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

Lecture-6
Constraint Satisfaction Problems: Backtracking Search, Part-3

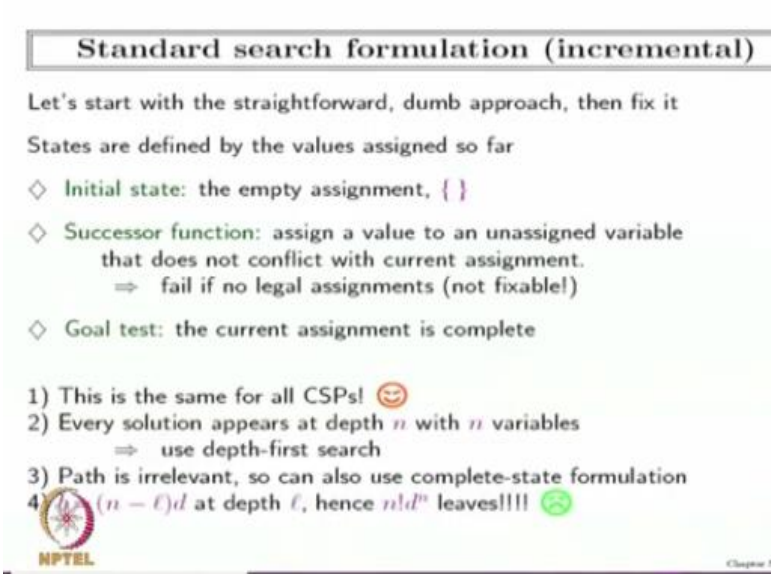
Now, the first thing we need to do is let us say if we can define a search formulation for CSPs. If I have to solve this problem using search, systematic search what will be my search space what will be a state in the search space, assignment to all the variables. Now if I always say assignments to all the variables are my talking systematic search, a local search remembers this is a difference we studied earlier.

When I usually solve the problem with local search I search in, I move in the solution space. But when I am using systematic search, I typically do partial assignment space that is typically what I do. So, I will work in the partial assignment space in the partial assignment space, my initial state would be, no assignment whatsoever my initial state would be I have not assigned any variable whatsoever what will be my successor function?

Let us say I have already assigned L variables what will I do? I will have to assign some other variable how many options do I have? n - L options for each variable, how many possible values do I have? d values. So how many children do I have n - L into d children simple stuff makes sense and all the solutions are that which depth all the solutions final solutions are at which depth n depth when all the variables have been assigned.

Because all the solutions I had to fix depth I can sort of use depth first, there is no infinite search tree here is the difficult question how many levels do I have? In this way that I have a define the problem how many children at the top level n into d. How many children at the second level n - 1 into d, how many children in the next level n - 2 into d and all these trees they get keep multiplying because for every parent I have these many children and so on so forth. So, how many totals levels n factorial d to the power n how many total assignments in my problem d to the power m, what is going on.

(Refer Slide Time: 03:22)



Let us take a second think about something is going on we are not doing something. So, let us so first let me just revise what we discussed. Let us start with a straight forward dumb approach level that fix it states are defined by partial assignments, initial status, the empty assignment successor function is assign a value to an unassigned variable that does not conflict with the current assignment.

Yes, it is constraint satisfaction problems so, every time they assign a value, let us not assign a value where we have in conflict with the previously assigned values. So then at some point, if I cannot assign a variable a value then I backtrack gold testers current assignment is complete if it is complete all the constraints would be naturally satisfied because we are checking for constraints at every step.

This algorithm is going to be same for all CSPs irrespective of whether the variable means the location of a tile position of a student or what you know the timing of a class whatever it is, every solution will appear that depth n. So, you will have depth first search which is a good algorithm. However, we will have n factorial d to the power n levels were the only d to the power n possible assignments.

What am I doing wrong? Why am I creating more levels than the number of states? Why do we have more states? In search each state represents the exact path we took to each state we know this right we have done this in even uninformed search. Now here is the path important who

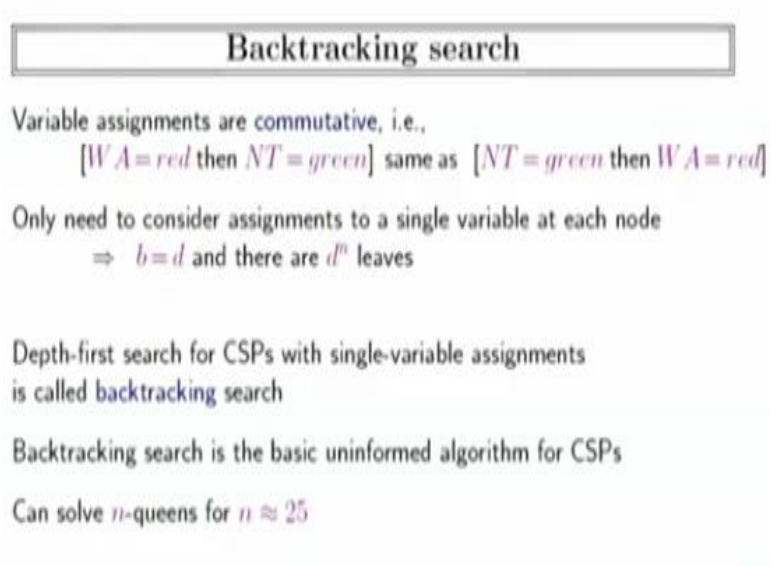
cares when they assign X1 true and X2 false first or I assign X2 false and X1 true in a different order, it does not matter.

It is the same state at the end of the day, but it will be a different path. Because first we will be assigning X1 and then X2 in the other alternative will be first assigning X2 and then X1 and therefore we will be creating not only all the state positions, but for each position we will look at each permutation of which specific order we assign them, but who cares for the specific order? We do not want this n factorial there, how do we fix it?

Go in order the variables assess, always assign X1 first and X2 second, will this work? It will work. Will it be a good idea? Because now I am not even thinking about what is the right order to assign these variables I have just fixed this order I do not quite want that. I do not want to have a global order particularly but I can still do something locally the small change to what I suggested somebody at a given partial node pick one variable.

Do not look at all variables, pick one best variable wherever this best comes from. You do not have a global best you are at a place where X1, X2, X3 have been assigned, then you say okay I can assign X4 I can assign X5 I can assign X6, let me assign X6 now. So at every point we think about which is the best variable to assign and then I look at all its values. When I do this, I will still get d to the power n levels.

(Refer Slide Time: 07:20)

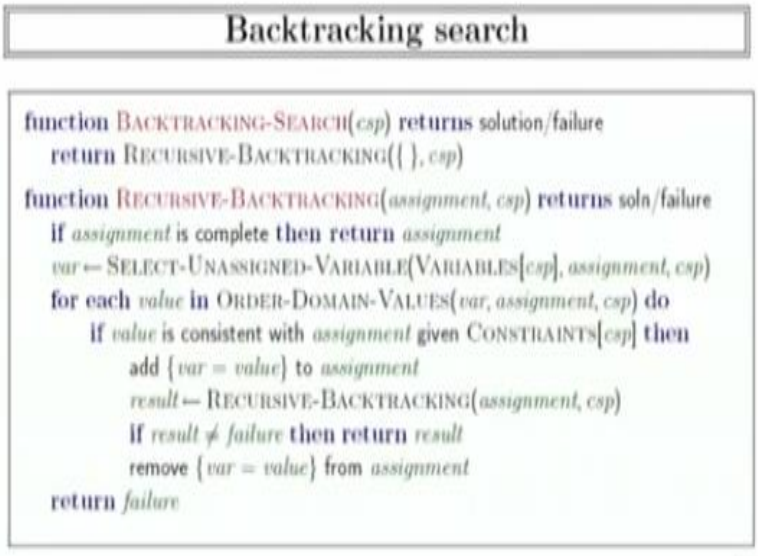


Which will solve my problem and this particular algorithm is called backtracking search. I know depth first search backtracks, it is a depth first search algorithm but the name backtracking search specifically refers to those depth first search solutions where at every node we only look at single variable assignments. Which single variable we look at may change in different nodes, but at every point, we only look at single variable assignments.

So at the top, we will pick some variable and then we will have d children. At the any next level, we will at 11th level, I will look I will pick one variable and then have d children. So, this gives me d to n levels, which is what I wanted and this club solution alone believe it or not, can solve n goings for n = 25 we will also do solutions which can solve n queens for n equal to thousand, so, there is more to come or even 10,000 and they will be different solutions, right, hopefully.

So, this is called backtracking search, you are basically going to do depth first search at every point, pick a variable and then assign values and then check if constraints are satisfied. If so, keep moving and eventually if they are not backtrack and then eventually find a solution that works for you this is called backtracking search.

(Refer Slide Time: 08:54)



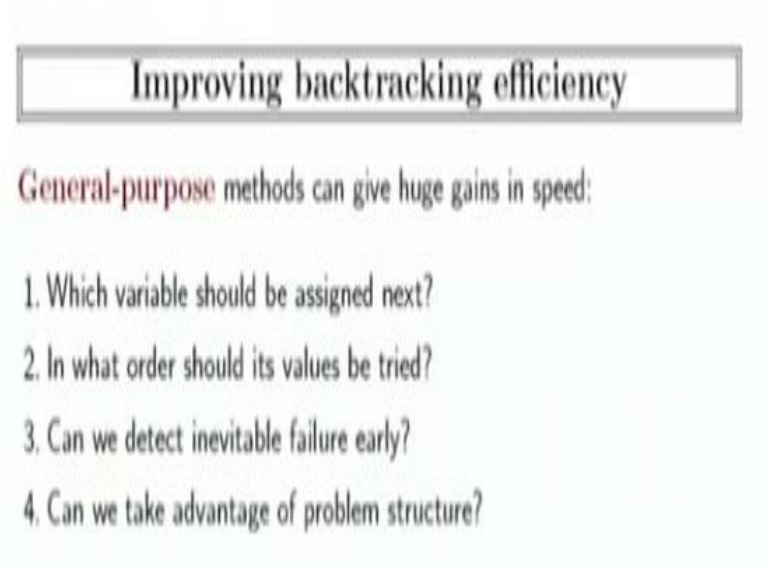
Now, and this is the algorithm you can read the algorithm I will just flash it on the screen basically you say select a variable v and then for each value, assign it check for whether it satisfies all constraints or not. Now, here is the question I want you to think about I have given a

general purpose algorithm here. What tricks of the trade I can bring in to make it even faster in practice, what are my levels? What are my buttons? What am I knobs?

What can I change at any point? What can I do differently to give me a better or worse algorithm? Can you think about it? The way you select the variable, well I have a chop and option which variable do I select first? Maybe there is some intelligence in which variable do I try first? Another thing, another thing which value do I try first, kamal say which value do I try first not only the variable.

But which value I want to try the variable with hopefully, gives me a solution first quickly, right? I do not have to backtrack so these 2 were simple ideas.

(Refer Slide Time: 10:18)



These are 2 other ideas that we will study and one idea is, can I detect failures early? Can I say that I have done some assignment even though there is no constraint violation right now, 5 steps down the line I will get a constraint violation or do not go 5 steps further? That is the most important magic in constraint satisfaction and we will talk about that and then lastly, can we take advantage of the problem structure so next class, we will start exactly from here. Thank you.