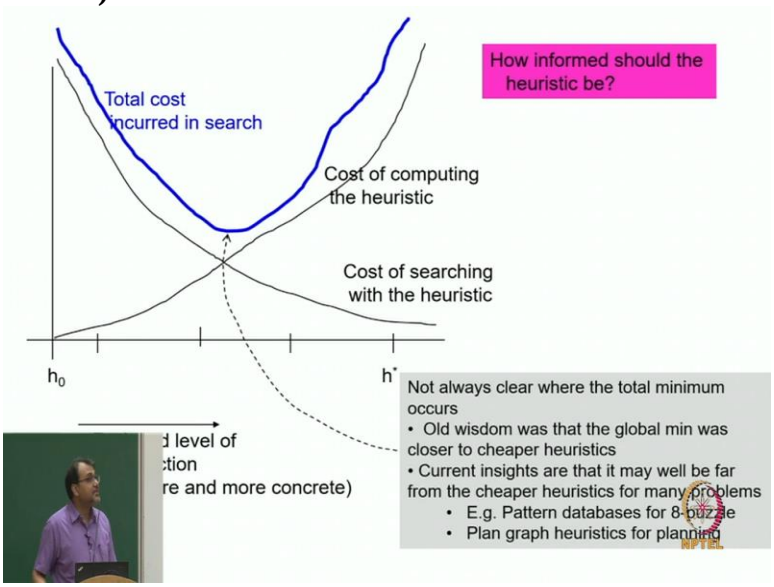
sArtificial Intelligence
Prof. Masum
Department of Computer Science and Engineering
Indian Institute Technology Delhi

Lecture-21
Informed Search: Pattern Database Heuristics*-Part 7

We should get started, I have 2 other things I would like to do, I would like to a talk about this very important phenomenon, which is how expensive should my heuristic be? That is actually an important question that we need to answer.

(Refer Slide Time: 00:31)



So in this graph, this is not a real, you know, graph drawn by empirical data, but this is intuitive graph. So on the y axis; we have the cost of computing the solution, so it is running time. Let us say it is basically running time. And I am showing you 2 graphs. And what I am wearing on the x axis is how informative the heuristic is. So most informative heuristic is the as we have discussed h star heuristic, and most uninformative heuristic is the h 0 heuristic.

So what h star, I know the exact optimal solution cost, and what is h 0, I know nothing, and I feel I am the goal. And so ideally what will happen is that the cost of computing the heuristic is going to increase, obviously. Because here I have to do no work whatsoever, I would say every heuristic is 0, that is 0 work, and so I will not spend any time in actually doing the computation. If I have to do h 1, then I have to at least count the number of misplace styles.

If I have to do h 2, then not only do I have to count the number misplaced, as I have to add the distances and figure and add them a little bit more work not too much. Of course, like in the previous example, if I have to do shortest path that it is been kind of an algorithm, if have to do minimum spanning tree then it is a different kind of an algorithm. I may end up doing something even better that it might be even more expensive algorithm.

So, what is happening is that as I increase the informativeness of the heuristic the cost to compute the heuristic increases and you can sort of assume that it increases somewhat exponentially. Because eventually the problems that we are solving and be hide at least and therefore, in practice, you know, they will be more exponential at least today. So, this cost is going to go up very fast.

On the other hand, if I give you a good heuristic, your cost of searching is going to reduce we just discussed that A star with h 1 to 39000 expansions with A star with h 2 only 6000. So, therefore, the cost of actually searching is going to reduce and then you have to ask the question, well, what is the right heuristic to compute and, in fact, the total cost that you will finally incurred is going to be something like this.

This will be the total cost, because in total cost, you also have to do heuristic computation plus you also have to do search. And so it feels like the right point, the right amount of heuristic computation is something like this. Of course, this graph is never given to you, you have to use some intuition, but this is an important intuition. That terrible heuristic is terrible but really good heuristic is also terrible.

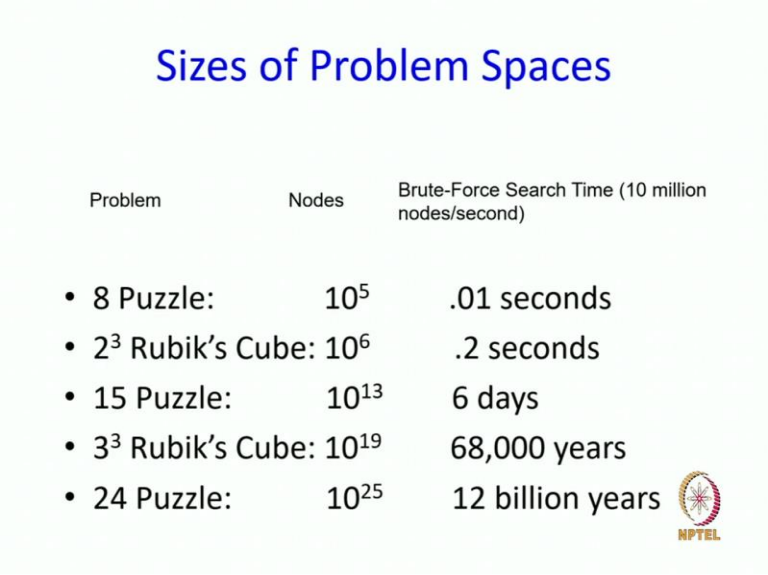
The really good heuristic is terrible because you will spend a lot of time computing it even if you end up spending your time searching for it and vice versa. So, therefore, but the question is where does this minimum occur? And, you know, search, researchers have sort of asked this question about sometimes, and sort of the conventional wisdom was that a global minimum is close to cheaper heuristics.

In practice, you always want your heuristic to be polynomially computable because if the heuristic itself is exponentially computable, then it is a much more expensive algorithm. But even the polynomial algorithm, what this is saying that the conventional wisdom is that not only should it be polynomial it should be lower order polynomial, you know, maybe linear advanced quadratic, almost never do you take time algorithms for us to competitions in practice.

But, there are some new insights that have come out in the last 10, 15 years, which have made people question this conventional wisdom. And in fact, people have found that cheaper heuristics may not always be the best idea, sometimes somewhat more expensive heuristics may be a better idea and the examples there is pattern databases that we will study and plan a heuristics. That we will not study planning after the heuristics are actually very important set of heuristics for planning problems.

I wish we could discuss this it is an area I did my PhD in so it is close to my heart, but we will not talk about it. But I will briefly talk about pattern database heuristics because it is an easy thing to learn, and it gives you a lot of value. So let us quickly talk about pattern database heuristics.

(Refer Slide Time: 05:24)



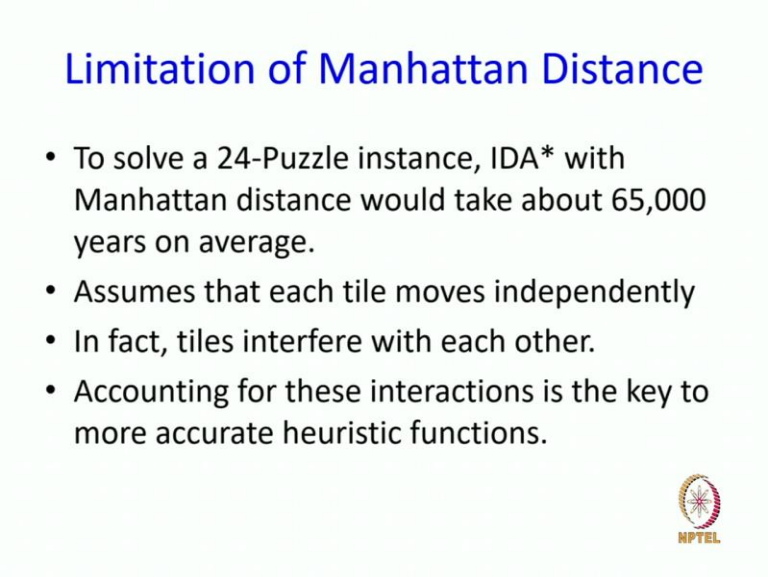
So let us say so and some of these slides have been taken by Richard Gauze who is now very senior, but also extremely well known and famous researcher in the field of search. He was at UCLA; he still is probably at UCLA. And so he says that look, this is the size of search space. If

I am doing Brute Force search, of course, we do not want to Brute Force search but just for completeness 8 puzzles is only 10 to the power of 5 nodes anyway.

So it should be completed in about 0.0 and second not take these numbers with a grain of salt because these are older slides and 1015 year old slides, so somebody would have changed a little bit. But when we are talking about 2 billion years, it is possible that the numbers have changed. But they have gone to 2 million years. So we do not care. It is still equally bad, we cannot do this. So 8 puzzle 0.01seconds Rubik's Cube, 2 by 2 Rubik's Cube 2 by 2 by 2.2 seconds, 3 by 3 by 3 Rubik's Cube.

If you do not give me any information, if you do not give me an admissible heuristic. Or any heuristic 68000 years, that is why Rubik's Cube supposed to be a difficult problem. At least 24 puzzle just an extension of 8 puzzle 25 cross 5 12 billion years. Of course, you would say who cares for you know, Brute Force let us talk about IDA star with some heuristics.

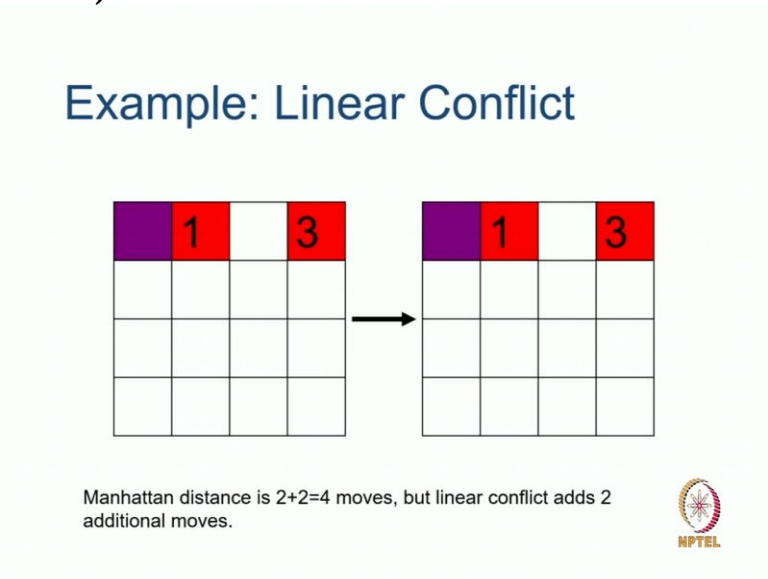
(Refer Slide Time: 06:48)



So here is performance of IDA star 15 puzzle random 15 puzzle instances solved by IDA star using Manhattan distance heuristics not bad optimal solution length is about 53 moves 400 million nodes are generated on average, average solution time is about 50 seconds when they say current machines. I do not mean current, I mean 10 years ago, or so. But what about 24 puzzle? So here 15 puzzles was 6 days 24 puzzle was 12 billion years 15 puzzle with Manhattan distance becomes 50 seconds. 6 days to 50 seconds.

Awesome, but what about 12 billion years to what in 24 puzzles. So let us talk about Manhattan distance where 24 puzzle. IDA star with Manhattan distance would take about 65000years, on average. Now, it is great. It is much better than 12 billion years. But you would not want to start a process right now. And wait for it to finish someday. Of course, what is the problem the problem is that the heuristic is good, but not great. It is not modeling a lot of things for example.

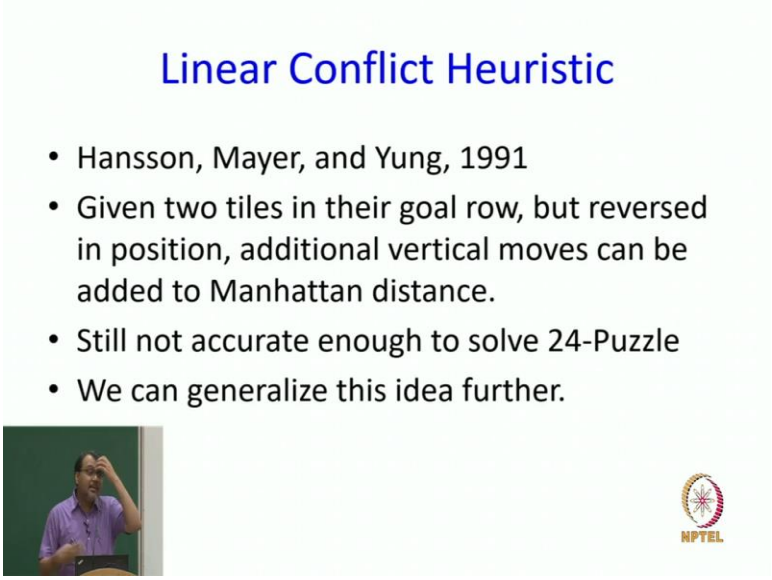
(Refer Slide Time: 08:06)



Let us think about a place where 3 1 needs to go to 1 3. What would Manhattan distance a 4 tiles 1 and 3, 2 and 2. So, the contribution to Manhattan distance for these 2 is 4. Can you ever do it in 4 you can never do it in 4. And the reason you can never do it in 4 is because there is a linear conflict in order to move through their shortest paths, they have to interact and once they have to interact 1 has to go around the other.

So in practice, you will need at least 6 you will need at least 6 why because you will move 3, then you will move on, then you will move on, then you will move on. Then you will move on, then you will move 3 or any which way in this order, but you cannot get to the other position until you move the first one and I will go around it. So if you add some kind of a linear conflict in your heuristic computation, then you will get to a more a better number.

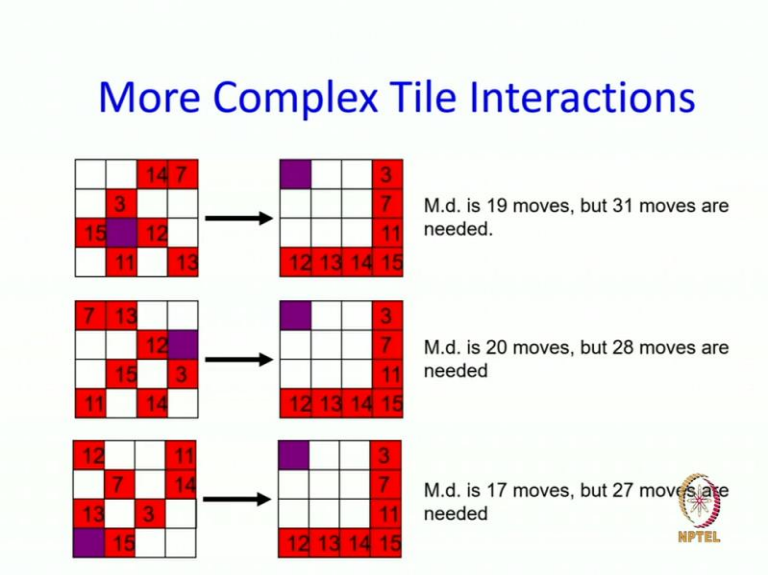
(Refer Slide Time: 09:20)



And more informed heuristic takes so this was developed in 1991 was given 2 tiles in the goal row but reversed in position, additional vertical moves can be added to Manhattan distance. Unfortunately, even though it is more than inform, it is still not accurate enough to solve the 24 puzzle. So what do we do? Now we understand the intuition. The intuition is that look, we were considering these tiles to be completely independent.

They could move on top of it, of course, they should not be allowed to move on top of it, they should only be allowed to move and there is a gap or at least if there are conflicts. And interactions those interactions should be taken care of. But unfortunately, if we start to take care of all interactions, then you will end up solving the original problem which will be extremely expensive to solve. And we have not gained anything. So, we have to carve this balance where in the heuristic computation we do something fast, but we still make it as informative heuristic.

(Refer Slide Time: 10:13)



Many more examples of this show up for example, if I have this set of red tiles, and I want to get to this set of red tiles, the Manhattan distance will say it will take 19 moves, but actually 31 moves are needed, look how bad sometimes this heuristic computation can get. Similarly, if I have a configuration like this and I have to get to the same configuration, Manhattan distances 20 but actually 28 are needed.

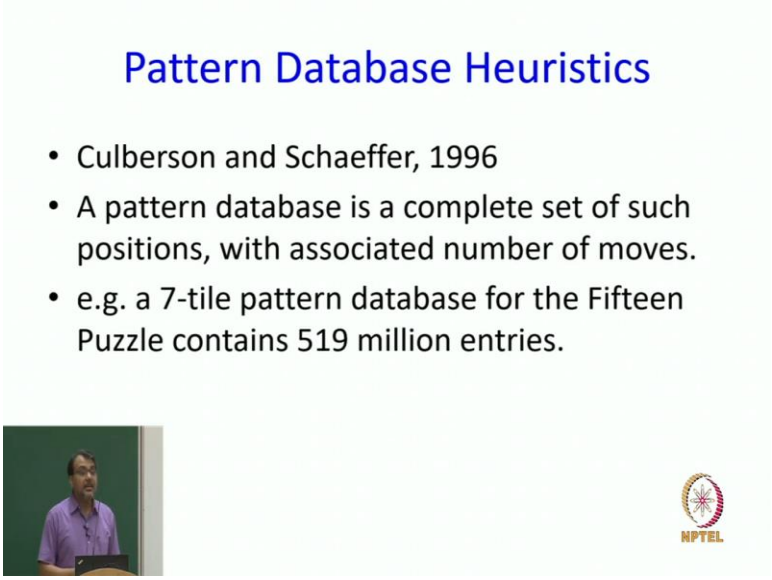
And if I have to go from this configuration to this Manhattan's distances 17, but actually 27 moves are needed. Now notice we have done something very interesting. I am showing you that tiles, 8 of them or 7 of them, but every other tile is not important. I have just called it white; I do not care for the number. Is these are the relaxation? I am in making sure that all the interactions of the red tiles are counted.

But I care for the final position of the whites that is also a relaxation of the original problem. So if I actually solve this problem without completely ignoring what is happening to the whites, the optimal cost that I will get will still be admissible. And that is what is called a pattern database heuristic. Unfortunately, just solving this is going to be expensive. So we will do this expensive operation once. And all the heuristic values will put in a database.

Here is where your database class comes in handy. And then you do an index on top of it so that your database is now quickly randomly accessible. And now when you actually do search you

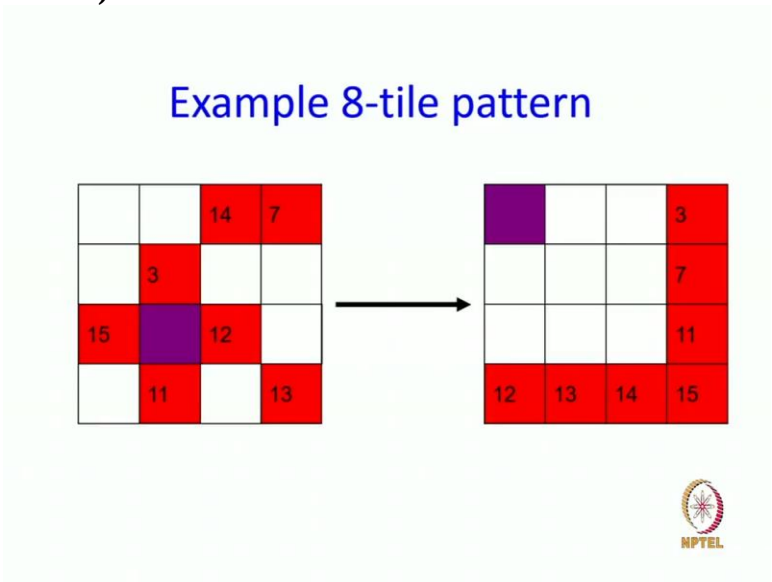
ask the database for the heuristic value you have cashed all the heuristic values you are not doing an on the fly competition.

(Refer Slide Time: 12:10)

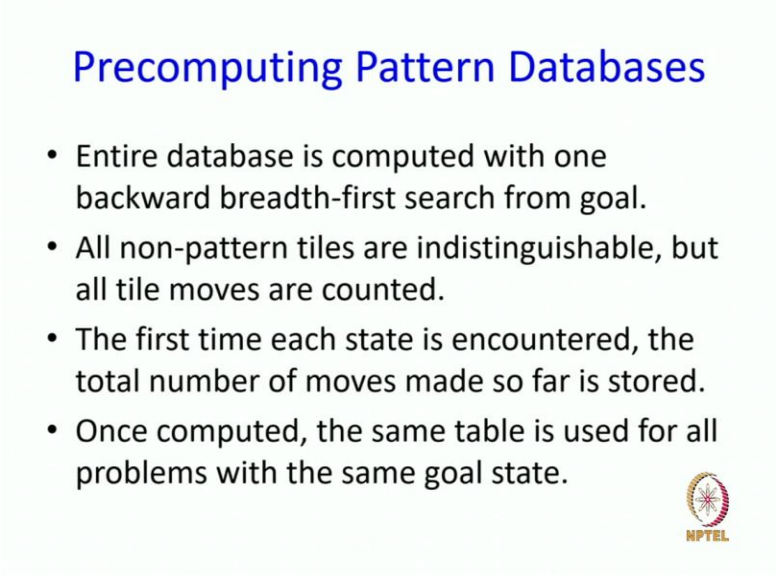


This is called pattern database heuristic it was developed in 1996. A pattern database is a complete set of a set of such positions with associated number of moves. Example a 7 tile pattern database like the one that I just showed you, for 15 puzzles contains 519 million entries. So for all these 500 and19 million states, I compute the patented the heuristic once and in fact, I do it in a batch so that I can make use of the property that many of these states come together and so on, so forth. And so I can, I do not have to do the computation. I can just compute it in one pass.

(Refer Slide Time: 12:53)



(Refer Slide Time: 12:57)

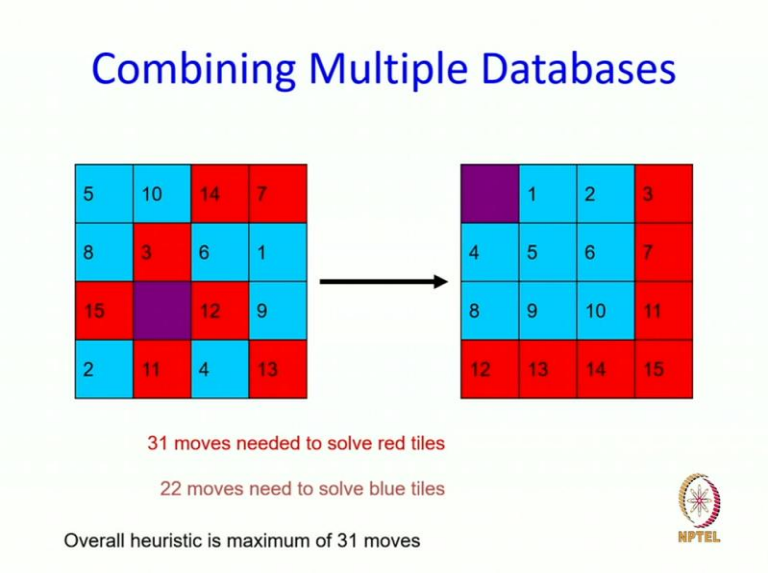


So, this would be one such a tile pattern, so entire database is computed with the one backward search that first search from the goal. By the way, I did not discuss in the last time but as we have A star in the forward direction, we can also do A star in the backward direction. And if you can also do A star in the backward direction, I can also do bi directional a star. And in fact, some of the one of the very recent, like 2, 3 years ago.

Best paper award at a top AI conference was understanding the details of bi directional A star algorithm. So believe it or not A star which came way back in 60s, there were things to be learned about that are bi directional version of that even till the last few years back with the best paper award, it was a fundamental contribution in the field of AI. So if you are interested in bi directionally A star check, check that out.

And note that here, all the non pattern tiles the whites are indistinguishable, but I am counting its moves I am not saying that you know, I will not count your moves, I will count the moves, I just do not care for the final position. Now once I have created this pattern database. I cannot just create one, but I can create many pattern databases. For example.

(Refer Slide Time: 14:16)



I convert my 8 puzzle problem into 2 sets of tiles, the red tiles and the blue tiles. And I have one pattern database for the 7 tile red tiles, and I have another pattern database for the 8 blue tiles. And now my pattern database number that says 31 moves are needed. And my pattern database blue says 22 moves are needed. So what will be the overall admissible heuristic? Is a good question I gave you 2 heuristics?

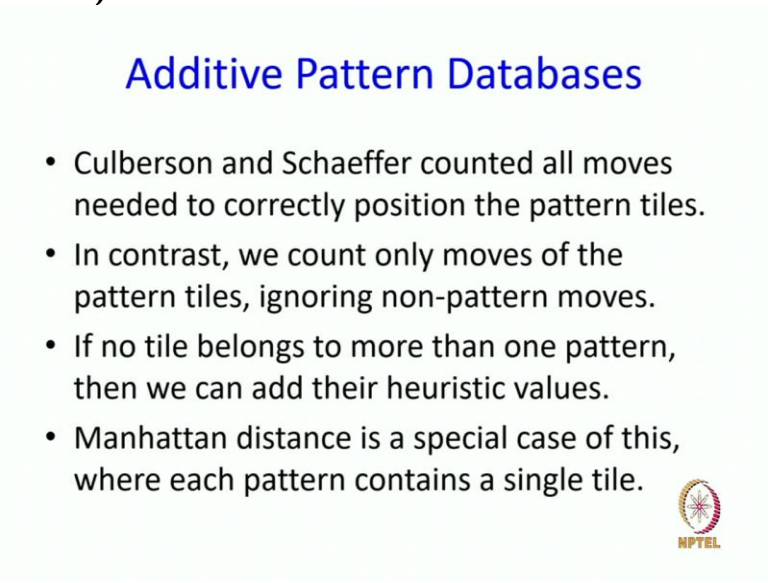
The first heuristics was counting red tiles carefully counting blue tiles as if they are indistinguishable but counting their moves. I gave you the other heuristic h 2 which counted blue tiles carefully counted the red tiles as if they are indistinguishable but counted its moves. If red says 31 in blue says 22, what is the best admissible heuristic you could come up with the max. Why? Because if I do the sum, I will be double counting possibly, because I have counted some of the greyed out tiles moves.

They may be repeated when they want to get to the current correct position. So, therefore, the overall atmosphere heuristics we can come up with is 31 moves. And people were not happy with this. Intuitively, you want to break your problems such that we finally get to the sum and that is admissible. Can you guess how you would do it with this red tile and blue tile setting with 31 you add 7 why would you add 7 to 31?, but that is and interactions.

It is possible that in the process of settling with the red tiles these blue quote unquote indistinguishable tile came in the right position. Then we will not be able to add anything there. So that something else yes, Shiva if we will first settle with the red tiles. And then come up with the positions of blue tiles but look at the time of solving red tiles blue tiles are indistinguishable. We make choices as if they are indistinguishable.

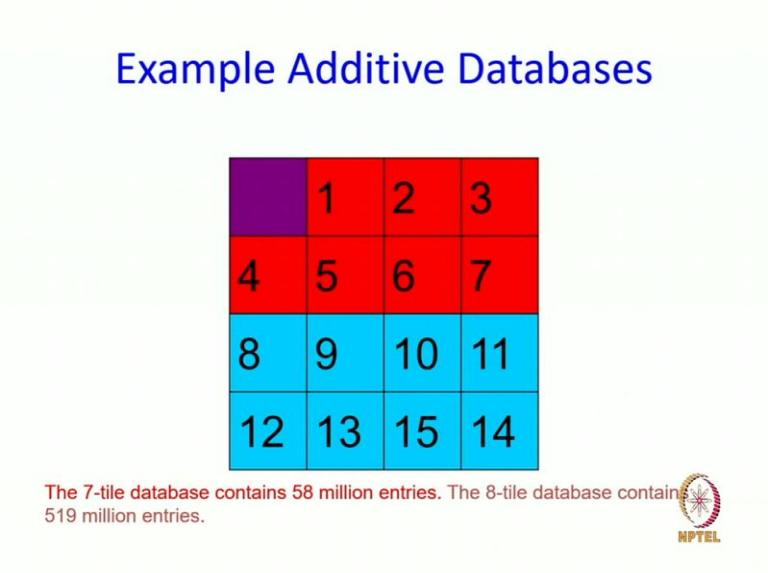
In fact, if we knew the numbers, the state space will become so large that we will not be able to solve this. We have divided it so that we can solve the status. There is another trick. Yes? What is the name? Yes, while moving that guys do not count blue tiles at all, only count the moves of the red tiles allow you to make as many blue tile moves as you want, just do not count it. So individually you will come up with a lower heuristic value, but then it will be additive because you are never doing double counting you and never even counting the blue tile most.

(Refer Slide Time: 17:42)



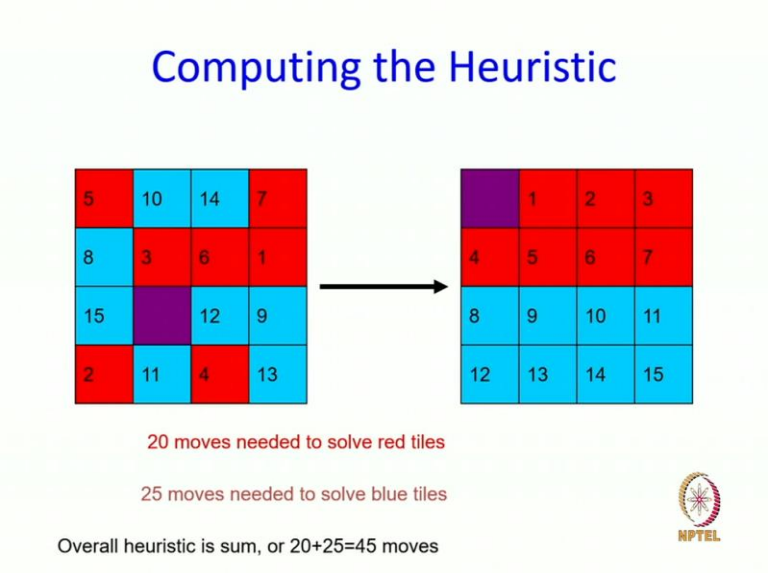
So this is what is called additive pattern databases. We can only count moves of pattern tiles ignoring the non pattern moves. And if no tile belongs to more than one pattern like red and blue are separate. Then we can add heuristic values. In fact, Manhattan distance is a special case of this, where each pattern contains a single tile. Everything else is greater because you are not counting the other source.

(Refer Slide Time: 18:14)



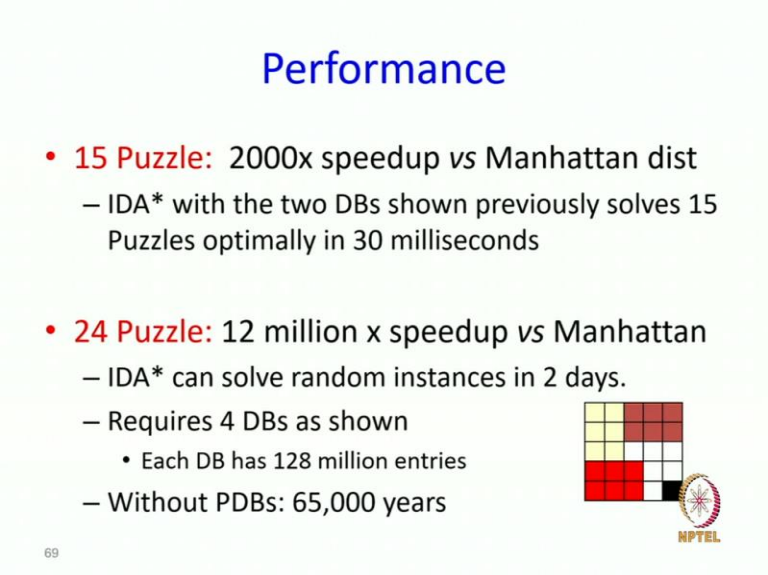
So you can divide it in any which way you can say these are my red these are my blue or whatever. In this case, the 7 tile database will contain 58 billion entries in the 8 tile database will contain 500 and 19 million entries.

(Refer Slide Time: 18:24)



And for example, if the blue database says that the 20 moves are needed to solve red tiles and 25 moves and needed to solve blue tiles, then you can come up with a better heuristic 45 which is admissible.

(Refer Slide Time: 18:37)



So now just the final slide to show you that it works. This gives you 2000x speedup vs Manhattan distance. IDA star with the 2 databases shown previously solved 15 puzzles optimally in 30 milliseconds. Maybe it is not counting heuristic the computation time so that is not fair but still it is fast, but Interestingly what happens in 24 puzzle whereas you would have needed 65,000 years earlier, it can solve a random instance in 2 days.

2 days is still not very good, but at least it is much better than 65,000 years. And also we are talking about you know, 15 year old papers. So, probably now we can solve it in a few seconds. For 24 puzzles they use 4 databases additive using this 6 tile, this 6 tile, this 5 tile and this was it this 6 tile and the 6 tiles so 4 6 tile databases. And that sort of gives us the final result that we were looking for. So, let me summaries.

So, we have been discussing this whole idea of search algorithms. And we have now come to a nice landmark point where we have studied a lot of search algorithms which are going from the start state and developing a tree and sort of maintaining part of the tree in my memory or visiting it and backtracking it and things like that, there is memory implications, there are time implications whether you maintain the frontier or you do depth first.

We have also figured out how to use addition heuristic function additional information to do things faster. And we have done several examples to think about how quickly they can solve

some problems. If you have a good heuristic, we have discussed how to compute good heuristics, at least admissible heuristics you require domain relaxations, and then these relaxations can sometimes become large.

So, sometimes you want to do them in a batch, compute all states and then put them in a database and sometimes you will state space is too large you cannot do this. So, you break the state space into multiple sub parts, compute heuristic on top of them, put it in multiple databases and max or add them. These are called pattern databases. So, this is our story up to now and in the next segment in the next class starting tomorrow.

We will talk about local search a very different but in some ways, in my opinion, more practical, even more practical than these algorithms. A different mechanism for doing search and solving problems will stop you.