is the minimum.

(Refer Slide Time: 27:53)



## Dominance

For two **admissible heuristics**  $h_1$  and  $h_2$

If  $h_2(n) \geq h_1(n)$  for all  $n$  then  $h_2$  **dominates**  $h_1$

$h_2$  is **better** for search

A heuristic  $h_a$  is better than another  $h_b$  (i.e.,  $h_a$  dominates  $h_b$ ;  $h_a \geq h_b$ ) if  $\forall$  node  $n$ :  $h_a(n) \geq h_b(n)$ .

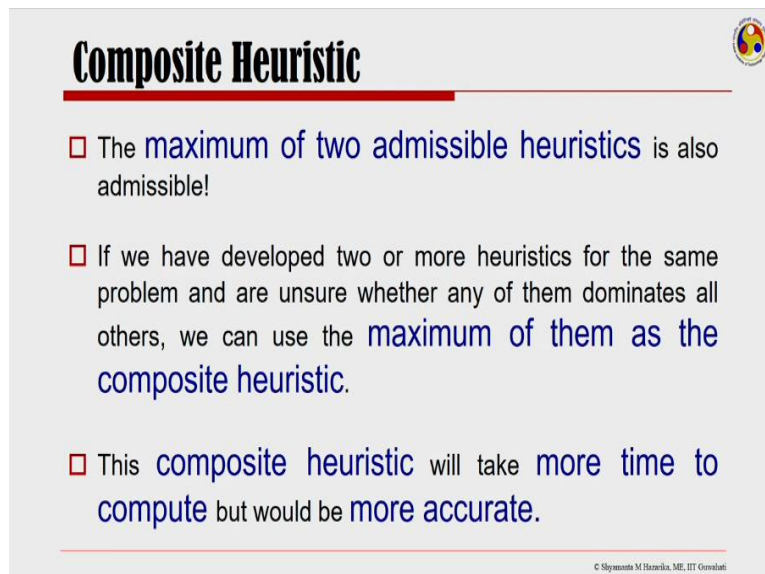
For example, for 8-puzzle, **Manhattan distance dominates Hamming distance**. Using a better heuristic means we have to explore fewer nodes before we find the solution.

© Shyamanta M Hazra, ME, IIT Guwahati

Let us now understand what we mean by dominance of heuristics. Let us say we have 2 admissible heuristics,  $h_1$  and  $h_2$ . If for any node  $n$ , all these nodes that I am talking of, the heuristic value  $h_2$  is more than; for 2 admissible heuristics  $h_1$  and  $h_2$ ; if  $h_2(n) \geq h_1(n)$  for all  $n$ , then  $h_2$  dominates  $h_1$ ;  $h_2$  is better for such.

A heuristic  $h_a$  is better than another heuristic  $h_b$ ; that is  $h_a$  dominates  $h_b$ . If all nodes  $h_a$  is more than equal to  $h_b$ . For the 8-puzzle game we had highlighted 2 heuristics, if you remember. The total of the Manhattan distances and the total of the misplaced tiles. The Manhattan distance dominates the hamming distance heuristics.

(Refer Slide Time: 29:12)



## Composite Heuristic

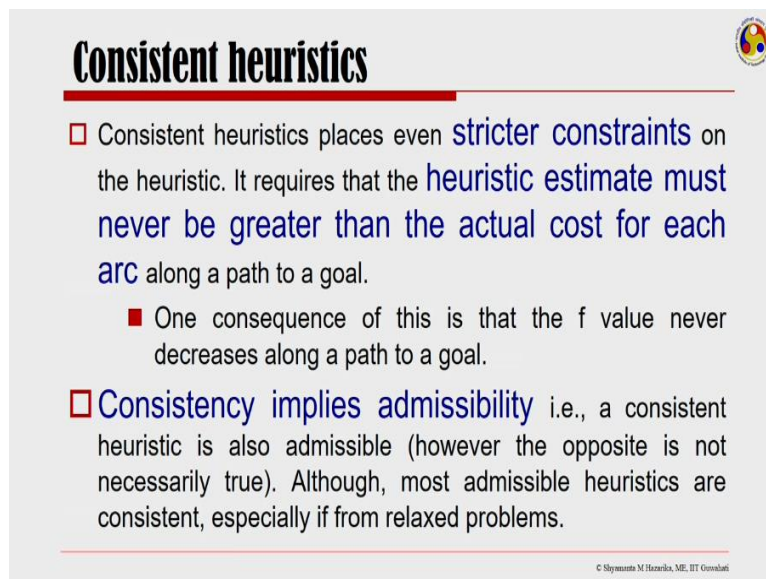
- The **maximum of two admissible heuristics** is also admissible!
- If we have developed two or more heuristics for the same problem and are unsure whether any of them dominates all others, we can use the **maximum of them as the composite heuristic**.
- This **composite heuristic** will take **more time to compute** but would be **more accurate**.

© Shyamanta M Hazra, ME, IIT Guwahati

Let us now move on to understand what we mean by composite heuristics. We have, let us say a couple of heuristics, we do not know which heuristic dominates which other. Then, the maximum of 2 admissible heuristics is also admissible. Going by that, if we develop 2 or more heuristics for the same problem and we are unsure whether any of them dominates all others, we can use the maximum of them as the composite heuristic.

This composite heuristic will take more time to compute, but would be more accurate. This is what makes composite heuristics interesting.

**(Refer Slide Time: 29:58)**



**Consistent heuristics**

- Consistent heuristics places even **stricter constraints** on the heuristic. It requires that the **heuristic estimate must never be greater than the actual cost for each arc** along a path to a goal.
  - One consequence of this is that the  $f$  value never decreases along a path to a goal.
- **Consistency implies admissibility** i.e., a consistent heuristic is also admissible (however the opposite is not necessarily true). Although, most admissible heuristics are consistent, especially if from relaxed problems.

© Shyamanta M Hazra, ME, IIT Guwahati

Let us now look at more closely another heuristic that is referred to as consistent heuristic. Consistent heuristics are those which places even stricter constraints on the heuristic. It requires that the heuristic estimate must be greater than the actual cost for each arc along a path. So, 1 consequence of this is that, the final value that I compute for a node to reach to the goal never decreases along a path to the goal.

Consistency therefore implies admissibility. That is, a consistent heuristic is also admissible. However, the opposite is not necessarily true. Most admissible heuristics are consistent, especially if from relaxed problems.

(Refer Slide Time: 30:51)

## Consistent heuristics

A heuristic is consistent if for every node  $n$ , every successor  $n'$  of  $n$  generated by any action  $a$ ,

$$h(n) \leq c(n,a,n') + h(n')$$

If  $h$  is consistent, we have

$$\begin{aligned} f(n') &= g(n') + h(n') \\ &= g(n) + c(n,a,n') + h(n') \\ &\geq g(n) + h(n) \\ &= f(n) \end{aligned}$$

i.e.,  $f(n)$  is non-decreasing along any path.

© Shyamanta M Hazra, ME, IIT Guwahati


So, let us look at what we mean by consistent heuristic. Let us say we have the cost to a node given to us, from  $n$  to this given node. Or if the total cost to the goal is  $h(n)$  and  $h(n)$  can be looked at given another successor node or  $n'$  of  $n$  generated for any  $n$ .  $h(n)$  would be the cost of coming from  $n$  to  $n'$  because of an action  $a$ . And then going down by an estimated  $n'$  which is  $h(n')$  and  $s$  to  $g$ .

If  $h$  is consistent, we have the total cost of arriving at the goal from  $n'$  to be the cost of coming to  $n'$  which is  $g(n')$ . And then, the estimate of going from  $n'$  to  $g$  which is  $h(n')$ . But we have already seen that  $g(n')$ , the total cost is about the total cost that  $n$  plus the cost because of action  $a$  on  $n$  to  $n'$ . So, that we can substitute here.  $g(n')$  getting substituted by  $g(n) + c(n,a,n')$ .

And  $h(n')$  is the estimate that is already added. Now, given  $g(n) + c(n,a,n') + h(n')$  to be greater than or equal to  $h(n)$ , the  $f(n')$  that we have is actually greater than equal to  $g(n) + h(n)$ . And this is nothing but  $f(n)$ , the cost of node  $n$  of coming here. So,  $f(n)$  we see is non-decreasing along any path. So, let us look back on what we said about consistent heuristics that it put stricter constraints no doubt. But once we have that consistent heuristic, the  $f$  value never decreases along a path to the goal.

(Refer Slide Time: 33:05)

## Heuristics – a Trade-off




- Heuristics have a **trade-off between quality of estimate and work per node**. As heuristics get closer to the true cost, we expand fewer nodes but usually do more work per node to compute the heuristic itself.
- At the **two extremes** are the **null heuristic** (which always returns 0 and requires the least amount of work) and the **exact cost** (which can only be found out by conducting search itself and requires the most amount of work).

© Shyamanta M Hazarika, ME, IIT Guwahati

Finally, we want to take 2 notes that a heuristics is actually a trade-off between quality of estimate and work per node. As heuristics get closer to the true cost, we expand fewer nodes. But usually do more work by node to compute the heuristic itself. At the 2 extremes are the null heuristic which always returns 0 and requires the least amount of work. And the exact cost which can only be found by conducting search itself and requires the most amount of work.

(Refer Slide Time: 33:42)

## Choosing a Good Heuristic



- For a given problem, there might be many different heuristics one can choose. However, some heuristics are better than others.
  - We say the **heuristic is better if it needs few nodes to examine** in the search tree.
  - We also call this kinds of heuristics are better informed.
- Efficiency is Very important in Picking a Heuristic.
  - It is very **important to consider the efficiency of running the heuristic** itself.
  - More information about a problem may mean more time to process the information.

© Shyamanta M Hazarika, ME, IIT Guwahati

Choosing a good heuristic is therefore a very vital problem. For any given problem there might be many different heuristics one can choose. However, some heuristics are better than others. We say heuristics is better if it needs few nodes to examine in the search tree. We also

call these kinds of heuristics are better informed. Efficiency is very important in picking a heuristic. It is very important to consider the efficiency of running the heuristic itself.

More information about a problem would mean more time to process the information and maybe more inefficient heuristics. So, while we look at informed search strategies in our next lecture, we would also once again look back on heuristics. Today, the lecture was on understanding what heuristics are, what is the motivation of using heuristics and giving a couple of properties: a. what do we mean by admissible heuristic? b. what is dominance? c. what do we mean by a consistent heuristics? Thank you very much.