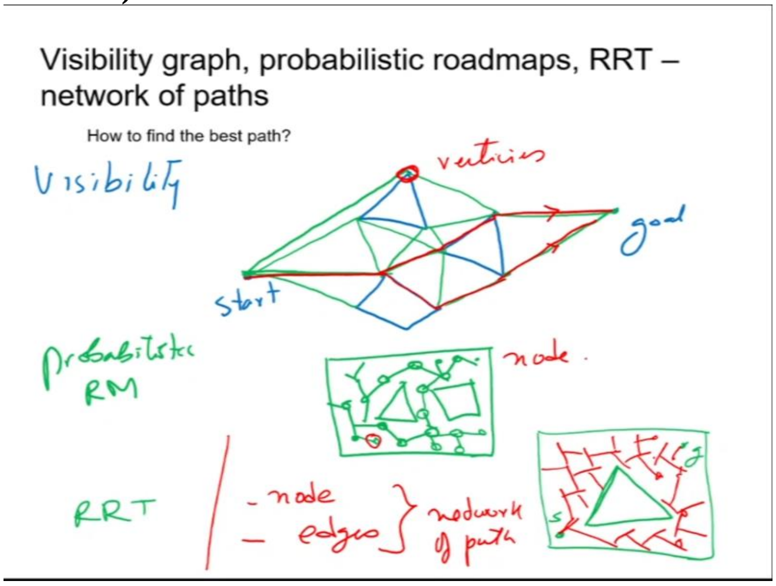
Robot Motion Planning
Prof. Ashish Dutta
Department of Mechanical Engineering
Indian Institute of Technology – Kanpur

Lecture - 15
Basic Search Algorithms

Hello and welcome to lecture number 15. In the last class, we looked at the potential field method in 2D and then we saw how to use the potential field method in 3D applications. And then we looked at navigation functions also. So, today we will move on to the next topic, which is basic search. That means whatever we have done till now that we have used things like visibility graph, then probabilistic roadmap methods in which we found a network of paths.

But now, when you found a network of paths you need to find out which is the best path. Now, how do you know which is the best path now, what do you mean by the best path probably the shortest path, so today we will look at basic search algorithms and then we will move on from here. So, today we will be looking at basic search and then we will look at trajectory planning.

(Refer Slide Time: 01:01)



Visibility graph, probabilistic roadmaps, RRT – network of paths

How to find the best path?

Now, what we have been doing so far is we looked at various methods like roadmaps methods in which we had a visibility graph method in which you found a network of paths to take you from initial point to a final point, then we looked at probabilistic roadmap methods, then we looked at RRT, now all of these methods, for example just very quick revision of what we have done is in visibility graph method.

What we have done is we have a number of obstacles and what we do is we connect these obstacles by straight lines. So, these are obstacles this is my start point and this my goal point just for example, And what we do is we basically connect each of these vertices by straight lines that is what we call the visibility graph method. So, basically what we are doing is we are getting a number of pathways we are trying to generate pathways that will take it from the initial point to the goal point.

Now, this is approximately what we have been doing; now what we see is that there can be more than one path. So, for example, in this case we have one path which can go like this so, there are more paths actually, there can be large number of paths, there is one path this way also. So, what we see is that there can be one path that will go like that is going to go from here like this, like this, like this and like that.

There can be another path which can go from let us say from this side like this, like this, like this. So, these are our paths, now, if there are so many paths, then we need to choose which is a good path or which path should we follow, so as you can see that there are a number of paths a very large number of paths. And now the question comes which one would you follow. Similarly, if you look at probabilistic roadmap.

So, in probabilistic roadmap methods, what we do is again we have, let us say a workspace like this in which we have a number of obstacles. So, there is an obstacle here, there is an obstacle here and my start point is here and my goal point is there for example and what we do is we generate a number of paths. So, these are the various nodes. So, basically, we generate nodes which are possible.

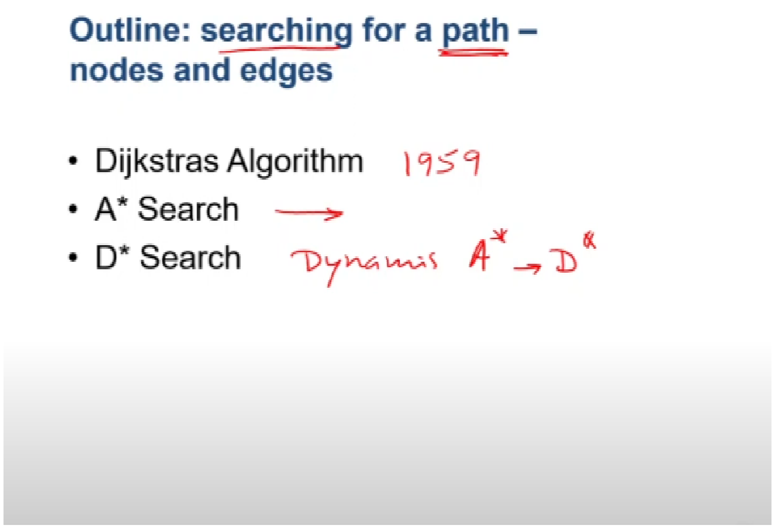
And then we connect nodes by edges and see whether I can connect it from the goal point to the, so these are nodes and 2 of the nodes are connected by edges, but in all these cases, what we see is that we are making a network of paths, then the question comes which path am I going to take. Another example was we looked at RRT, rapidly expanding random trees, in which case we had let me draw it here, just a very quick revision.

Where we had maybe an obstacle here like this and this was my start point. That is my goal point. So, the start and that is goal, what we do is we grow a tree from here. So, we grow a tree like this and these are branches which are going out, so these are various branches of the tree which are going out. And then we try and see whether we can connect the nodes or we can connect, so these are different branches which are going out.

And then we try and see whether we can connect the goal point to the start point by following some path. Again, here, you see that there are a lot of paths. So, the question comes that, how do you search for a path not to 2 terms that we are using here number one is a node, so each of this is a node, that means a robot, it is possible for the robot to be at that point. So, we can call a node. In this case, we are calling vertices. So that is a vertex. So, the node and a vertex essentially mean the same thing here. And then nodes are connected by vertical paths, pathways or edges.

So, let us call them edges, so nodes and edges, so nodes and edges makes network of paths. So, we have nodes and we have edges. And this collection of nodes and edges are make up what we call a network of pathways.

(Refer Slide Time: 05:01)



Outline: searching for a path – nodes and edges

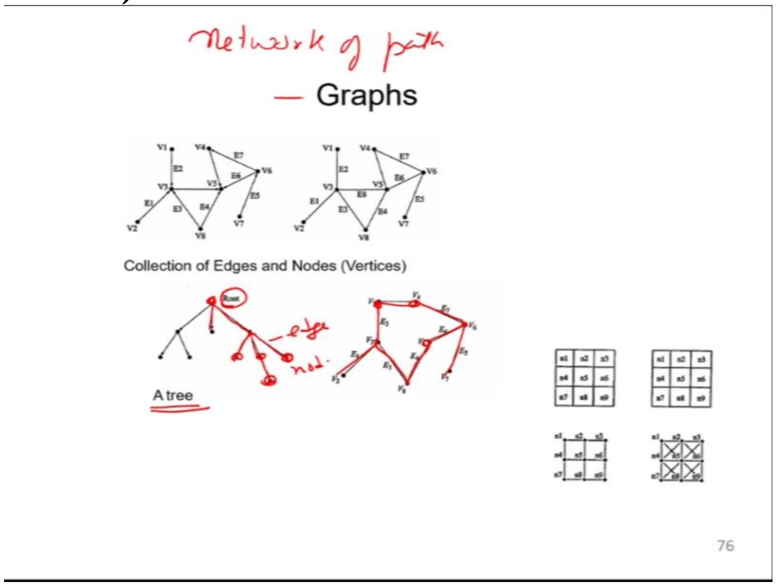
- Dijkstras Algorithm 1959
- A* Search $\longrightarrow$
- D* Search Dynamis $A^* \to D^*$

There are several algorithms which are there; we look at some basic search algorithms. Now, the search algorithm basically means what is searching for? It is searching for a path, we know that it can be a best path for example, it can be the shortest path, or it can have other constraints. For example, time can be a constrained, dynamic cost can be a constraint, energy can be a constraint.

So, it depends on work do you mean by an optimal path, what is it that you are looking for, as I come along, I will be explaining what this means. So, the first algorithm the easiest is or the earliest is in somewhere in 1959 by Dijkstras. So, Dijkstras was a computer scientist. And he was basically looking at the shortest path when there is a network of paths. Now, modify algorithm for that or a more advanced one is A * search and then from A * search we have what is the D * search or dynamic A *.

So, this is my dynamic A * which is called D *. So, we will look at very basic search algorithms today. And then we will move on to look at trajectory planning.

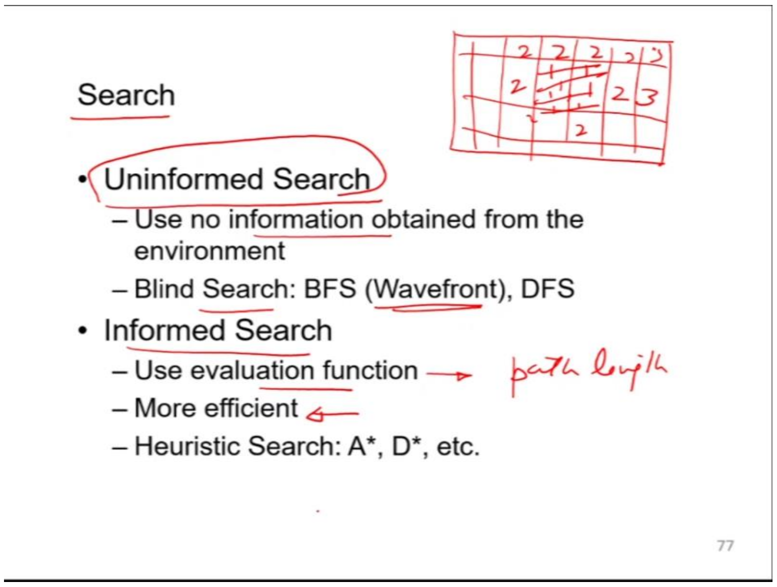
(Refer Slide Time: 06:05)



Now, what we are basically looking at is a network of paths or graphs, in our case, it is a road map, basically it is a network of roads. Now, there can be different kinds of types of graphs, for example, you can have a tree type, in which case you have a root. So, there is my root, from the root you are having branches, so the branches are going out like this, each of this is a node and these of this an edge and that is a node.

May can also have a graph which is like this, for example, a collection of edges and nodes, nodes which are not tree type. But a graph where you are having nodes like this and these are edges, so these are 2 types of networks that you can have for graphs.

(Refer Slide Time: 06:56)



Search

- Uninformed Search
  - Use no information obtained from the environment
  - Blind Search: BFS (Wavefront), DFS
- Informed Search
  - Use evaluation function → path length
  - More efficient
  - Heuristic Search: A*, D*, etc.

77

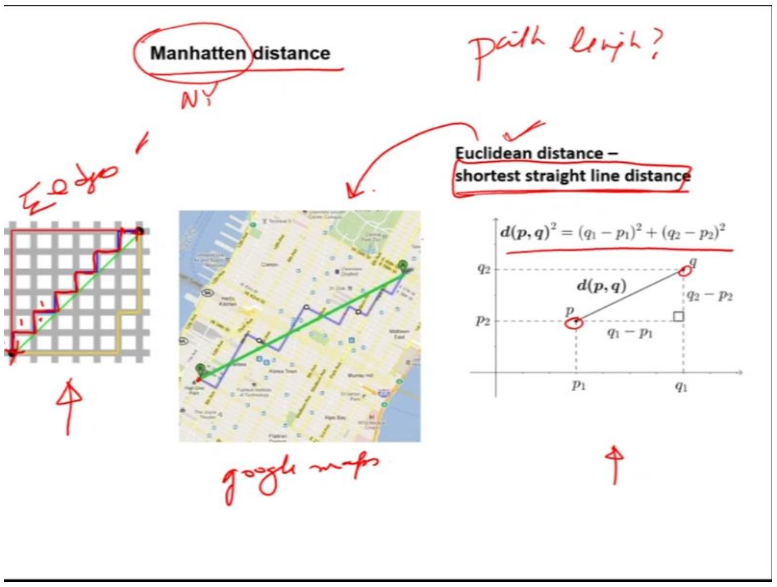
Now, we can divide this search in 2 types. One is an uninformed search, this uninformed search users no information obtained from the environment, it can exist some kind of a blind search. So for example, when we talked about wavefront, planning wavefront planners, they were basically blind searches. So, basically means something like this, this is a wavefront search. So, we have an obstacle. We have a number this is 11111.

The next is 222 so this is basically a grid search. So, we make grids like that. And, and what we do in this wavefront search is that if there is an obstacle here, we name that as 1 of that grid, the next grid will become 222, then the next grid will become 3 in that way. So basically, in this case the search is propagating. It is like a wavefront that is why the name comes from. And it is a blind search.

It does not know what other obstacles are on the way, what is the goal where is the start point, it is simply searches like a wave which is expanding from one point and going in all directions. So that is uninformed search. So, the other kind of search is what we call informed search, where we can basically evaluate a function, the function could be evaluating the path length.

Suppose I want the shortest length path, then I am going to evaluate a function which will take the sum of all the edges and say that, if you go this way, it is fastest. Obviously, these are more faster they are more efficient. So, informed search is going to be faster than uninformed search.

(Refer Slide Time: 08:37)



Manhatten distance
NY
path length?
Edge
Euclidean distance – shortest straight line distance
$d(p,q)^2 = (q_1 - p_1)^2 + (q_2 - p_2)^2$
$d(p,q)$
$q_2 - p_2$
$q_1 - p_1$
google maps

Now, when we are talking about a distance, let us take this example which we are all familiar with, for example, we are trying to go from a point this is Google Map, for example which I am sure all of you have seen. Now, I want to go from this point to this point, so this my start point and this my goal point on a Google map. Now, it may appear that the path length will be shortest if it is a straight line path length.

From the start point to the goal point is a straight line like this, But on this Google map, which is very clear that you can go only on where the path exists or on the roads like you can fly across like that. So, the road is like this, like this, like this, like this, like this, like this, like this, wherever there is a road you have to follow the road that you cant go whether is no road. So, there are 2 distances we will be talking about.

One is called the Manhattan distance, this name obviously comes from Manhattan in New York, that if you want to go from one place to another place, you have to follow the streets and then get from one place to another place and because of which your path will not be a straight line path. It will be zigzag like this depending on wherever there is a road. So similarly, when we are looking at a grid like it is shown here, I want to go from here to here.

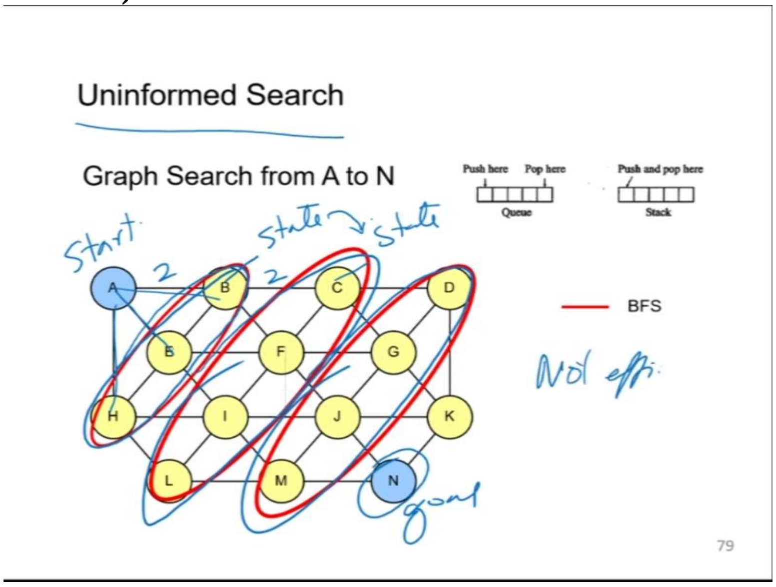
I cannot go straight because I can go on like this. So, grid distance is basically called the Manhattan distance. So, in this case for example, this could be 1111. So, basically sum of all these paths all these edges is what we call sigma for the edges is what we call the Manhattan distance. The other of course, you are familiar with is called the Euclidean distance. So, if I have a point p here and the point q there which has coordinates.

So, this has point (p_1,p_2)and (q_1,q_2), then the distance the shortest distance of the straight line distance is given by this. So, this can be x or y. So now, in this particular case, we see that this is the shortest straight line distance this is called the Euclidean distance. Now, obviously, you cannot apply that here why? Because you cannot actually go straight.

You have to follow wherever there is a road; you have to follow the road. So, in path planning also when we are looking at the path, when you are looking at a path here, for example, all these examples I showed, you want to go from here to here, from the start point to the goal point, and there is a straight line, the straight line is probably like this but you cannot go on the straight line, so you have to follow a path.

Which will probably be from here to here, then that so this is very similar to the Manhattan distance wherever there is a road, you can go only there. So, these are 2 concepts that we use, one is called the Manhattan distance to take you from one point to another point. So, when we are saying length, it could basically be the Manhattan path length, or it could be the straight line distance which is a straight line distance. This is called the Euclidean distance between 2 points, so when we are saying path length, I have to specify which one I mean, the Manhattan distance or the straight line paths, let us proceed.

(Refer Slide Time: 11:53)



Uninformed Search

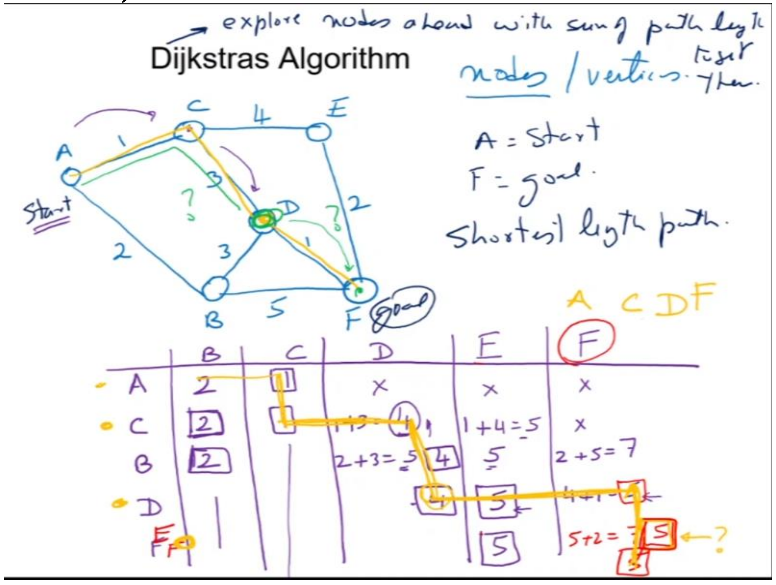
Graph Search from A to N

So, the first is uninformed search, the uninformed search, for example you are at the start point here and you want to go to the goal point end. And you do not have any more information means you do not know the path length, you do not know, what is the cost at each state. In that case, you can simply start searching. Once you search these nodes, then you search these nodes, then you search these nodes.

And then finally from there you come and see where is the goal, so this is basically a way of searching where you have no prior information either regarding path length or about the transition of states. Now something to node about is that when we are talking the way we are writing this graph is this is one state, this is another state. So, it is going from one state to another state which we call this state transition.

So, there could be a state transition cost. For example, the distance to the goal could be a state transition cost. And there could be a path length for example; the path length could be to 2 units. So, this is an uninformed search where you are searching without any prior information, then you want to go from the start to the goal, this is of course not very efficient, not efficient and for very large searches, it will become very slow.

(Refer Slide Time: 13:09)



	B	C	D	E	F
A	2	$\boxed{1}$	x	x	x
C	$\boxed{2}$	$\boxed{}$	1+3= 4	1+4=5	x
B	$\boxed{2}$		2+3=5 $\boxed{4}$	5	2+5=7
D			$\boxed{4}$	$\boxed{5}$	4+1=$\boxed{5}$
F				$\boxed{5}$	5+2=7 $\boxed{5}$ ← ?

Now, the first of the informed search, what we talked about is the informed search is the Dijkstras algorithm. So, Dijkstras algorithm goes something like this. Suppose we have a graph like this this is A this is C in this my point these are the various I will just draw the various states so, these are various states you can consider them to be a network of paths and we want to go from one state to here and this is here.

So, let us say that this is my state A, this is my state C, this is E, we are just calling them they can be the vertices or nodes in a graph, this is this is F. So, now, what we see here is that these are various nodes of the graph. Similar to the visibility graph, these can be vertices nodes vertices, as in the case of visibility graph, now what we are seeing is the path length is there, this is a path length which will take.

So, if you want to go from A to C that the path length is one from C to E it is 4, from C to D is 3, from B to D it is 3, here it is 2, then here it is 5, here it is 1 and here it is 2. Now A is my start point, let us say A is a start point. And my goal point is given here, this is my goal, so A is start and F is goal. So, let us write it down a start, let us write it here A is start. And F is goal now I need to find the shortest length but find the shortest length path.

Now, this is a very simple graph, there are just 3 or 4 branches which are going on. But it can be very complex with a lot of nodes and vertices and edges, so you can have a very large number of paths their searching is not very easy. Here, you can made as which one is the fastest one. Now Dijkstras algorithm what it does, it explores nodes ahead with some of path length.

So, Dijkstras algorithm it explores nodes ahead with sum of all paths with some path lengths to get there. To get there, so we will start with A and then I will go to C. And so the path length from A to C is 1. Now if I go to E, then the path length to E is it has come from A to C, C to E, so it to be 5, I hope you understand that. Now, what we do is we start off by making this table and as I explained before, we have to write a programme which will do this.

So basically, we are making a table. So, I start off with A, this is my A, then there is B, C, D, E and F. So now, what we do is, I start off start point is A, so, I am starting from A, and from A to B, so there is A and there is B, what is the path length between A and B it is 2. So, from A I go to B, it is 2 from A if I go to C it is 1 is the path lengths already given. Now, you cannot go to D and you cannot go to E and you cannot go to F from it straight.

Now out of this, what we see is which one is the smallest shortest path length; the shortest path length is C. So, what I do is I box it and I bring it down. And this is C, so let me write it here. This is 1. So, I put this in a box and I bring it down. And this is my C. So, the next node that I should try and explore is C because it has the lowest path length. So, I bring C in here, now from C I want to go to B, I am from C there is no direct path to B, but you can go by A.

So, basically what I am saying is this please note this that now this 2 will come down here as 2 there is no direct path, but there is a path by B. So, these 2 comes down here is 2 now from C to B, C to C is 1 and C to D now, if you want to go from C to D, so C is here and D is here what is the total cost. It is the cost of the previous node how much from A to C then C to D so, it is 1 + 3.

So, from A to C it is 1 and then from C to D it is 1 + 3 = 4. Now, what about C to E and going from C to E, C to E, A to C is 1 and then from C to E is 4, so that is 5, that is the path length now from C you can go to F so you can go there, now in this row, which one is the smallest path length. The smallest path length is 2. So, this smallest path where here. So, the smallest path length is 2, this is 4, this is 5, this is 1 and this is 2.

So, what I do is I box it and then I bring it down that means the next node that I am going to explore is going to be B. So, this is my next node that it comes down here. So, I box it and I bring it down, now I can cross over C, so I do not need to go to C anymore. So, C is done already. Now I am in B so you can visualise you are at B, now from B you are going to D so if you are going from B to D, so from A to B was 2 and B to D is 3 which is equal to 5, and what about E now, B to E, you cannot go directly from B to E.

But you can go where the previous column, previous column is saying that the path to E is 5, so I bring 5 down, please note this. Now, what about from B to F, from B to F you can go from B to F, you have already travelled A to B is 2, and B to F is 5, so, that is 7. Now, in this row which one is the lowest. So, in this row we find that so, point I missed out just note here that this is the previous one was 4.

So, that that means, there is a path B to D which is less than 5, so, I bring 4 down here, so, I am going to bring 4 down please note this carefully, So, in this particular case, then although this is 5, B to D is shown 5 but on top there is a 4 that means there is a path which has a lesser cost and so, I bring the 4 down and box it so, I bring the 4 box down here. So, in this row 4 is the smallest.

So, I make a box on 4 and I bringing down now so which was this, this was D. So, this is D the row is D. Now from D I do not need to go to B into that I do not need to go backwards to B so I can carry it out. Now what about D to D? No D to D there is a 4 there is already 4 here.

Now, what about D to E? D to E is 5 because there is a so D to E, there is 5, you can go by a 5 so what I do is I bring 5 down.

And what about D to F, so D to F is we got 4 to come to D I have gone 4 A to C C to D that is 4 plus D to F is 1 so, this is 5. So, in this row, you will see that there are 2 low numbers one is 5 the other is this one is also 5. So, you can choose either of the 5s. So, let me choose this one, this 5 I box it may bring it down now and which columns are corresponding to it corresponds to E.

So, from E, so, this one is E. So, let me just drop this out this one is E, so, corresponds to E so, this one is E. So, from E now, I need to go to F so, you can see that from E to F to come to E it is 5, 5 + 2 = 7, but 5 was already small. So, I bring this a box it and I bring it down. So, compared to 7, 5 is less that means there is a path which goes to F which is lesser cost this the meaning, now I need to find which is the path.

So, how do I do that, so, what I do is now that means 5 is the minimum path length. So, what I do here is I follow the 5s up so from here, so now what we do is we see that there is a 5 here, now what I do is I take the the what is the lowest here the lowest is F now, that is my F, so I put F here, the last one is going to be F, E and F and in F to F, we just bring this down here that is F now. So, it is A C B D F that is the way we have come.

Now what we can do is we look at this column 5 and go up. So, I can go up like this. So, from 5 I go up like this, these are all 5s, now the next one is 7, so I go up 555, then I come left to 554, that means from F, I am coming to left, left, left left, I am coming to 4, this is the lower one then I go up here again. So, I go up here it comes to 4 and then I can go left it comes to 1 and then I go up, it comes to 1 again.

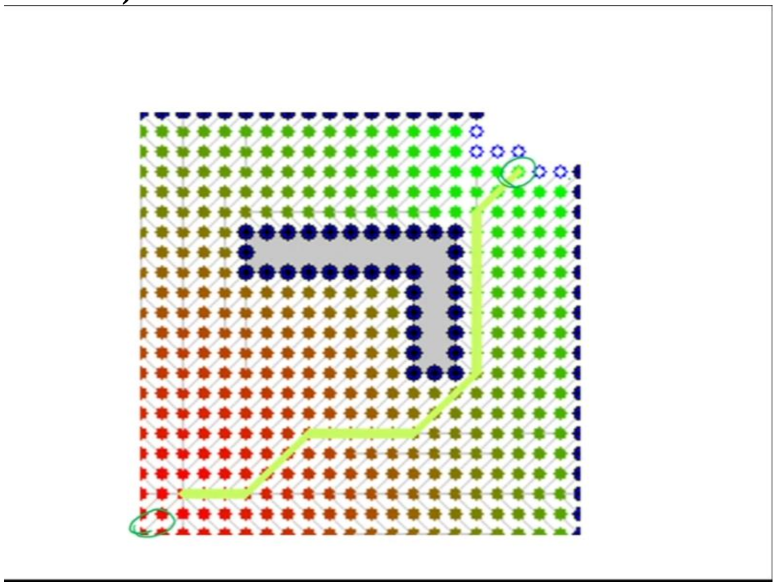
So, basically I am following the lesser number path. So, from 5 I go up so these are all 555 then I go left. So, from 5 again 5, it comes to 4, now 4 I go up, it comes to till 4 and then I go left it comes to 1 and then when I go on to it comes to 5, now what will be the path. So, wherever there is a horizontal line that is going to be so it is going to be F is the first one so F then the next is the horizontal line which is D.

So, let me write down F is the last one if just before to that is D, which is D here and from D basically I take the next one which is the horizontal line which is C this one so, I take C and then to A so, it is A so ACDF, so ACD and F is the path that will give me the shortest path AC. So, ACD and F that is the shortest path and what is the length of the path? The length of the path is 5. So, basically what we are seeing here is that using this algorithm basically we can find what is the shortest path to take you from one node the start node to the goal node.

Now, this if you can imagine is that it is exploring all the nodes. And if there are many nodes and if there are many paths, then this is going to becomes low. Now, I suppose, it could be a little bit more informed search, for example, if we knew which is the direction of the goal, then I could search in that direction instead of going in all directions, because I can make it faster, is not it?

So, first of all what I could say is, if I made node D suppose I am at node D and I know I have to go to goal F, then I can have first of all this distance and I can have a heuristic which will tell me that maybe the heuristic can be the distance and the distance from D to F if I have that idea, then I can add this and that and I know which side is the goal and my search is going to become faster.

(Refer Slide Time: 26:43)

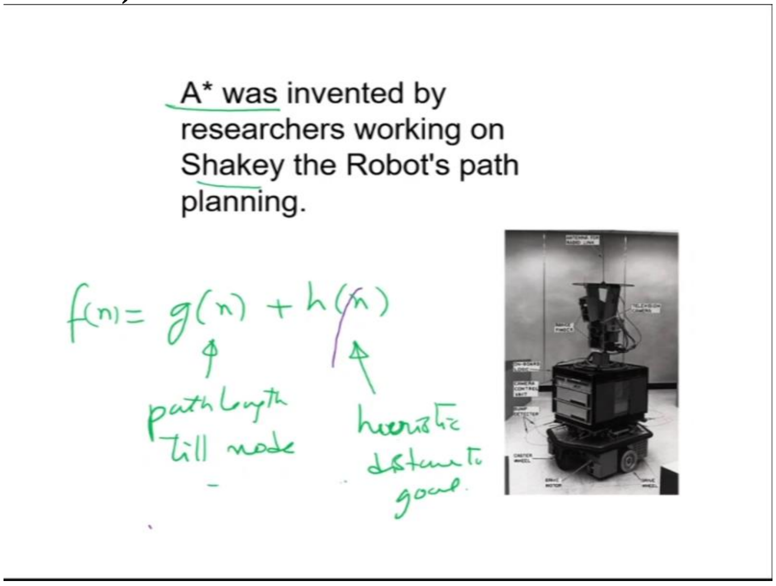


And that is exactly what is done in the A *, this is showing the Dijkstras algorithm. So, every node is searching the adjoining nodes and is going towards the goal that is the way we have done, but there are a large number of nodes here. So, what is happening is that it is going the search is taking quite some time because it is searching on all the nodes and then finally, it will find what is the shortest path?

So, you can see that it is taking some time to go from a start which is there, the goal which is there. So, it is exploring all the nodes adjoining nodes and then it is finding it is going to the goal and then is finding a path. So, this is Dijkstras algorithm and Dijkstras algorithm is a little slow because it does not really have any information or idea where the goal is. So, it searches all adjoining nodes and then goes towards the goal and then tries to find out where the goal is, this is exactly how we wrote our table.

Now, if you can think of a faster way to search it, suppose I gave it some information that was the distance to the goal, then it could try and go in particular distance which will be closest to the goal and that is basically my A * search, so this is Dijkstras.

(Refer Slide Time: 28:05)



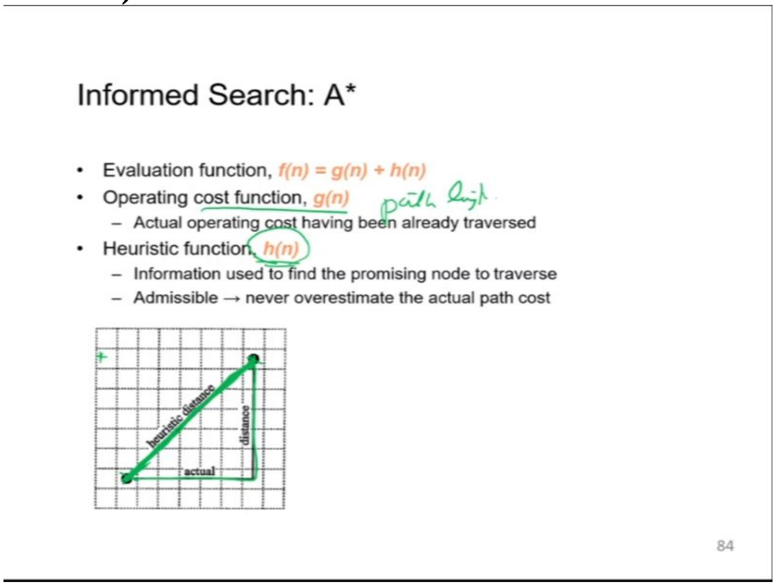
A* was invented by researchers working on Shakey the Robot's path planning.

$$f(n) = \underbrace{g(n)}_{\text{path length till node}} + \underbrace{h(n)}_{\text{heuristic distance to goal}}$$

Next is A*, so A* was invented by researchers working on Shakey the robotics path planning, earliest robotics path planning algorithms. So, Shakey was the mobile robot which was made in Stanford and the algorithm that was used was A*. Now, what A* does is it evaluates a function fn which is a sum of to g(n) + h(n). Now, this can be the path length till that particular node. So, for every next node suppose you are at node A and your next node is node B. Then this is the path length till this node till node and this is a heuristic which can be the distance to goal.

So, basically in the first case, in Dijkstras algorithm, we were just looking at the path lengths, we did not have any heuristic which was telling you how far is node from the goal whereas here I have a heuristic and the heuristic carry the distance to the goal.

(Refer Slide Time: 29:19)



Informed Search: A*

- Evaluation function, $f(n) = g(n) + h(n)$
- Operating cost function, $g(n)$ path length
  - Actual operating cost having been already traversed
- Heuristic function, $h(n)$
  - Information used to find the promising node to traverse
  - Admissible → never overestimate the actual path cost

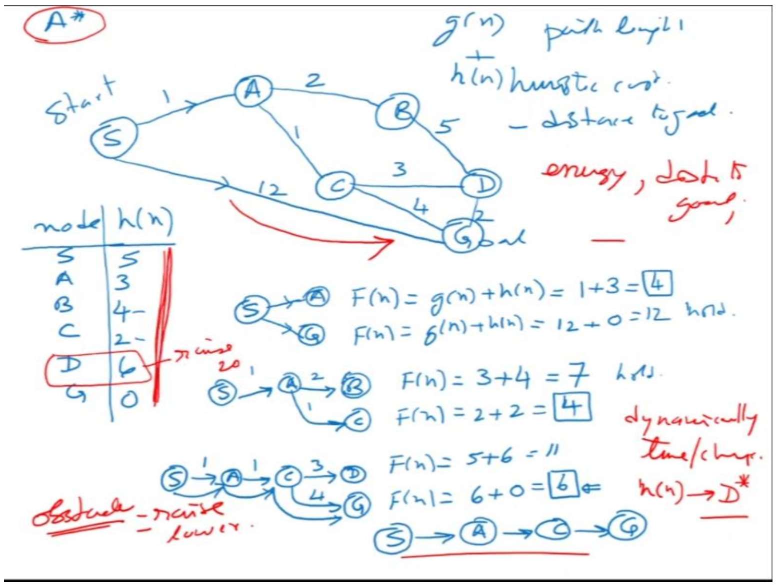
heuristic distance
distance
actual

84

The distance to the goal could be the heuristic distance, so this is my heuristic distance. So, this is the actual distance which is the Manhattan distance and the heuristic distance is this distance. So, from this node to this node, this is my distance. So, if I talk about this node, this node to this node the distance could be more. So, basically the informed search A* we evaluate the operational costs which is the path lengths which is here path length.

And the heuristic function which is h(n) my information used to find the promising node to traverse and because h(n) is there it is in a way, it is saying that which node has the least distance to the goal, I should go in that direction. So, let us look at how this is done. Let us take an example a very simple example again, for A* algorithm, so this is my A*.

(Refer Slide Time: 30:22)



So, let us take a graph like this where this is my start node, S, this is node A, node B, then a node C is here, node C, node D and then we have node G. Let us say my graph is like this, it goes this way A to B, B to D, D to G and C can go to D as well as to G and here, A goes to C an S can go to G, this is an interesting graph. So, S is the start node and G is the goal node this is goal, G is goal and this is start, now the path lengths are 1 2 5 get 1 3 4 2 and 12 now, this is one set these are the path lengths.

So, here we have the g(n) which has to do with the path lengths as in the previous case, but we also have a heuristic function sorry we have a heuristic cost, now, the heuristic cost is h(n) now, this could be the heuristic cost could be the distance to go from that particular node. So, let us say I put my nodes for every node say for node S, A B C D C D and G, each of these nodes there is a cost associated now, this cost as I said it could be the distance let us assume it is the distance 5 3 4 2 6 0.

So, in this case it is made up of g(n) + h(n) now, let us start off from the start point S, S is my start now, from S you can go to A and you can go to G. So, you can follow this path or you can follow that path. Now, if you simply look at this graph, it would appear that why not take the straight road this one S to G you have a cost of 12, path length is 12, why not take that one. So, let us see whether that is correct or not. So, from S I go to A and at A basically what we are doing is we are evaluating the function F(n) = g(n) + h(n).

And so, from S to A what is the length of the path that is 1 plus what is the cost associated with A the cost associated with state A is 3 so, this is 4. Now, S to G what is F(n) it is g(n) +

h(n) which is equal to what is the path length it is 12. So, path length is 12 and what is the cost the h(n) heuristic that is 0. So, this is 12 now, out of these 2 which one is smaller for smaller so, I box it as in the previous case I make a box and I keep this on hold. That means the next node that I have to search is A not G why because the cost of A is less.

Now from then my second step is S to A and S to A from A you can go to B or you can go to C, we can see that from S is going to A now for A you can go to B or you can go to C, now if we go to D, then this function F(n) is equal to here so, it is D, so A to B = 3 + 4 which is equal to 7. Now, how do I get 3, so, to come to D, so, from S to A is one, then from A to B it is 2 write that is the path length.

So, F(n) = 3 to come to from S to A to B it is 3 + 4, 4 is the transition state transition heuristic cost which is here, which is 4, so, total I get is 7. Now, what about from S to A to C it should A to C, F(n) will be equal to you get C you have come from S to A which is 1 and the A to C it is 1 so there is 2 plus and the heuristic cost for C is 2, so, it is 4. Now, out of 7 and 4, basically this one is smaller, so I box it and I keep this on hold.

Now, which basically means that I am going from S to A to C now from C you can go to D and so I have gone from S to A from A I have gone to C and from C I can go to D and I can go to D or I can go to G so, when I am going to D the function F(n) = D where is D? D is here. So, from C to D I have gone from S to A which is 1 then A to C which is another one and then C to D which is 3 so, it is 5 plus the heuristic cost is 6 so, it is equal to 11.

Now, what about C to G? C to G is a C to G has a cost of 4 or the path length so, F(n) is equal to how much is this equal to C to G = 6 + 0 so it is 6. Now out of these 2 which is smaller 6 is smaller. So, I box it. Now if you see we have reached G and which was the smallest which was the the shortest path it was S to A to C C to G, I hope you can visualise that. So, this is giving is 6 and that is the shortest path lengths.

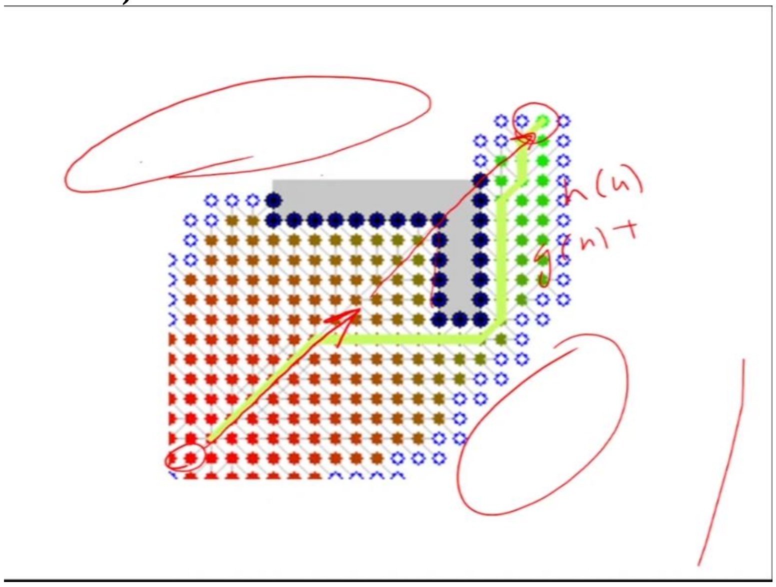
So, what you can see here is that this one is giving us the giving us the shortest path length, so S to A, A to C and C to G that is the shortest path length. So, although by looking at this graph, it was interesting I said because weather looking at this graph, it appears that this path length would be the shortest because it is a straight path but it is not because this one is smaller, it is shorter.

So, this is basically A*. Now in A* algorithm what we saw is that we have a gn which is the path length and we have a h(n) which is the heuristic cost the heuristic cost can be anything it can be it energy, it can be distance to goal. Now it can be anything else. So, in this particular case, this is faster than your next Dijkstras algorithm. Now, suppose you can imagine that we have already assigned which and the state transition like this.

But when the robot is traversing suppose there is an obstacle now then what it can do is suppose there is an obstacle. So, what the algorithm can do it can raise or lower some of these transition costs. So, for example, if it is going towards D and here it finds that there is an obstacle. So, what it can do it can raise this ones cost and make it maybe 20. So, the robot does not go in that direction again or if it finds that history path in some direction it can lower the cost.

So, you can dynamically change. So, dynamically tune or change this h(n), the values of the heuristic hn and this is what we call the dynamic A* or it is called the D*, so we looked at blind search, then we looked at a Dijkstras algorithm, we looked at A*. Now D*, or dynamic A* is exactly same as A*, the only difference being that you can change this you can raise or lower those way as you go along, so the robot does not get stuck somewhere or if there is an obstacle, the robot understands that there is an obstacle there now.

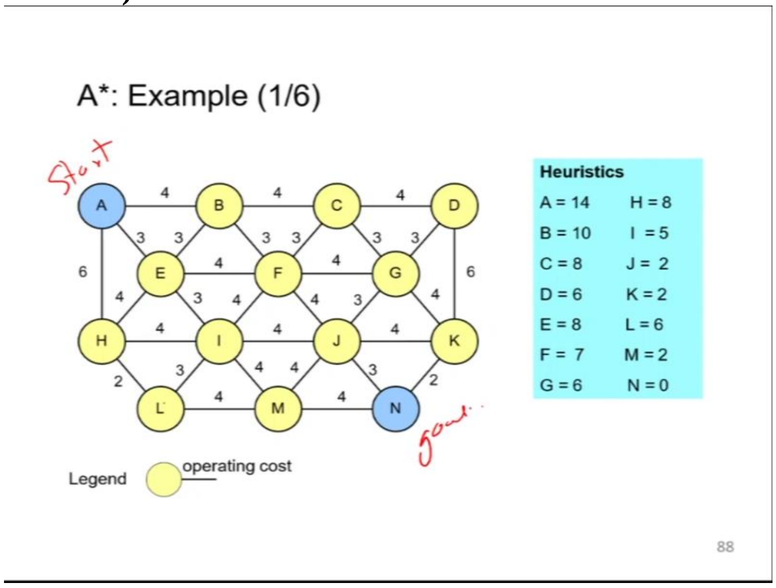
(Refer Slide Time: 39:25)



So here, you can look at an example that where there is a start point and there is a goal, now the robot is starting off by going straight towards the goal, because that is the direction which the distance is least and as it is hitting that obstacle, what it is doing is finding surrounding areas, and then it is going again towards the goal and it finds the path. That is the path. Now you can see it is interesting that it did not search here, it did not search here like unlike the Dijkstras algorithm where it was searching everywhere.

So, here you can see, let me it starts off simply by going in the direction of the goal, why because it knows the heuristics that h(n), h(n) is in that direction and g(n) + h(n), h(n) is a straight line direction towards the goal. So, that is why it is going towards the goal first, then it hits the obstacle and then finds the bypass and then goes off this way. So, it did not search anywhere, that did not search anywhere. So, this is much faster.

(Refer Slide Time: 40:19)



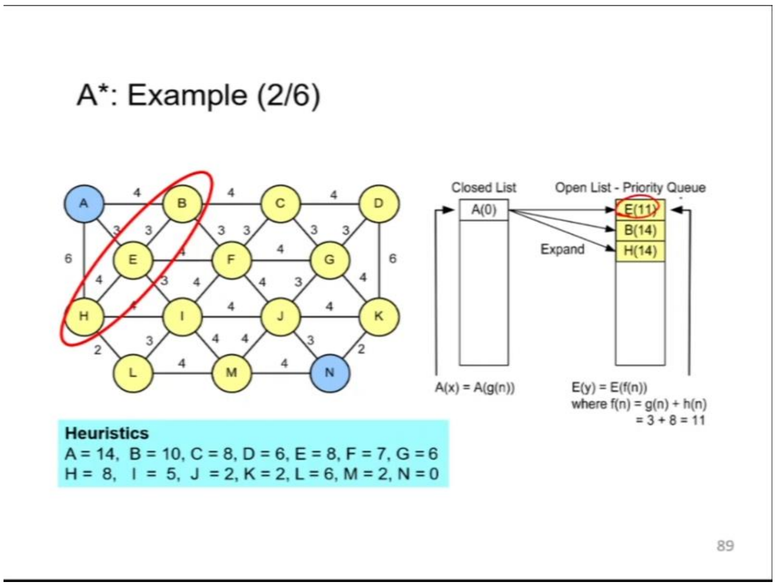
A*: Example (1/6)

Heuristics	
A = 14	H = 8
B = 10	I = 5
C = 8	J = 2
D = 6	K = 2
E = 8	L = 6
F = 7	M = 2
G = 6	N = 0

This is another example of a Dijkstras algorithm. So, these are various nodes. Now, in the previous case, the example I just solved, there are very few nodes, there are 12345, only 6. But you can imagine that in a real world situation where there are many obstacles, there are many, the robot is going there many obstacles to this very large workspace, the network of paths can be very, very large or if you are looking at a grid search, the number of grids could be very, very large. So, in such a case, what we can search this way.

So, this is A*, and this is exactly how it operates. So, we start off from A which is my start. And this is my goal point.

(Refer Slide Time: 41:04)



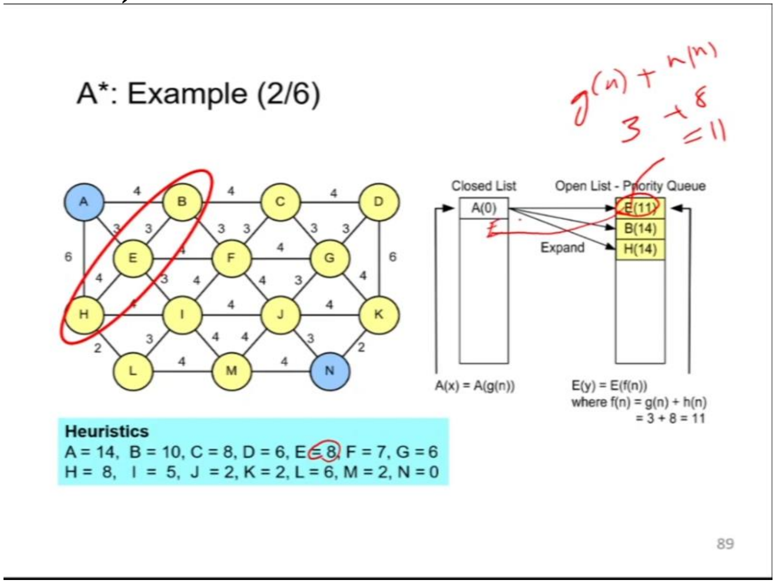
A*: Example (2/6)

Heuristics
A = 14, B = 10, C = 8, D = 6, E = 8, F = 7, G = 6
H = 8, I = 5, J = 2, K = 2, L = 6, M = 2, N = 0

89

So, I start from A and this is the list from A (0), I am starting from A (0), I am searching this nodes, E B and H, so from here, I am going here, here and here. And I am writing it this way, so I am maintaining this list. And then from the list, I am expanding exactly the same way that I did before. Now out of this, you can see that which one has the least cost. E, B and H, it is E so the next one that I am going to do is is going to be E.

(Refer Slide Time: 41:34)



A*: Example (2/6)

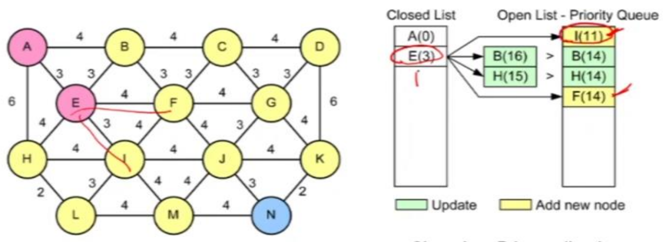
Heuristics
A = 14, B = 10, C = 8, D = 6, E = 8, F = 7, G = 6
H = 8, I = 5, J = 2, K = 2, L = 6, M = 2, N = 0

89

So, the next one I want to start so E comes there. Now again, the way I am getting the value of E is essentially 3. So, you can see E is 3 the g(n) + h(n) where is E, E is 8. So, h(n) is 8 that is how I got 11 which is equal to 11. So, exactly the same way that I have done it in the previous example. So, we get E will come here now. Yes, E has come here. Now E is connected to what?

(Refer Slide Time: 42:06)

A*: Example (3/6)



Heuristics
A = 14, B = 10, C = 8, D = 6, E = 8, F = 7, G = 6
H = 8, I = 5, J = 2, K = 2, L = 6, M = 2, N = 0

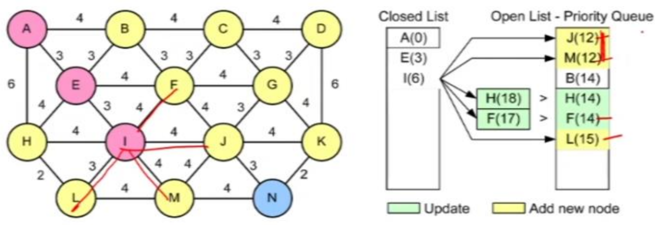
Since A → B is smaller than A → E → B, the f-cost value of B in an open list needs not be updated

90

E is connected to F I. E connected to F and I and B and H are there from 4, so I am adding I and H to this K, now out of this the cost is least for I. So, next I will come there.

(Refer Slide Time: 42:24)

A*: Example (4/6)

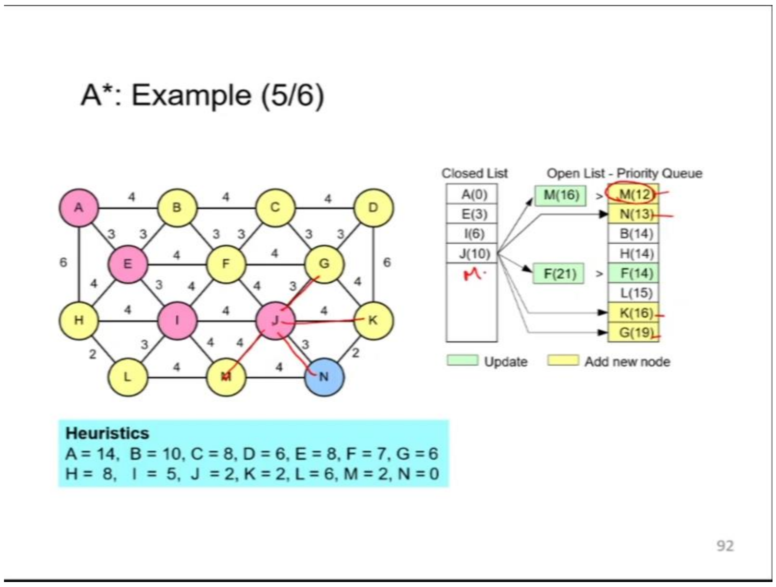


Heuristics
A = 14, B = 10, C = 8, D = 6, E = 8, F = 7, G = 6
H = 8, I = 5, J = 2, K = 2, L = 6, M = 2, N = 0

91

So, this is how I am expanding. So, I comes J here now I is connected to J and M, L and F. So, L and M and J and M and the search is connected to L and F. Now here which is the least these 2 a least so you can take any one.

(Refer Slide Time: 42:42)



A*: Example (5/6)

Closed List

A(0)
E(3)
I(6)
J(10)
M·

Open List - Priority Queue

M(16)	>	M(12)
		N(13)
		B(14)
		H(14)
F(21)	>	F(14)
		L(15)
		K(16)
		G(19)

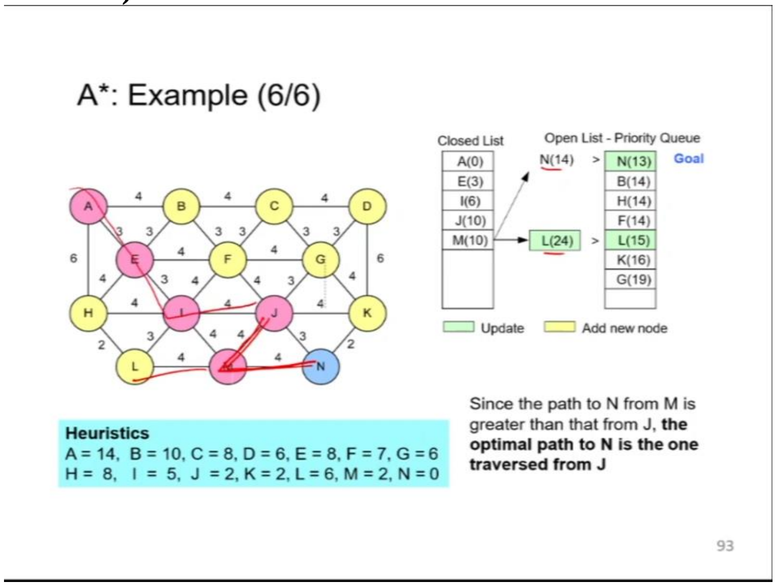
🟩 Update 🟨 Add new node

Heuristics
A = 14, B = 10, C = 8, D = 6, E = 8, F = 7, G = 6
H = 8, I = 5, J = 2, K = 2, L = 6, M = 2, N = 0

92

So, I took J now where is J connected to K G M and N, so J is connected to M N so I am taking M N G and K G and H sorry, sorry G and K. These are the associated costs again you find which is the least cost out of here, which is the least cost, so it is M. So, M will come there now.

(Refer Slide Time: 43:06)



A*: Example (6/6)

Closed List

A(0)
E(3)
I(6)
J(10)
M(10)

Open List - Priority Queue

N(14)	>	N(13)	Goal
		B(14)	
		H(14)	
		F(14)	
L(24)	>	L(15)	
		K(16)	
		G(19)	

🟩 Update 🟨 Add new node

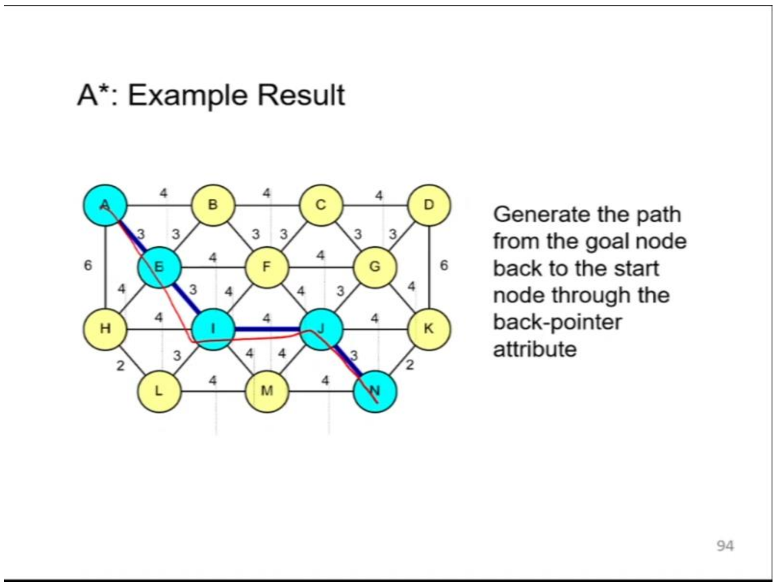
Since the path to N from M is greater than that from J, **the optimal path to N is the one traversed from J**

Heuristics
A = 14, B = 10, C = 8, D = 6, E = 8, F = 7, G = 6
H = 8, I = 5, J = 2, K = 2, L = 6, M = 2, N = 0

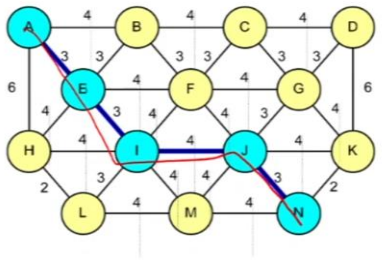
93

So, M is come there, now M is connected to what M is coming to L this side N in this side. So, M is connected to N M is connected to L. So, what is the cost, so we find the cost again?

(Refer Slide Time: 43:18)



A*: Example Result

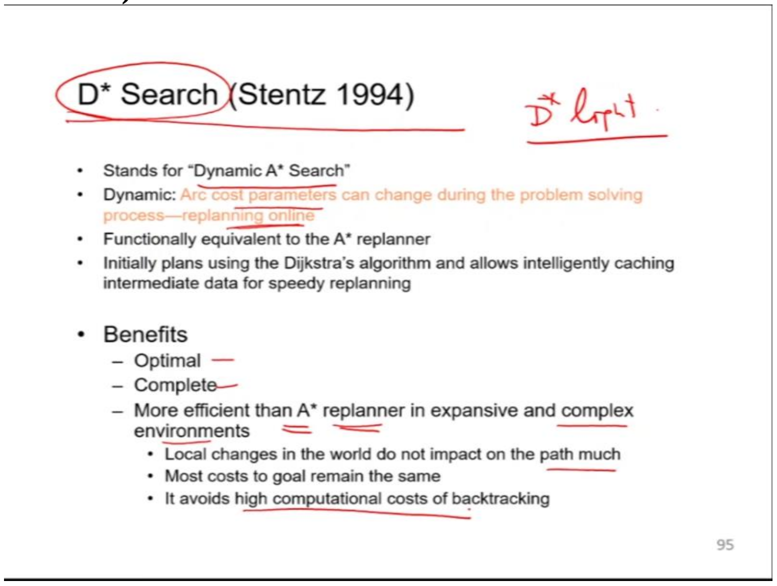


Generate the path from the goal node back to the start node through the back-pointer attribute

94

Now, what we do find is that although it when this side, this does not this connection does not give you the least cost. So, I connected this way and what we find that is because the optimal path is not this one, not this one, but this one, this is my optimal path. So, the optimal path is this one know. So, my optimal path is this one. Now, it is exactly the same way that I had done in the previous example. But this is using you can say pointers in which we are keeping nodes or we are entering nodes and throwing out nodes depending on which one is the least cost.

(Refer Slide Time: 44:11)



D* Search (Stentz 1994)

D* Lite.

- Stands for "Dynamic A* Search"
- Dynamic: Arc cost parameters can change during the problem solving process—replanning online
- Functionally equivalent to the A* replanner
- Initially plans using the Dijkstra's algorithm and allows intelligently caching intermediate data for speedy replanning

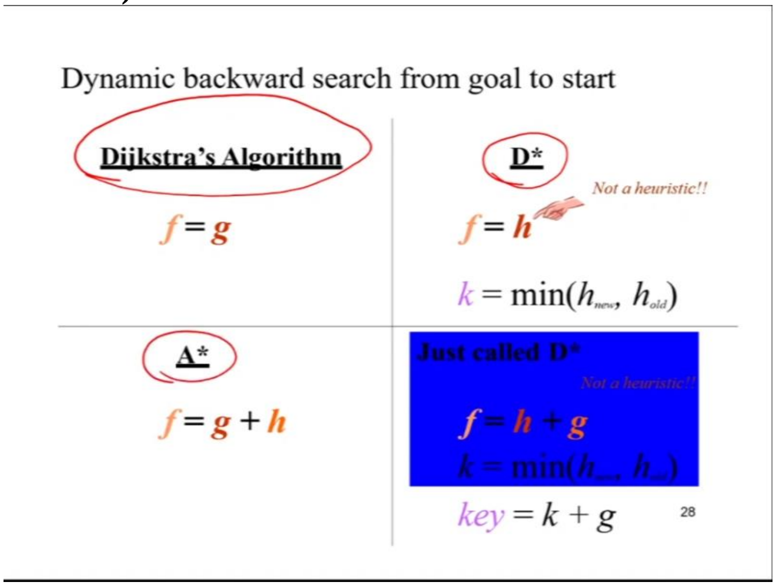
- Benefits
  - Optimal
  - Complete
  - More efficient than A* replanner in expansive and complex environments
    - Local changes in the world do not impact on the path much
    - Most costs to goal remain the same
    - It avoids high computational costs of backtracking

95

Now, the next thing that we will need to look at is D* and then we will look at, now in D* as I mentioned the D* basically stands for Dynamic A*. Now in dynamic A* the arc cost parameters can change during the problem solving process. So, replanning can be done online say for example, there is some place where there is an obstacle. So, you can replan that area very quickly or dynamic in case where the system is moving there is a dynamic obstacle now.

In case of dynamic obstacle you have to dynamically change the light that is basically where we use the D * there is also something called D * light. So, the D * light is also there. Now, the benefits of the D * is that basically is complete optimal complete it is more efficient than A* because re-planning is expensive and complex environments where the environment is very complex. Now the local changes in the world do not impact on the path much. So, most costs to go remain the same and it avoids high computational cost of backtracking.

(Refer Slide Time: 45:19)



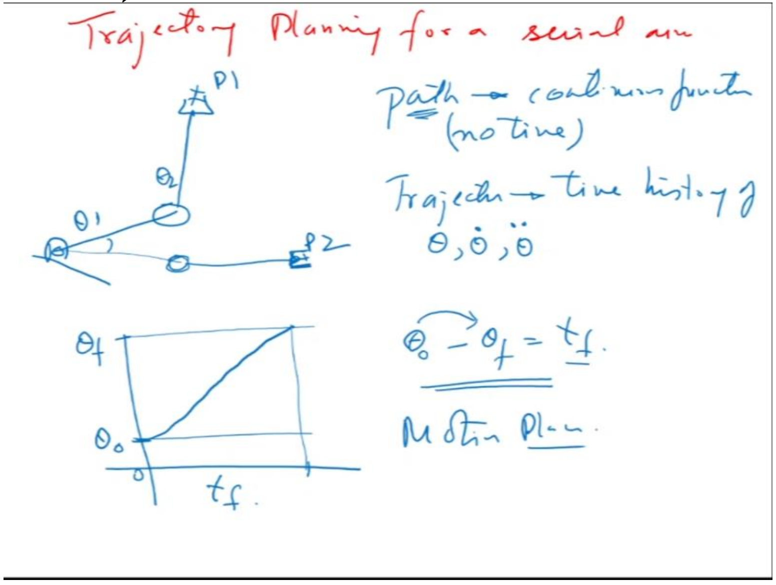
Dynamic backward search from goal to start

Dijkstra's Algorithm	**D***
$f = g$	Not a heuristic!! $f = h$ $k = \min(h_{new}, h_{old})$
A*	Just called D* Not a heuristic!! $f = h + g$ $k = \min(h_{new}, h_{old})$ $key = k + g$

28

So, basically, this is the basic differences between Dijkstras algorithm and D*. And this is A* so, these are the comparisons between Dijkstras algorithm A* and D*.

(Refer Slide Time: 45:34)


Trajectory Planning for a serial arm
P1
θ2
θ1
P2
Path → continuous function (no time)
Trajectory → time history of
$\theta, \dot{\theta}, \ddot{\theta}$
θf
θ0
0
tf.
$\theta_0 \rightarrow \theta_f = t_f$.
Motion Plan.

Now, what we look at next is trajectory planning, trajectory planning for a serial are, now, what is what is this what this actually means is that when I have a to link manipulator for example like this, I want to go from this point P1 to the point P2 I want to go like this, I want to go here to this my point P2, the actual planning will not be done in the Cartesian space, it is done in the joint space.

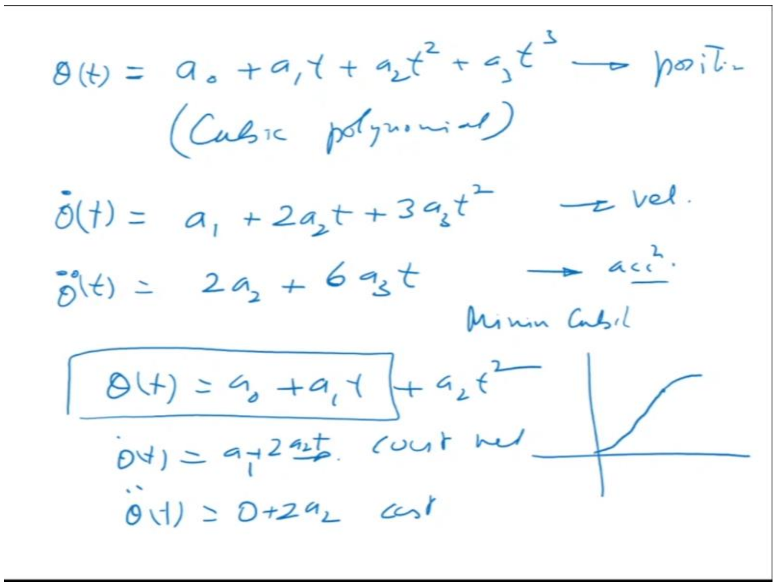
So, I want to go from here to here I may be able to go via this point also I can call it P3, but the controller basically will control the joints, θ_1 and θ_2 that is what is your controlling. So, the controller will actually move the joint and then it is going to take the end effector from point P1 to point P2. So, a path is basically a continuous function, so, path is a continuous function and there is no time here.

Whenever we talked about all the earlier examples I talked about there was no time, time was not there at all, but to go in the real world from one point to another point, you have to control the joints, the joint angles. Now, this is basically what we call a trajectory. So, trajectory is is the history time history, time history of θ, $\dot{\theta}$ and $\ddot{\theta}$ .So, here there are 2 θ, θ_1 and θ_2.

So, I need to control θ_1 and I need to control θ_2 separately, by giving fitting trajectories to θ_1 and also fitting trajectories to θ_2. So, in this case, what we do is let me draw an example here. So, this is my let us say θ_0 it has to go from θ_0 to θ_f in time t_f this time t_f. So, basically what we have to see is the trajectory has to be fitted which will take the joint from θ_0 to θ_f in time t_f.

So, θ_0 to θ_f in time t_f at joints now, because of this joint motion there will be an end vector position change. So, here I have to fit a trajectory which will take it from θ_0 to θ_f in time t_f. So, path is not enough we have to come to motion planning. So, in motion plan I have to worry about the trajectory of motion. So, how do I fit what, what is the equation of the trajectory that will take me from θ_0 to θ_f. So, in two-link manipulator case there will be 2 trajectories this is independent joint control.

(Refer Slide Time: 48:27)



$\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \longrightarrow$ positi.

(Cubic polynomial)

$\dot{\theta}(t) = a_1 + 2a_2 t + 3a_3 t^2 \longrightarrow$ vel.

$\ddot{\theta}(t) = 2a_2 + 6a_3 t \longrightarrow$ acc^n.

Minim Cubil

$\boxed{\theta(t) = a_0 + a_1 t} + a_2 t^2$

$\dot{\theta}(t) = a_1 + 2a_2 t$. const vel

$\ddot{\theta}(t) = 0 + 2a_2$ cost

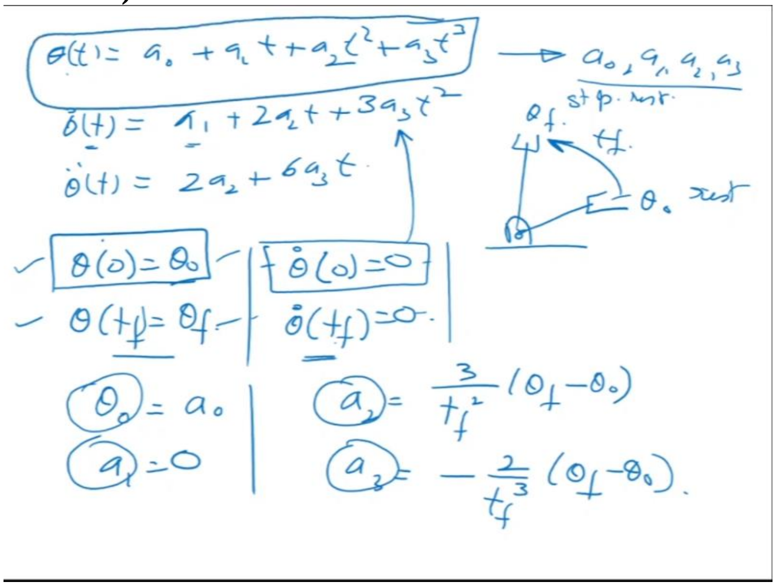
So, here we can say that we are getting a trajectory which is going to be, $\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$. So, these are cubic polynomial now, I cannot fit an equation less than order 3 why, because if I only put in t, then what will happen is if I take $\dot{\theta}(t)$ then what will happen is $\dot{\theta}(t)$ of this equation is equal to so, I take the derivative of this to get so this is position, what is velocity I take the derivative of that which is $\dot{\theta}(t) = a_1 + 2a_2 t + 3a_3 t^2$

I differentiate it again this is my velocity now if I differentiate it again what will I get $\ddot{\theta}(t) = 2a_2 + 6a_3 t$ this acceleration. So, now I have got my position velocity acceleration equation. Now, suppose I had used only $\theta(t) = a_0 + a_1 t$ what would happen is if I differentiate it will become a_1 which is a constant. So, this constant velocity $\ddot{\theta}(t) = 0$, so, acceleration is 0.

So, you cannot represent motion the trajectory by using this equation. Similarly, if I took only $\theta(t) = a_0 + a_1 t + a_2 t^2$, then what will happen is your velocity will become plus twice a_2 t and your acceleration will become equal to 2 a_2, there is a constant. Now, these also are possible why because you know that when a body is when the joint is moving, it will have to accelerate remain constant and then it will have to decelerate again like that.

So, neither can velocity be constant nor can acceleration be constant, obviously acceleration and neither of them can be 0, so it will vary as we move along. So, what I am trying to explain is that you have to take minimum cubic finally, you will take less than cubic polynomial your velocity and acceleration will have problems and you will not be able to move the robot link will not move.

(Refer Slide Time: 50:50)



$\boxed{\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3} \rightarrow a_0, a_1, a_2, a_3$

$\dot{\theta}(t) = a_1 + 2a_2 t + 3a_3 t^2$

$\ddot{\theta}(t) = 2a_2 + 6a_3 t$

$\checkmark\ \boxed{\theta(0) = \theta_0}\ \ \boxed{\dot{\theta}(0) = 0}$

$\checkmark\ \theta(t_f) = \theta_f\ \ \dot{\theta}(t_f) = 0.$

$\theta_0 = a_0 \qquad a_2 = \frac{3}{t_f^2}(\theta_f - \theta_0)$

$a_1 = 0 \qquad a_3 = -\frac{2}{t_f^3}(\theta_f - \theta_0).$

θ_f stp. rst.
t_f.
θ_0 rest

So, now, so we have taken our 3 equations that is $\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$ and $\dot{\theta}(t) = a_1 + 2a_2 t + 3a_3 t^2$. And $\ddot{\theta}(t) = 2a_2 + 6a_3 t$. Now, this has 4 constants that a_0 a_1, a_2 and a_3. So, I need 4 conditions in order to be able to solve them, so when we looked at this manipulator, let us take the example of a single link manipulator.

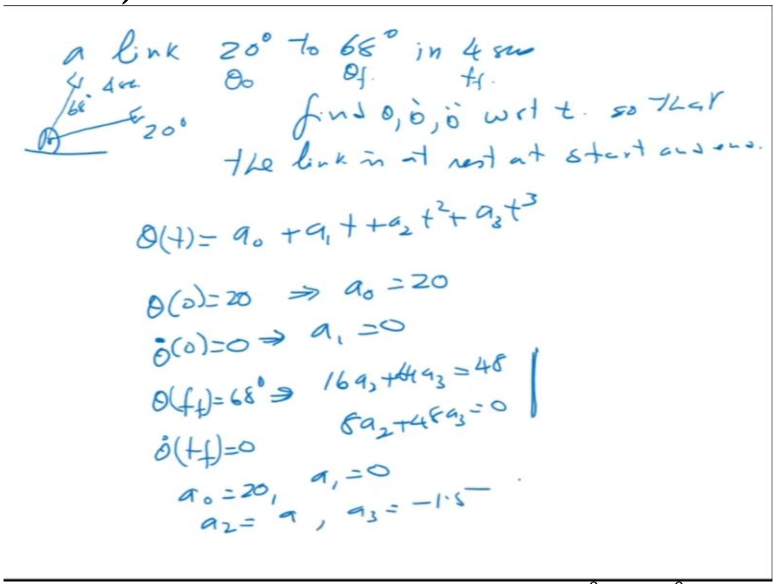
It is moving from let us say from this point this position to this position in time t_f, $\theta_0 + \theta_f$. So, as start position at θ_0 it is a θ_0 at which has reached the final position $\theta(t_f)$, the position is θ_f, so, I have 2 constraints. Now, what about velocity here it is a stop which is a rest. So, before starting this is at rest and here also it stops motion. So, it is at rest again. So, it is a rest means at time = 0.

The body was at rest velocity was 0 and velocity at t_f is also = 0. So, I have 4 equations now, so 4 equations, 4 unknowns here. So, what I can do is I can put this one one of this at a time in these equations and I can solve so, when I put this in, so, I put them in this equations. So, for example, if I take $\theta = 0 = \theta_0$, so what will happen is θ_0 time = 0 t is all 0 = a $_0$. That is from the equation 1.

So, but from here, I get the first one. So, in our physical θ_1 I get the first constant, then if I put the second one here into the velocity equation, what will I get a time = 0 velocity = 0, so, time = 0 means t = 0, so, velocity = 0 means a_1 = 0. So, this constant a 1 will become 0, because I have got to have them straight, then I have 2 more equations, I can put this one and this one, in the position equation and in the velocity equation, and I can solve.

I can solve $a_2 = \frac{3}{t_f^2}\left(\theta_f - \theta_0\right)$ and $a_3 = -\frac{2}{t_f^2}\left(\theta_f - \theta_0\right)$. So, I have solved for the 4 constants. I have solved for the 4 constants a_1, a_2, a_3 and a_4. So, now what we can do is we can write our equation in terms of this. So, what we can do is I know the values of this, this, this and this, provided I know what is the start point end point. What is the start velocity end velocity, so, having known this I have solved for this equation for this equation I have solved the constant values.

(Refer Slide Time: 54:11)



a link 20° to 68° in 4 sec
θ_0 θ_f t_f

4 sec

68°

20°

find $\theta, \dot{\theta}, \ddot{\theta}$ wrt t. so that the link is at rest at start and end.

$\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$

$\theta(0) = 20 \Rightarrow a_0 = 20$

$\dot{\theta}(0) = 0 \Rightarrow a_1 = 0$

$\theta(t_f) = 68^0 \Rightarrow 16a_2 + 64a_3 = 48$

$\dot{\theta}(t_f) = 0$ $\quad 8a_2 + 48a_3 = 0$

$a_0 = 20,\ a_1 = 0$

$a_2 = 9,\ a_3 = -1.5$

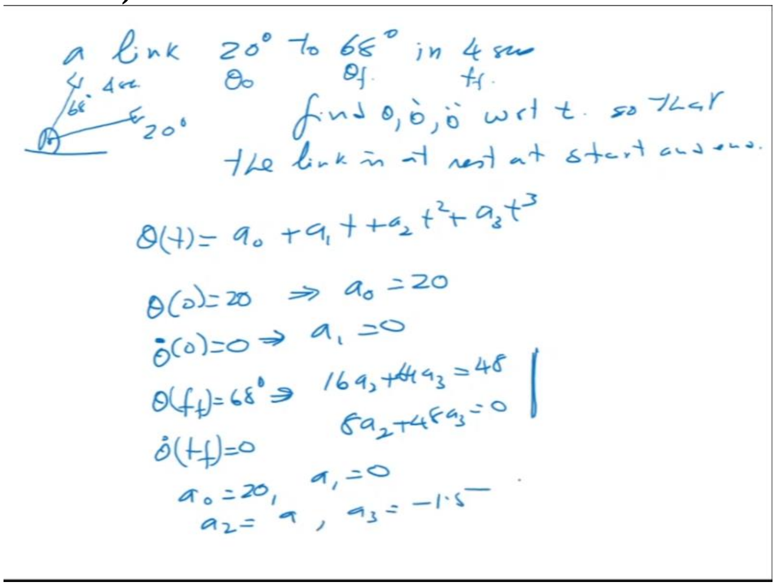
So, let us look at a very simple example a link moves from 20^0 to 68^0 in 4 seconds, let us say we have a single link is moving from 20^0 to 68^0 in 4 seconds. So, there is a single link which is moving from 20^0 to 68^0 in 4 seconds calculate the pulse position velocity and acceleration find θ, $\dot{\theta}$, $\ddot{\theta}$ with respect to time.

So, that the link is at rest at start and end that means velocity at start and velocity at end is 0. So, I can find because you know the polynomial $\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$. And by using the previous values that we found whatever we found here and here, I just put in the values. So, θ_0 so, this my θ_0, this my θ_f time t_f is there. So, I can find out very very easily putting these values I get θ_0= 20. This gives me a_0 = 20, θ_0= 0 gives me a_1 = 0 and θ at t_f = 68 ^{0}.

This gives one equation $16a_2 + 64a_3 = 48$ $\dot{\theta}(t_f)$ = 0 gives the other equation $8a_2 + 48a_3 = 0$.

Now, I have 2 equations 2 unknowns, I can solve them very easily and find a $_0$= 20, a_1 = 0, a_2 = 9 and a_3 = -1.5.

(Refer Slide Time: 56:13)



a link 20° to 68° in 4 sec

θ_0 θ_f t_f.

find $\theta, \dot{\theta}, \ddot{\theta}$ wrt t. so that the link is at rest at start and end.

$\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$

$\theta(0) = 20 \Rightarrow a_0 = 20$

$\dot{\theta}(0) = 0 \Rightarrow a_1 = 0$

$\theta(t_f) = 68° \Rightarrow 16a_2 + 64a_3 = 48$

$\dot{\theta}(t_f) = 0$ $\quad 8a_2 + 48a_3 = 0$

$a_0 = 20, \quad a_1 = 0$

$a_2 = 9, \quad a_3 = -1.5$

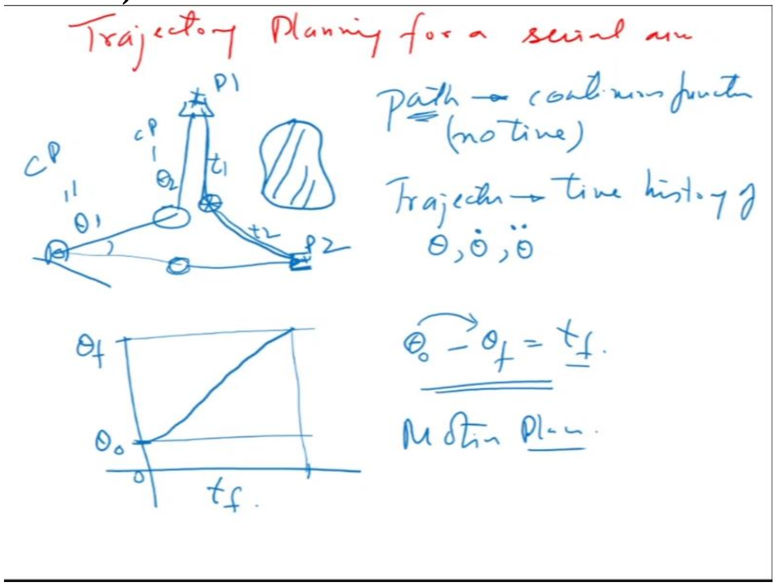
So, if I put this equation into the equation what we do get is $\theta(t) = 20 + 9t^2 - 1.5t^3$ and $\dot{\theta}(t) = 18t - 4.5t^3$ and we have $\ddot{\theta}(t) = 18 - 9t$. So, what we can do now is that, so, I can plot basically, so, it has 4 seconds, time t_f = 4 seconds and this has moved from 20^0. So, I said 20 is here and 68 is there in 4 seconds.

So, now I can plot, now how do I plot so, I divide this into this time instance of 0.1 seconds each, I take the 0.1 second in time, so, 0.1 second intervals of time, put the t in here and find what is position velocity acceleration. So, what we do get is position varies something like this, like this that is my position θ; your velocity will be varying something like this. So, this is a halfway mark and the halfway mark it will be maximum like this, like this and like that.

So, halfway this is not halfway so, exactly halfway mark. So, let us say this my halfway mark, so that is 2. So, this is my θ_0 and my acceleration is going to vary like this, it will come down like this, like this, this is my $\ddot{\theta}$, how do I get this values? I simply took different instants of time put them in this equation for position velocity acceleration, whatever values I got, I plotted them here.

So, what you see that we have got the trajectory of the joints and this trajectory of the joints is the one that will be used by the controller which is going to move the joint from here to here.

(Refer Slide Time: 58:01)


Trajectory Planning for a serial arm
P1
CP
CP
θ2
t1
θ1
t2
P2
Path → continuous function (no time)
Trajectory → time history of θ, θ̇, θ̈
θf
θ0
0
tf.
θ0 → θf = tf.
Motion Plan.

So, in the case of a multi degree of freedom manipulator like the example I showed some time back here, if joint will have one cubic polynomial. So, this will have another cubic polynomial, they will fit the trajectories onto that cubic polynomial and then operate in real time. So, then to operate to take the end effector from the start point to that end. Now, suppose you want to go via this point, then you can go from here to here and then here to here.

So, whatever trajectory we have found for example, in the case of obstacle avoidance, this is an obstacle I want to go from here to here my path is something like this. So, I can put a via point here and say the path will first trajectory will take from here to here that is my trajectory one, the second trajectory will take it from here to here. And this is how the planning is done at the trajectory level or the motion planning system.

So, today we looked at the basic search algorithms that how do we search given a network of paths so, after we find the best path, then you have to go in the path. In the case of a mobile robot it is the robot has to be actuated to go on the path. In the case of manipulator given the end effector path, you may have to find the joint trajectories as by fitting a cubic polynomial. So, we will stop here and then continue in the next class. In the next lecture, we will be looking at motion planning with kinematic constraints and then we will move on to grasping and applications. Thank you.