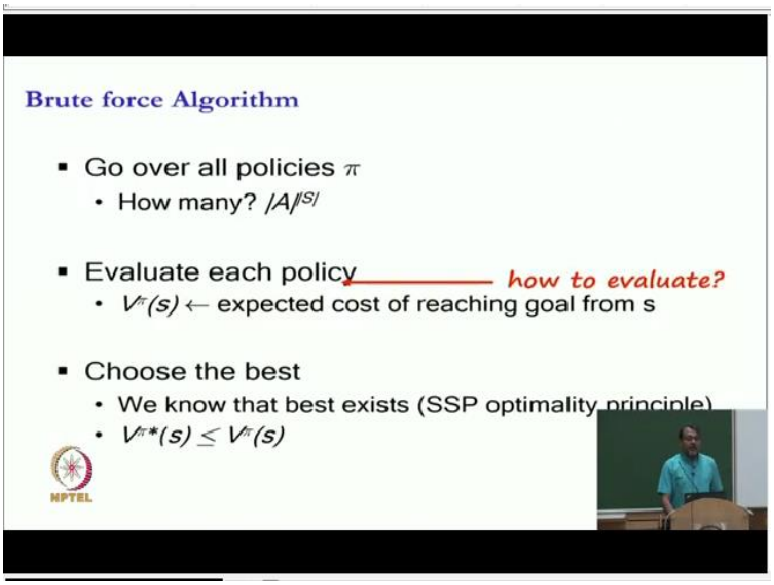
Artificial Intelligence
Prof. Mausam
Department of Computer Science and Engineering
Indian Institute of Science Education and Research-Delhi

Lecture-74
Markov Decision Processes:
Value Iteration
PART-5

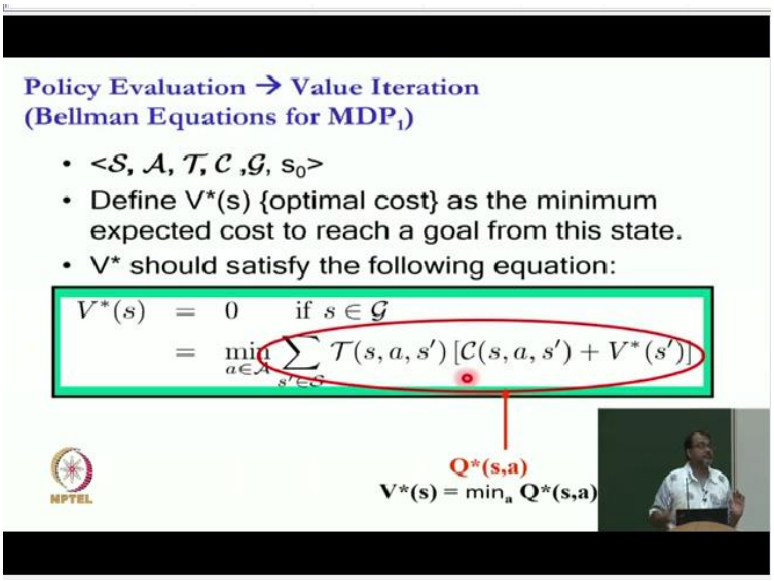
Now, I want to take the leap and the leap is we only started with a brute force algorithm, right. The brute force algorithm we started with was that you start with some you enumerate all policies pi.

(Refer Slide Time: 00:38)



And then you evaluate the policy of the pi and then pick the best. Now, we have solved the brute force problem, because now we know how to evaluate a given policy, but that is not sufficient for us, right, because we cannot enumerate an exponential number of policies. That is just not going to happen. So, what we will do is we will take this an intuition of iterative policy evaluation and convert it into an algorithm called the value iteration algorithm, which would go beyond the brute force.

(Refer Slide Time: 01:16)



But before we do this I want us to write this equation together, because if we can write this equation together you will know exactly what is coming next okay. So, now I am not giving you a policy pi, I want you to compute the value of the best policy, I want you to compute the optimal value starting in state S. So, I am making some changes, I am saying let us say V star of s, is the optimal cost starting in state s which basically means, it is the minimum expected cost to reach a goal starting in state s.

Up until now, we were saying it is the expected cost to reach a goal, if I am following policy pi starting in the state. Now, I have taken away if I am following the policy pi part and then called it the minimum expected cost. And this is the point I want everyone here because this is the most important slide of this whole set of slides. I want to write down a system of equations for V star. And of course, if I am in goal I have to do nothing.

So, V star of s is going to be 0. Now, if I am in some other state s, what will I do first, what will I do, I will take an action. Do I know which action at least I knew which action pi of s, I do not have pi now, I do not know which action, let us say I take action a. If I take action a in status s wherever I reach, some state s prime, if I reach some state s prime how much costs do I pay right away, C of s a s prime. This is what I pay right away.

Now, how much do we pay long term we reach s prime, from s prime also we will take the best actions possible. So, what is the minimum we will pay starting in s prime, V star s prime. So, what is the total we are going to pay, if we take action a in state s and reach state s prime, is this term C s a s prime + V star of s prime. Now, there are 2 problems. Problem number 1 we do not know s prime, problem number 2, we do not know a.

Think logically, over s prime, what do we have to do is s prime in our control or not. Yes or no. No if it is not in our control, what do we take expectation. So, we take expectation over s prime. This is almost very close to where we were earlier. However, we also do not know a and a is in our control, if a is in our control what will we take, the minimum we have the cost that we want to minimize, we will take the action which gives the minimum cost we are going to compute a min.

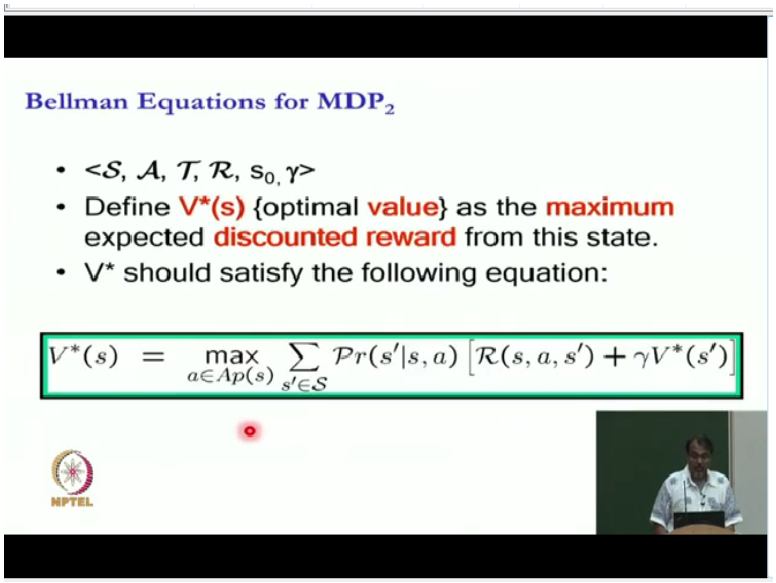
And this is the system of equations that governs a full Markov decision process. If you have learned how to write this system of equations, everything else is detail. What is this term without the minimum. This term is a function of s and a. And this term says that in state s I took a and did the optimal thing ever afterwards, why because in s prime, I am still taking V star. This function is called the Q star of s, a.

Again, q star of s, a is defined as in state s, I took action a and then did the optimal thing in the future. And you can also write V star of s as minimum over a Q star of s, a. That is the same thing. This general principle of defining this set of equations and by the way, these are called Bellman equations. It is very standard phenomenon you should be able to write it for a slightly different version of an MDP.

Let us say we are in the maximization world, we have rewards and we have infinite horizon, but discounting. So, what will change. First of all I am not given the cost function I am giving you the reward function. So, let us first define V star of s, what would be V star of s, it is the maximum expected discounted reward starting in state s. Second change will I have this base case, I am not giving you a goal.

I am just saying that your life is to just keep accumulating reward over long term. So, will I have this best case, I will not have the best case. The other change I will make is that this min becomes max, this cost becomes reward, one more change, important change, think about this, I am getting some immediate reward r. Then I have this state s prime and whatever long term reward I get all of this is multiplied by a discount factor gamma. Because that is all happening one step down.

(Refer Slide Time: 07:26)



So basically my set of equations become this. Now, let us take a couple of minutes and sort of intuitively think about what is this model telling you, what is this model saying. The model is saying take that action that optimizes your long term cost, not just your immediate cost, but your long term cost. Now how many of us love to eat junk food. How many of us know that junk food is bad for our health.

How many of us still like to eat junk food, why, what are we optimizing, what is happening if we want to model this phenomenon. My action is to eat junk food. Of course, I think this action is the optimal action at this point in time. Would this set of equations, would that give you this action as optimal or can that give you this action as optimal, depends on the value of gamma Vibav says, let us look at this set of equations.

Which value of gamma allows us to say eating burger is the right thing to do for me today, gamma is 0 and a person who never eats a burger for them gamma is sort of like close to 1. So now think about what is going on. See this gamma is intuitively telling you what is your effective look ahead. How far are you willing to see. Because further than that, you know, the gamma rays 2 power n may becomes so small that it may start becoming meaningless.

So people who love to do things because, you know, let us say their emotions say that they have to do something without thinking about the long term consequences, a Markov decision processes says I can handle you, your look ahead is practically 0. You are only looking at one step, the next step and you are only interested in the immediate reward that you will get right. Whereas there are people who would you know, detach themselves from themselves.

They will take the long term view, they will say okay, I can do this now. Now I will enjoy it. I can go and play you know, video games today, but I will do badly in my exam tomorrow. So let me not play the video game today okay, and for them the gamma is higher. And one cornerstone of a Markov decision process is how far you are able to look at it or in other words, one cornerstone of intelligence is how far you are able to look at.

Somebody who can reason for the longer period of time now often what happens is nobody knows exactly how the probability is gonna turn out, nobody is able to see what is going to happen far into the future, but somebody who is able to analyze the future better, we intuitively call that person more intelligent. And the Markov decision process is getting at exactly that, because the V star of s prime term is sort of giving you the long term computation.

Of course, in a Markov decision process all the probabilities are known. So therefore, if we believe that the ability to look far into the future is one important aspect of intelligence, and MDP gives you that right from the beginning, first thing. The second thing and I believe it is a very interesting observation. That Markov decision process is very closely aligned to Gita. Because our favorite sholg from Gita is (FL) what does that mean.

Your control is in the action. Not in the pal, pal you can say outcomes. You can only control the action not the rewards, not the outcomes, and that is exactly what Markov decision process says. You have a choice of the action you can choose the best action. But after you take the best action you cannot pick the outcome. The outcome is designed by the nature and you have to take an expectation on top of it.

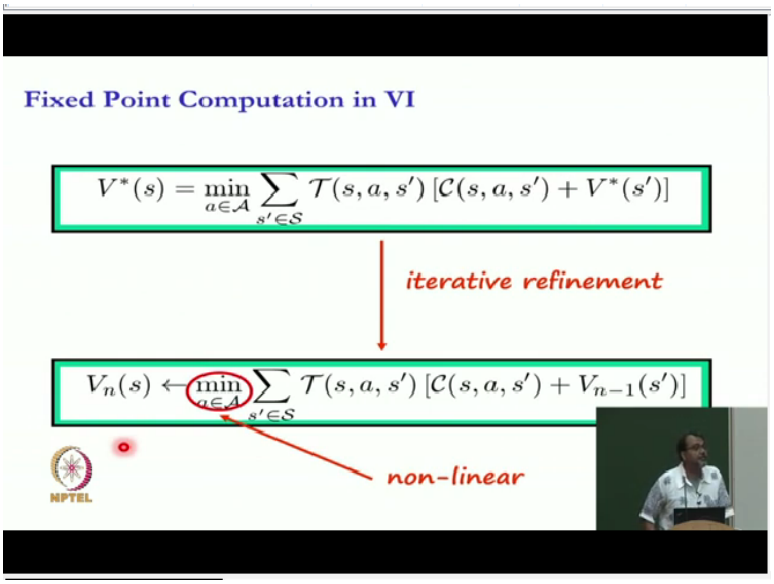
Of course Gita does not say that take an expectation on top of it. But it sort of knows it. However, Gita does not align in Markov decision process, in its second line of the same verse. So it does say that you should have control of the actions not in the outcome. But it goes on to say, do not do your action in the hope of the reward. And neither should you do in action. And Markov decision process makes no such commitment.

It says, if doing nothing has the optimal reward, do nothing. And always take the action that optimizes your long term value function. Now I will leave it to you, leave it to your philosophical mind to come and see if you can align these even better. And if you can, please let me know. But at least what is nice is that MDP is in sync with Gita's main point, that look you have to focus on the action and not the various outcomes that can happen.

That is it is very nice. It distinguishes between what is in your control and what is not in your control. And that is exactly what a Markov decision process does. So in some ways, it is a nice model right. The other good thing is that we already know how to solve a Markov decision process because we have already done iterative policy evaluation. There we were given such a system of equations like this.

And from there we created a set of approximations, which will lead us to the solution. So we will do exactly that.

(Refer Slide Time: 13:52)

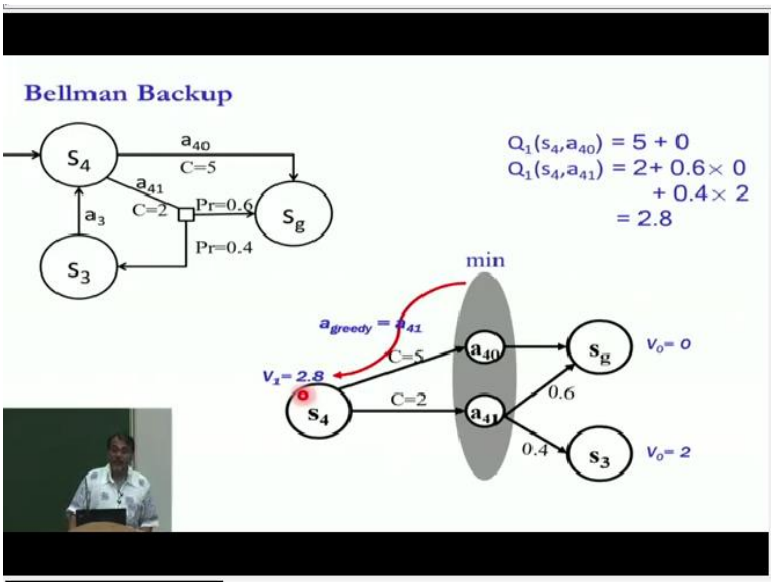


We have let us say the original value function equation which is minimize over actions, take an expectation over C s a s prime + V star s prime. From there we say let us randomly initialize V star as a V 0 function. And if I give you n - 1 V n - 1 function, then using that in the right hand side, I can compute the left hand side the better approximation for V of s. Again, notice that the previous state of my algorithm is used in computing the next state of the algorithm.

If V n is a state, you can think of it that way. Second, notice that I have a min here and because of this min, these equations become nonlinear. So earlier, we had a system of linear equations. Technically, we could have solved it using any of my matrix manipulation methods, but I also did the iterative algorithm, not because I absolutely needed it there. But because I absolutely needed here.

Here, I do not have a choice, because I do not have a system of linear equations, by adding them in, these equations become nonlinear. Let us so get a quick example. So let us say this is our domain.

(Refer Slide Time: 15:23)



And now I have to figure out it for each state which action do I do. So I will define the concept of a Bellman backup, this Bellman backup, would be running this equation for the state s okay, so let us say I want to compute Bellmen backup at state s 4, let us say, I give you some V 0 function, which is the goal has value 0, and s 3 has value 2, and this is random initialization. And now I want to compute V 1 of s 4.

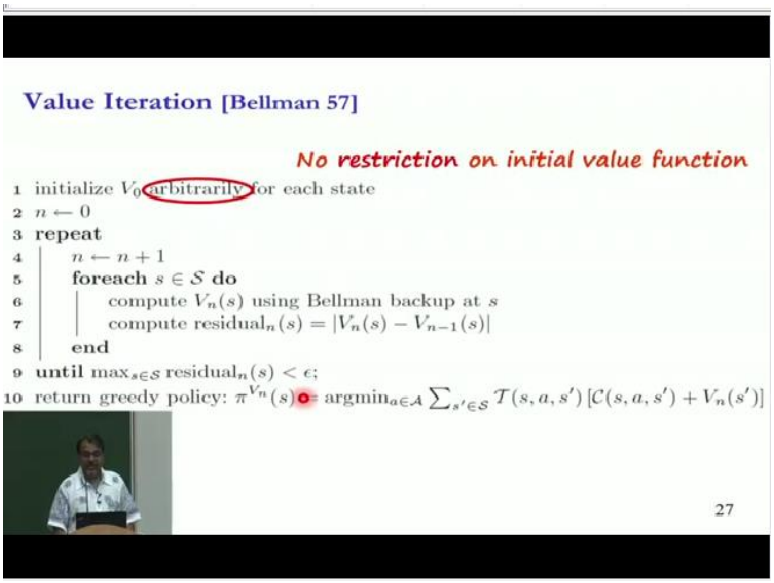
Why V 1 because I am using V 0's in the right hand side so I will be computing the next approximation. Now to compute that first I will compute the current value of Q S 4 a 40. This would be using the current approximation of V 0 and back sending the value basically sending the value. So, the goal sends its value 0. So, the action a 40 looks to be about 5 + 0 right, immediate cost 5, long term cost 0.

I could alternatively take action 41 that would be 2 + 0.6 times 0 + 0.4 times 2. Now notice that S 3 may not have value 2 but that is my current approximation. When I repeat this many, many times it will finally converge. So now, I know that currently based on my current information, a 40 looks like cost 5, a 41 looks like cost 28 expected costs 28. So, I will take a min over the 2 actions that I am given.

So, that gives me the greedy action as a 41. We call it the greedy action, because at current state of the world this is the best action I know, current state of the value function, this is the best

action I know, this is also called the one step greedy action or one step look ahead, greedy action or a one step look ahead action. So, when I give you a V n - 1 function, you can always check which action is the greedy action for every state using this one step look ahead similar. This allows me to compute V 1 of s 4 as 2.8. This is completing one bellmen backup.

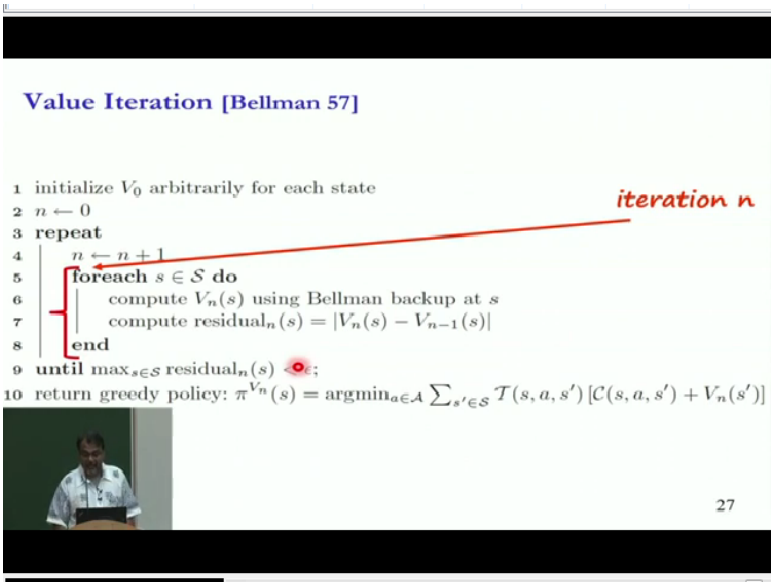
(Refer Slide Time: 17:55)



And now using this we can create our first algorithm a good algorithm for solving in a Markov decision process. It is called the value iteration algorithm. Now, in this algorithm we first initialize V 0 arbitrarily, again, no restriction on the initial value function, you can initialize basically any finite number you can initialize, it will all converge. Somebody asked me at the end of last class, that if I initialize it closer to the optimal, would it not converge sooner absolutely.

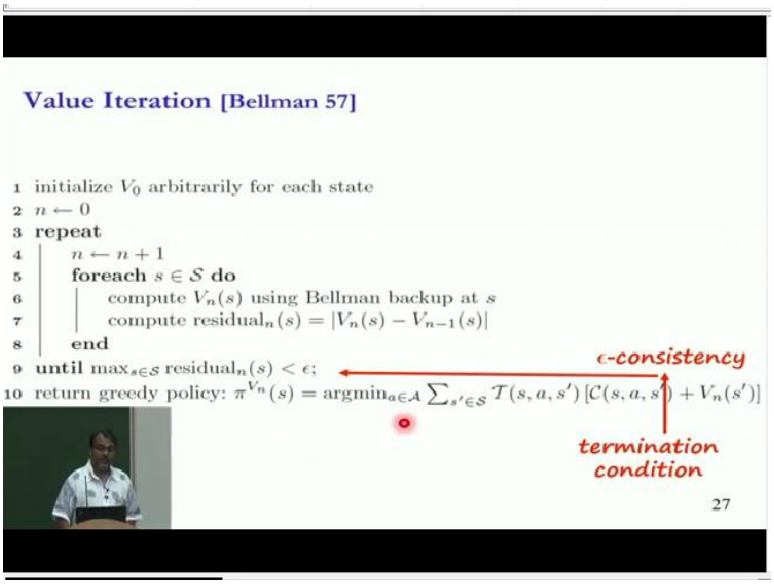
If I initialize it as we start then it will converge in one iteration. And if I initialize it closer, it might converge much faster. So, yes, and there is theory behind how to do this, but how to prove these, but we will not get into that.

(Refer Slide Time: 18:41)



And then we will have the nth iteration and in the nth iteration again, we will go over each state. And in each state, we will compute the new approximation V n, using the V n - 1 function and the Bellman backup equation. And again, we will compute the residual, residual will tell me how much did the value of this state change. And again, I will stop when my max residual is less than epsilon.

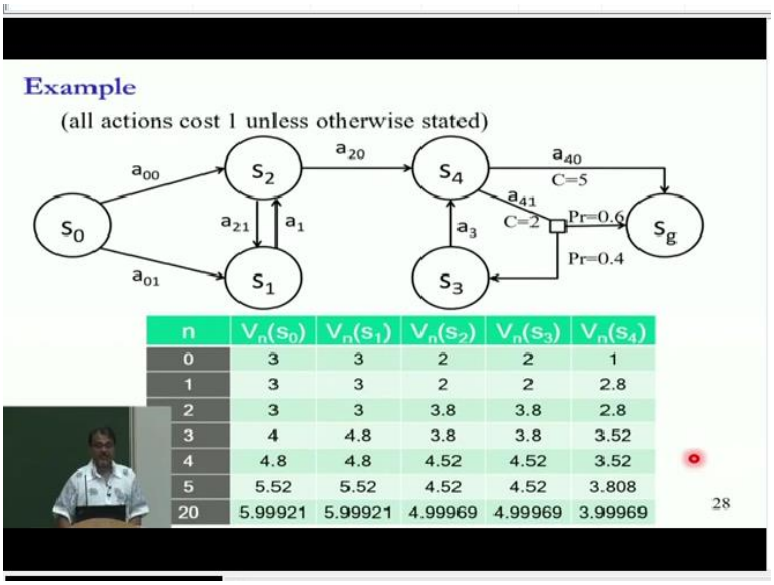
(Refer Slide Time: 19:08)



Again, it is the epsilon consistent value function is what this algorithm computes and that is extermination condition. Last but not the least, what do we return. We do not return the V n function because our goal is to return the policy, not the value function. So, we have some V n function, we will do one step greedy look ahead and output the policy. So, that we will call pi of

V n. That is, if I use V n here. And do an argmin instead of min which action is the best and that is my policy that I have.

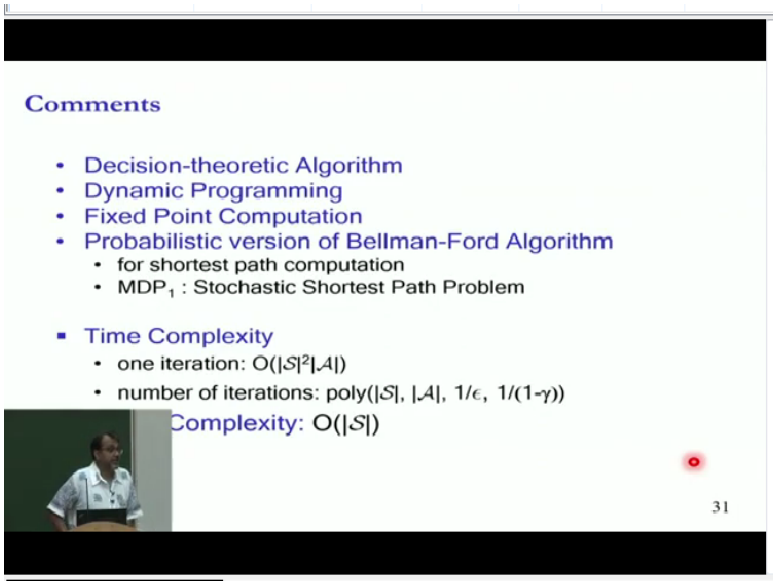
(Refer Slide Time: 19:47)



n	$V_n(s_0)$	$V_n(s_1)$	$V_n(s_2)$	$V_n(s_3)$	$V_n(s_4)$
0	3	3	2	2	1
1	3	3	2	2	2.8
2	3	3	3.8	3.8	2.8
3	4	4.8	3.8	3.8	3.52
4	4.8	4.8	4.52	4.52	3.52
5	5.52	5.52	4.52	4.52	3.808
20	5.99921	5.99921	4.99969	4.99969	3.99969

So, for example, if I randomly initialize everything does not matter how I initialize and I run value iteration, then over a few iterations as you can see n going from 0 1 2 3 4 20 you start to see that these numbers are converging and these numbers converge to as arbitrarily close to the optimal as you want determined by epsilon okay. So, some comments on this algorithm.

(Refer Slide Time: 20:13)



It is called a decision theoretic algorithms. Go back to what we discussed in decision theory. We said what is decision theory, decision theory is when we assign values to each decision and take

the best action after that it is a principled way to figure out which is the right action to make take and this is the decision theoretic algorithm. It is a natural extension to what we studied in the previous set of lectures.

It is also a dynamic programming algorithm. Why because at every point we have the n - 1 value function that we have stored and we are using it to compute the nth approximation therefore, it is a dynamic programming. It is also one way to achieve fixed point computation. So, if you think about what is going on this particular set of equations. This particular set of equations is we are trying to find that V star which satisfies this particular set of equations.

Think of it as the point right V star determines V star. It is sort of internally consistent, sort of equations. That is the fixed point. And we start from some random point. And we start to get closer and closer and closer to the optimal point. This is called fixed point computation, we are trying to get as close to the fixed point as possible. And we are doing it in some steps by applying this operator many, many, many times.

You can think of it as the probabilistic version of the Bellman Ford algorithm. This will require some thought, but what happens in a Bellman Ford algorithm, you have some of the V n function and you send it across the neighbors. And they compute their new V n + 1 function and you do it n times. And you are guaranteed to converge. That is sort of what happens in a Bellman Ford algorithm.

Bellman Ford obviously has no probabilities. It is a shortest path algorithm. So when you try to extend Bellman Ford to stochastic graphs, that is called a stochastic shortest path problem and the natural extension to Bellman Ford would be value iteration okay, because there is cyclic structure, that you can come back to the same state therefore, you will not be able to finish value iteration in n steps, you may go on for longer. That is it. You can also compute time complexity. So, what is the time complexity of one iteration.

Let us look at this. Let us look at the time complexity of a Bellman backup first. For one state, I have to compute the new approximation, what is the time complexity, I will first go over all. All

actions and for each action I may have to take expectation over how many states, all states in the worst case you know go you can check this, see this is one Bellman backup, I have to do min over A. So, I have to do A amount of computation and have to sum over S.

So, I have to do S amount of computation. So, one Bellman backup is of order as s a in one iteration I do s Bellman backup, so, therefore, my total time complexity becomes s square. Similarly, how much space complexity do I have at any point in time I have to maintain the V n - 1 function and V n function, that is order s and then what you cannot prove but can be proven I mean you and I cannot prove it easily requires some more theory of dynamic programming is in the number of iterations will be polynomial in S A, 1 by epsilon.

And 1 over 1 - gamma. So epsilon is too small, obviously, when number of extra iterations increase, if gamma is too close to 1, that means that I am expecting a much longer look ahead, then it will require much more competition. Questions on value iteration, yes. You could have multiple fixed points. So Sruthi asks, can I have multiple fixed points for this set of equations. And that is a deeper question than you initially set out to us.

But under the assumptions that we have defined, so what are the assumptions that we have defined for the value iteration algorithm, well, I skipped here, the assumption is that there should exist at least one proper policy, proper policy means a policy that is guaranteed to take me to the goal from every state. So, if there exists any one proper policy and all costs are positive, then this will converge.

Now, if that does not happen, for example, if there does not exist any one proper policy then you have to change our objective function, because then we do not even know what is the meaning of optimal cost right, because there are 2 objective functions probability of reaching the goal and expected costs of reaching the goal if I reach the goal right. So, then that becomes a dual optimization problem which we will not talk about.

There is another complication, sometimes what can happen is costs maybe 0 or negative, if costs are negative and you have cycle, then you might actually keep accumulating lots of negatives,

and you may get cost to V - infinity that also creates trouble. Same as in the Bellman Ford algorithm. You do not want a negative cost cycle, right. That is the other problem. But if costs are 0, that also creates some trouble.

And therefore, we will in our world not get into all those complicated MDPs. But I can say that, you know, I have been part 8 to developing some of the theory in this 4, so one of my PhD students, Andre Columbo, who is now at Microsoft Research. He did a bunch of work to expose what is the difficulty level of each kind of MDP. So, he defined a series of problems, stochastic shortest path generalization of stochastic shortest path.

And then in the most complicated, well formed problem he called, he did not call somebody else who have done it and that became equivalent to stochastic longest path problem. So, the longest path problem is an NP complete problem, whereas the other problems are polynomial in the state space okay. So again that takes us into territory that becomes harder but for our purposes if my costs are positive.

And there exists at least one proper policy this has only one fixed point. Moreover, if my gamma is less than 1 there is only one fixed point. In cases where gamma is equal to 1, and costs start to become 0 life becomes complicated and that is what we are not going to study. But I can point you to references if you are interested in, any other question. If gamma is less than 1, it is always guaranteed to be well formed the MDP is always guaranteed to be well formed.

And in fact, the reason is very clear. The maximum reward from any point is always less bounded by max reward divided by 1 - gamma right, because my value function is always well formed. So if you do not have goals, if you have discounted reward infinite horizon problem, life is easy. Who would have thought that indefinite would be harder than infinite. Who would have thought that infinite would be harder than indefinite which one do you think.

Indefinite horizon is harder or infinite horizon is harder. How many say indefinite, how many say infinite. A little bit of a split, but actually it is true that indefinite horizon is a harder problem than infinite horizon, believe it or not. And the reason is discount factor because an indefinite

horizon I can always say discount factor as 1 because I am guaranteed to stop somewhere. Infinite horizon it never stop.

Therefore my discount factor is always less than 1, that makes the problem much easier, believe it. Anyway, there is a full theory behind Markov decision process just so you know, Bellman came up with Markov decision process formalism in 1957. This was developed in the operations research community. And then later, the AI community took this on in the late 80s as the model for long term decision making under uncertainty.

And after that there is a lot of activity that has happened in the field of AI using the MDP formalism. So we have made it our own. At this point, we do not think of MDPs as a model from operations research. But once upon a time it came from there. I also told you my personal story when I came back to IIT Delhi, made one of my senior professors they asked me, what are you working on I told him, I am working in a AI.

And they gave me a smile like because that AI has been a bad word for a very long time. And you know, 10, 15 years ago AI was still considered weird thing to do with the people who are not rigorous enough do AI. And so then later one of the persons asked me, okay, what are you doing. So I told him, I am working in this MDPs and so on. So that is my PhD area. He said, oh, you are working in MDPs.

Then say you are working in operations research. Why do you say you are working in AI, you are short selling yourself, from that point to today, where all of you want to take AI just because you can get jobs and you know, become be rich and famous, which is the goal of everybody's life anyway. Think about how much transition AI has had and again, it may go down tomorrow. We know this because we have seen many vendors right.