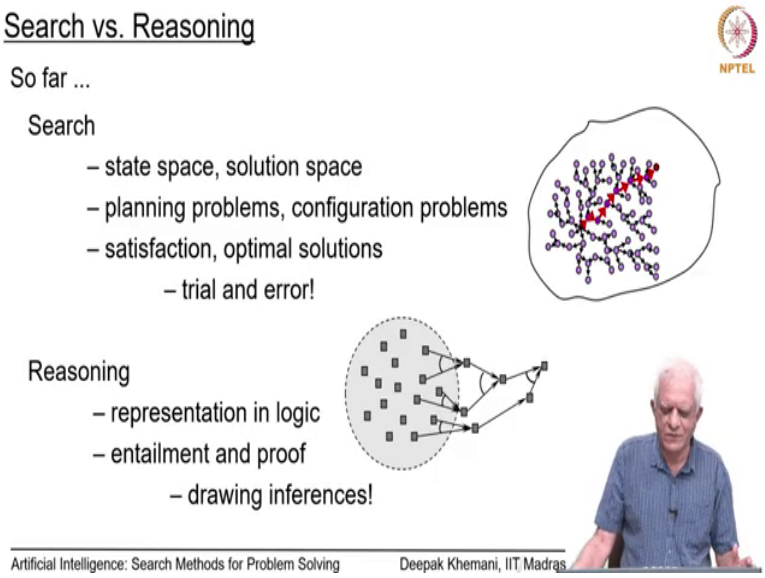
Artificial Intelligence: Search Methods for Problem Solving
Prof. Deepak Khemani
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Chapter – 09
A First Course in Artificial Intelligence
Lecture – 88
Constraint Processing
Constraint Satisfaction Problems

So, here we are on the last leg of our course, the last lap I might say and in this week we want to look at this area of Constraint Processing or which is also called is Constraint Satisfaction. And we want to basically highlight the fact that search plays an underlying role in constraint processing, but also in the reverse direction constraint satisfaction is a way of solving problems in which you can combine search with reasoning.

We spent about a week or so, looking at reasoning representation and reasoning using (Refer Time: 00:59). And, in this week we will see how you can you know combine search and reasoning to kind of solve problems in the most efficient way. So, this is from my Chapter 9 of my book, but if you look elsewhere you will find complete books on constraint processing and in fact, you can have a complete course as well as we do in this NPTEL series ok.

(Refer Slide Time: 01:33)



So, let us get going. So, far we have looked at two things; search and reasoning. So, in search basically we explore some space, it can be a state space or it can be or solution space, we can be solving a planning problem or we can be solving a configuration problem, we may be looking for a solution or we may be looking for an optimal solution.

So, in SAT for example, if we remember any solution is as good as another whereas, in other things like TSP or path finding optimal solutions are based, but the theme of search is trial and error. You try out different options mentally in some sense and then select the sequence of decisions or actions that you will make essentially.

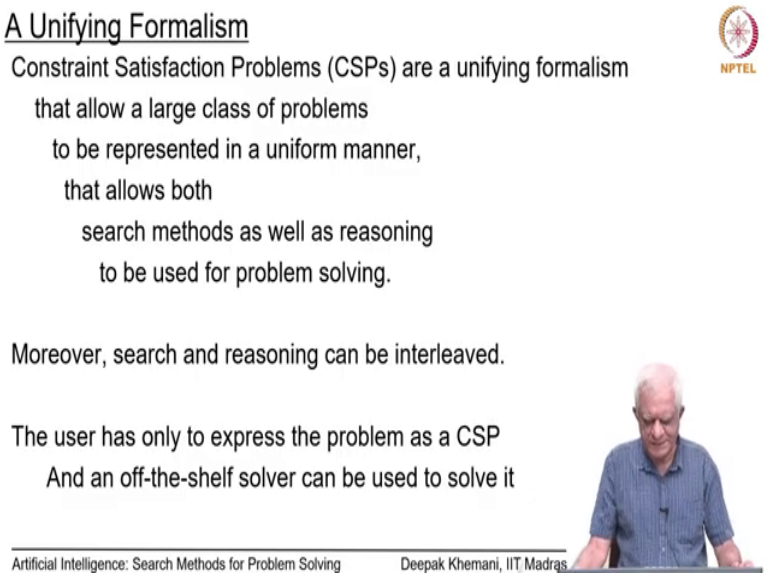
Then we spent a week or so, looking at now representation and reasoning and essentially we saw the notion of validity, the notion of entailment, the notion of proof. And then we kind of

underline the fact that finding a proof is essentially a search based method essentially, but the key features was drawing inferences.

So, you can as I said earlier also sometime you can imagine an intelligent agent being a layered creature essentially. So, as a lower most layer we have search operating, but at a higher level you have maybe reasoning and other processes happening. This is very similar to what happens in nature because the lowest model low lower most layer in nature you have physics which is you know how atoms interact and so on and so forth.

But, as you go up as you go up the taxonomy or hierarchy chemistry comes into play how molecules combine and interact with each other. Then biology comes to play as to how you know larger molecules interact in special ways and so on and so forth.

(Refer Slide Time: 03:32)



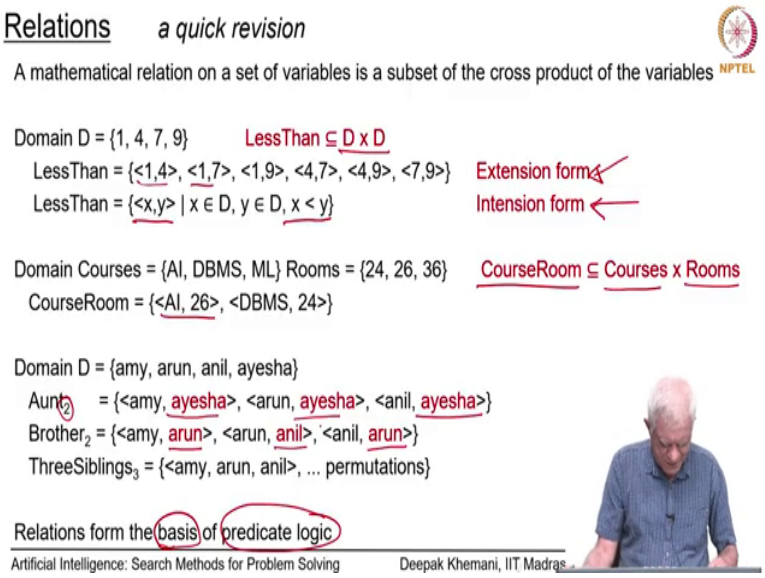
So, what we are going to do in this week is to look at a unifying formalism and that formalism is expressing problems as constraint satisfaction problems. And CSPs as we will call them as unifying are a unifying formalism that allow a large class of problems to be represented in a uniform fashion.

And that allow both search methods as well as reasoning to be done for problem solving essentially. So, in some sense it unifies the two approaches that we have been looking at. And what is really interesting about working with CSPs is that you can interleave search and reasoning that you can do search at some point and you can do reasoning at a different point.

And we will see in this week 1 course one algorithm which combines search with reasoning and we will see some research and reasoning independently as well for the same kind of problem.

So, the idea is still the same as we have had throughout this course that the user expresses a problem in some form and some of the self solver will solve it essentially. So, earlier we talked about algorithms like A star, now we are going to be talking which are used for constraint solvers essentially.

(Refer Slide Time: 04:52)



Before we move on to constraint satisfaction problems let us do a very quick revision of relations. Relations is foundation of discrete math's and everybody who is exposed to a little bit of computer science would know. So, this is just a quick relation, quick revision. And as you know relation is or a mathematical relation on a set of variables is a subset of the cross product of the variables essentially.

It can be only on one variable also. It can be on more than one variable and we have seen some of those things. So, let us look at numbers. So, for the sake of brevity we will just use a small domain in which there are only 4 numbers; 1, 4, 7, 9 and we want to model the relation which is LessThan. Now, of course, we are familiar with all this stuff and it is as I said a quick revision.

Now, the typical way of representing a relation is in the intentional form that is what we are used to doing. So, essentially the core of our representation is the fact that there are 2 variables, x and y they come from the domain D and they satisfy the property x less than y. And if they do then we have tuple of made up of x and y which is a subset of as we have said here D cross D essentially. So, not every element of D cross D belongs to the relation only those in which x is less than y.

Now, in this week we will confine ourselves to relations which are expressed in extension form. So, which means they are explicit that we state explicitly, what are the pairs of numbers which belong to this relation. So, for example, 1, 4 belongs to the relation; 1, 7 belongs to the relation and so on, we list out the whole thing explicitly. And the reason we are going to focus on this is that algorithms that we will see much easier to explain using the extension form of relation representation.

If you are building our system and you have to have a large domain then of course, you cannot rely on an extension format. You can always write programs which will convert them from the intention form to the extension form essentially.

So, this is a relation on a set of numbers in particular 4 numbers and the LessThan relation. Here is the relation between courses and rooms essentially. So, we have the domain of course, is in which we have listed 3 courses here; AI, DBMS and ML and we have 3 rooms in this example, does not have to be 3 as long as we are talking about a subset.

And in this particular example we are talking about a relation called which course is in which room and it is a subset of courses cross domains as you can see here and it is again a pair in this case. Again this is a binary relation between 2 elements and the relation basically says for example, that AI is in room number 26 and DBMS in room number 24 and that is what this one says you may have more courses and more relations.

Here is a another domain which is the domain of human beings. There are 4 humans here; amy, arun, anil and ayesha and there are 3 relations shown here. The first one is Aunt and if

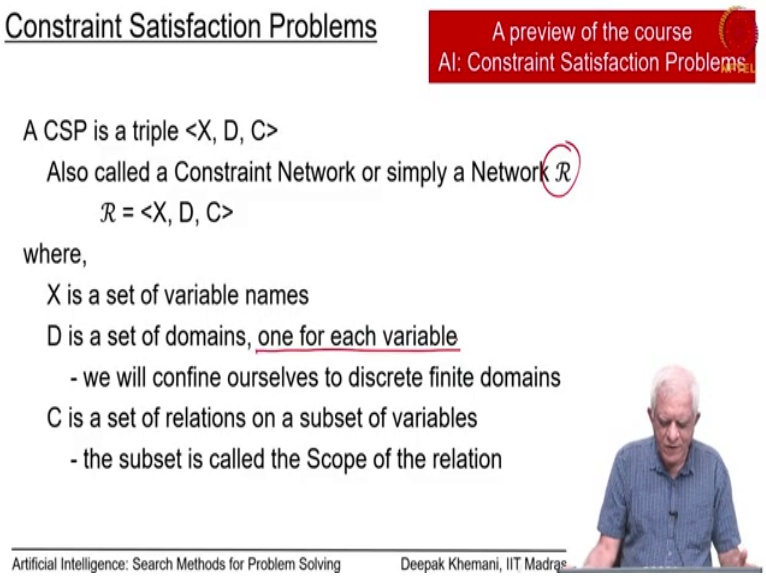
you see this number 2 here that number 2 is the aridity of the relation. Basically, in the first two examples we did not mention the aridity explicitly and we kind of said that it is a binary relation.

But, this 2 basically says it is a binary relation, it is between 2 elements and this Aunt relation tells you who is the Aunt of who. So, what I have highlighted in red is the name of the Aunt and there are these 3 people whose Aunt she is essentially. Likewise I have said here that arun is the brother of amy, anil is the brother of arun and arun is the brother of anil essentially right.

We are talking about the small family here, but you will notice that we had to do this explicitly. We have to say that arun is the brother of anil and anil is the brother of arun essentially.

If you have not done this and we would have had to specify additional properties on relations. For example, that it is symmetric and so on and so forth. In which case if you are using logic and we have already seen at least a glimpse of the relations form the basis of the predicate logic because all these can be predicates. Arun sorry Aunt or Brother or ThreeSiblings and 2 or 3 elements can be the arguments that essentially.

(Refer Slide Time: 10:17)



So, that is a very quick review of relations. Now, let us move on to our main course as we might say, which is constraint satisfaction problem. So, first we will define constraint satisfaction problems then we will look at some examples of constraint satisfaction problems, how we can express problems as constraint satisfaction problems.

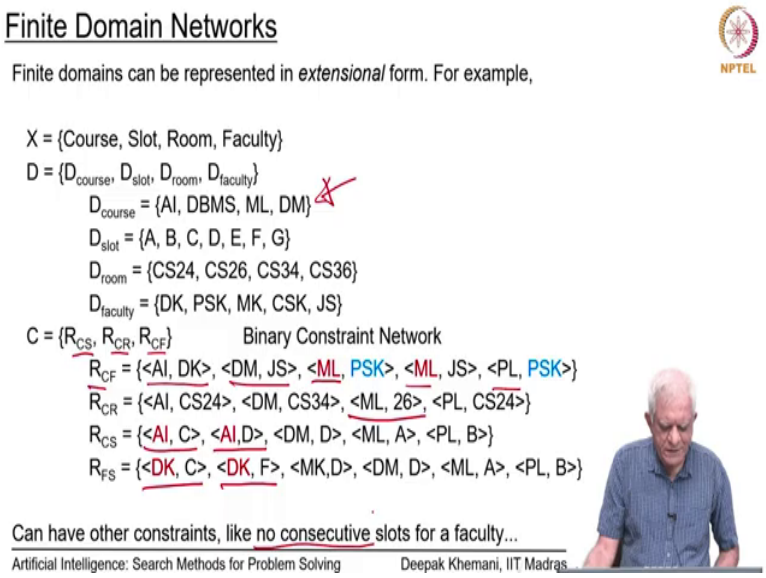
And then eventually we will have a sneak preview of how to solve constraint satisfaction problem. So, the idea is going to be again similar that posing the problem is a CSP is one thing, but solving a CSP is a completely independent thing and which is not dependent on which problem you are working on.

So, CSP is a working on. So, CSP is a triple, it is made up of 3 sets. The 3 sets are X and D and C. It is also called a constraint network or simply a network. We often use a symbol R to do that and many much of literature will use this notation.

So, we write that the network is made up of this triple X, D and C, where X is a set of variable names. D is a set of domain, one for each variable. Be careful about that. So, we saw courses had a separate domain and rooms had a separate domain and like that and C is a set of relations on a subset of the variables.

So, each relation is a relation on the subset of the variables and that subset for the particular relation is called the scope of the relation. So, like I did in the last week also let me make a little bit of a pitch inviting you to come and do the course on constraint satisfaction problems, it is a shorter course on eight weeks and it will be offered next semester.

(Refer Slide Time: 12:09)



So, we said that we will confine ourselves to extension form of relations and that kind of restricts us to working on what are called is fine finite domain networks. So, the domains are finite and therefore, the relations can be expressed in an explicit form essentially in an extensional form.

So, for example, if you are talking about teaching scenario the variables can be course, slot, room and faculty essentially. So, these are the 4 variables and the domains of each of them would be respectively the domain of course, is the set of courses as you can see here, the domain of slots is the set of slots.

So, for example, this slots may be named as A, B, C, D, E, F, G slots and rooms are set of rooms and faculty is a set of faculty. So, these are the 4 domains and we are talking about

relations over these 4 domains. So, typically if you are kind of helping your department; do course alocation scheduling you may have to worry about these kind of things.

Here we are talking about a binary constraint network and the network is got 3 relations in it. The first relation; there 3 are relations one is between courses and slots, the second is between courses and rooms and the third is between courses and faculty.

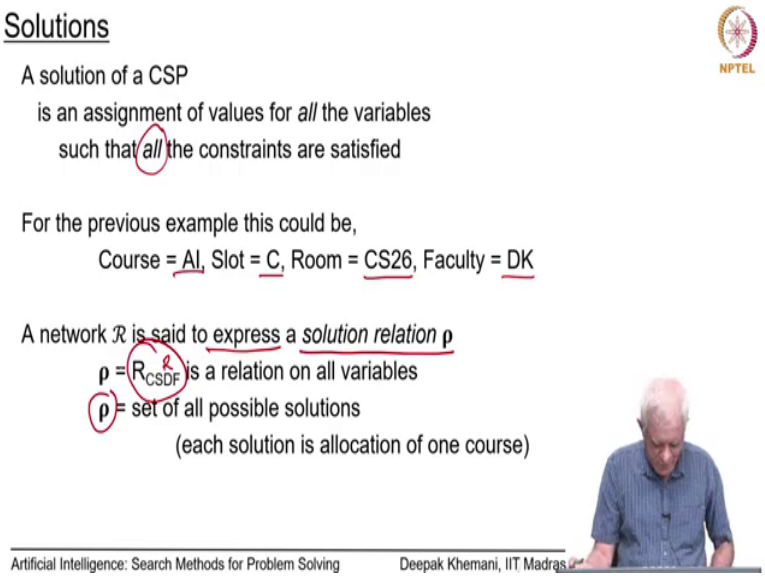
So, if we look at the relation courses in faculty then it basically lists out who can do what course essentially. So, for example, here I have said that DK can do AI and JS can do discrete maths. If you want to call DM discrete math, so, it could be data mining otherwise. Anyways you have to be careful that your name course is uniquely and so on. You can see that this is just this in some sense the sea of possibilities.

So, for example, the Course ML can be taught by two people here. One is PSK and one is JS. Likewise, PSK can teach ML, but can also teach programming languages. So, these are just the possibilities that can happen and what we are saying here by a relation is that that these are the allowed combinations essentially. Likewise, we can have relation between courses in rooms that is which some course can be in some rooms.

So, for example, ML can be in 26 for some reason the parliament may introduce this kind of constraints. And in terms of Slots so, for example, you might say that AI can be in C Slot or in D slot and likewise for other courses and faculty and Slot preferences. So, for example, DK may have a preference on C slot and F Slot and other people may have preference on other course.

So, these are in some sense the kind of inputs that you gave to constraint satisfaction problem and you can have other constraints. We will briefly talk about that shortly. For example, that a faculty member does not plot any consecutive slots that you do not want to teach head to head in 2 hours essentially. You want to take a break and collect your thoughts and you know come back for the next course and so on. So, that is a example of a network.

(Refer Slide Time: 16:03)



Solutions are that for each variable you have to find a value. So, a solution of a CSP is an assignment of a value to every variable or assignment of values to all the variables. So, the variables get values from their respective domains essentially and the value should be such that all the constraints should be satisfied. If the, if you do that, then you have a solution to the problem.

So, in this small example that we saw we could have these values. The course gets the value AI, slot get C, room gets 26, faculty DK. So, this is a solution, but notice that this is a solution only tells you about one course here essentially. So, that is how we have pose the problem the that that the variables are courses, slots, rooms and faculty and find us a solution.

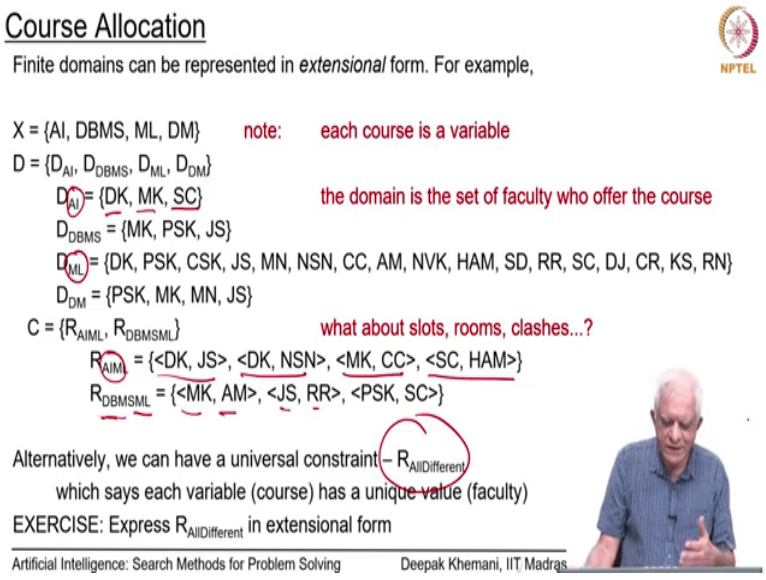
Maybe you can extend this to say find us all solutions, but then you know you have to worry about stating additional constraints. For example, the same instructor cannot be teaching the

same in the same slot two different courses. So, all those things will come into play, but this is just a small illustration, so, just to get us going.

We say that a network R is said to express a solution relation. So, it expresses a relation called solution relation. In our case there are 4 variables and therefore, the solution relation row as we call it is on those 4 variables; courses, slots, courses, slots, rooms and faculty.

So, this should have been Room here, CSRF essentially and this solution relation row is the set of all possible solutions essentially hm. In this example the way we have posed that the solution is just one course as we have seen here, but we can pose a problem differently to cater to more to more courses essentially.

(Refer Slide Time: 18:16)



So, let us try that a little bit. If you are worried about course a location then you might do the following that you can say that the set of variables is the set of courses. So, this is very different from what we said earlier. We said the variables were 4 things; what are the courses, what are the rooms, what are the faculty and what are the slots. Now, we are saying the variables are the set of courses only.

So, clearly you can see that this is more amenable to saying that allocate all these 4 courses to 4 faculty members. So obviously, the values must be faculty members essentially.

So, each course has a domain which specifies as to who can teach that course. So, for example, AI and MK and SC can teach AI, 3 people can teach DBMS and as you can expect in these times of COVID and ML, a large number of faculty members are interested in teaching machine learning.

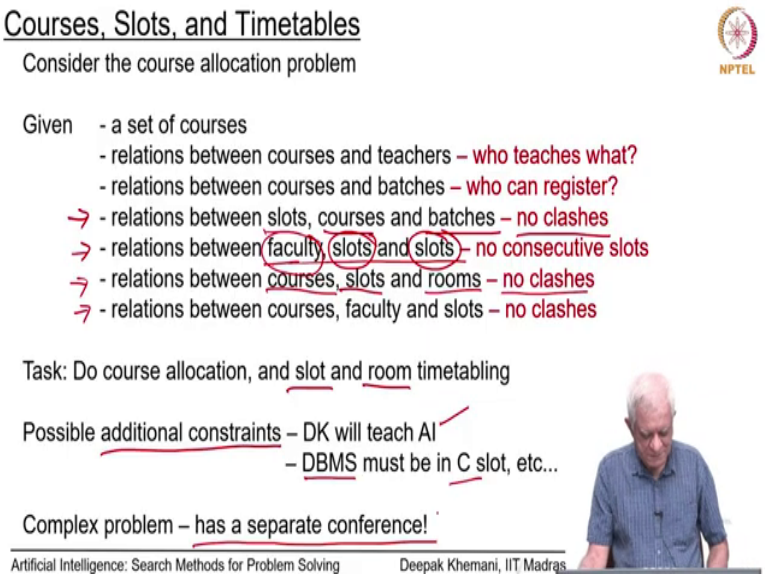
So, anyway, so that is what we have. We have a set of courses and each course has a domain which specifies as to which faculty can teach that course and the task is to now find an assignment to courses essentially. We can have domains, we can have constraints for example, here we have binary constraints and it says that the combination of AI and ML can be thought by these two people or these two people or these two people or these two people any of this combination is allowed essentially. This is again just to illustrate the problem.

In the real world you will have to do something little bit more sophisticated essentially and here I simply said that the relation between DBMS and ML is the to faculty who can teach that course respectively essentially. One constraint which is commonly used in specifying CSPs is called the all different constraints which says that every variable must have a distinct value essentially. So, the values must be all different.

So, you cannot say for example, that the same person will teach all 4 courses because that is not the kind of thing that you are looking for. Then you would use something like this and as a small exercise I will ask you to take a small example and express the all different relation in

extensional form essentially. How would you express that? We will see variations of this as we go along essentially.

(Refer Slide Time: 21:02)



So, now let us get to the real problem. It is not just faculty members you have worried about, it is also slots and timetables and not least of all the students essentially. So, the way you might really pose this problem is that given a set of courses in which there are relations which determine who teaches what, who can register which batches of students can register, what is the relation between slots and courses and batches. So, you cannot put the same batch for 2 courses in the same slot. So, there should be no clashes, clashes is a major problem when you are doing a location and timetabling

You may have relations like as I said earlier that faculties between this thing. So, notice that this is a relation between 3 elements. So, this, so, some of these; for example, this one this

one this one and this one these are all ternary relations which are relations over 3 elements. So, here the relation is between faculty, slots and again slots.

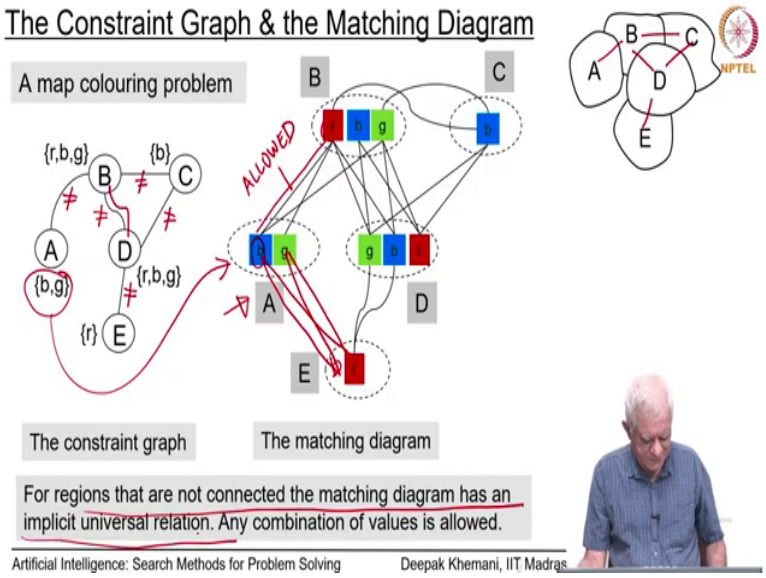
So, essentially we are saying that for example, faculty MK can teach in C slot and F slot. So, you specify 3 thing and you may specify other combinations for MK. So, any of those would be acceptable essentially. Then again you have to want to avoid clashes between course has slots and rooms.

You cannot have the course courses in the same slot in the same room and so on essentially; courses, faculty and slots and then you do course allocation. And you decide who which faculty will teach the course, which slot will the slot p in and which room will the course we essentially that is the problem of timetabling. It is not a easy problem. And lot of people will break their heads over trying to solve this.

In addition to finding any solution one often specifies certain additional constraints. So, I have mentioned this as additional constraints here. So, there is a core constraint satisfaction problem and then there is a specific instance of that in which you specify additional constraint. So, for example, you might say that this time when you are allocating DK will teach AI and DBMS must be in the C slot, maybe because of some reason and then these become additional constraints to which the solution is sort essentially.

Now, as I said course allocation and timetabling is not an easy problem and one often runs into all kinds of clashes due to which you have to iterate and so much so, that this whole domain has a separate conference. So, if you just look up on this on the way you will see that there is a conference on timetabling and scheduling of courses essentially, ok.

(Refer Slide Time: 24:01)



So, we have kind of informally described constraints then constraint satisfaction problems then the formal definition in terms of a network. And now let us talk of a graphical representation about constraint satisfaction problems and to do that let us look at a very popular problem for the map coloring problem. And I am sure you are familiar with this. So, there was this 4 color theorem that we mentioned at some point.

We said that any planar map or any map on a planar surface can be colored in 4 colors and this is a small example of a map. There are 4 regions as you can see A, B, C, D, E. Now, when you are specifying the constraint satisfaction problems you would also specify as a which colors are allowed. The 4 colors theorem simply says that 4 colors let us say red, blue, green and yellow are available for all countries.

But for some reason some regions may have their own preferences and in line with our notion of constraint satisfaction problems each of the countries will be a variable and the domain of the variable will be the set of colors that country is willing to use or region if you want to see. And the problem is to assign a color to each region from its domain such that no 2 regions no 2 adjacent regions have the same color.

So, A and B cannot have the same color, B and D cannot have the same color, D and E cannot have the same color, B and C cannot have the same color C and. So, this nationality lends us to a representation which computer scientists really like that you can say that these are the relations between these different colors that you cannot color them in the same color.

So, we often represent this as a constraint graph. As you can see the figure on the right I have converted into a graph where again there are the same 5 nodes; A, B, C, D, E and there are each region has its own domain. So, for example, B has RBG. A has only B and G and the relation which I have not specified, but I have just mentioned it verbally is the not equal to relation. It says that the values of these variables cannot be the same.

Notice that this is different from all different. It simply says that A and B cannot have the same value B and D cannot have the same value B and G cannot have the same value and so on. So, whichever two nodes are connected by an edge those two nodes cannot have the same value. So, this representation is called a constraint graph.

And, it is commonly used to talk about constraint satisfaction problems and you know the algorithms that people talk about an analysis of those algorithms and so on. The graph is always a graph in which there is an edge between 2 nodes, the simplest kind of graph. So, this is not a hyper graph that is what I mean to say. The relations does not do not have to be binary. Relations may be for example, ternary or even on more elements.

The graph simply says that B for example, is related to D. So, it could be that there is a relation between B, D and C which is over 3 or ternary relation, but the graph does not express as a graph simply says what is related to what essentially in the constraints that are

specified. So, this is the representation of the constraint the expressed constraint satisfaction problem.

Another common representation we use is the matching diagram. And in the matching diagram for each region or each variable we have a set and the set contains the domains of the variable. So, for example, A has domain B and G and that is expressed here into this set as B and G essentially. So, likewise for every other domain and edge in the matching diagram says that this is allowed.

So, here we are saying for example, that blue for A is allowed with red for B because as an edge between B and R essentially. So, the matching diagram explicitly tells you what combinations of values you can assign to the different variables. It is a very interesting diagram and it is also used effectively to analyze algorithms. So, we often you know move between the two representation, the constraint graph and the matching diagram.

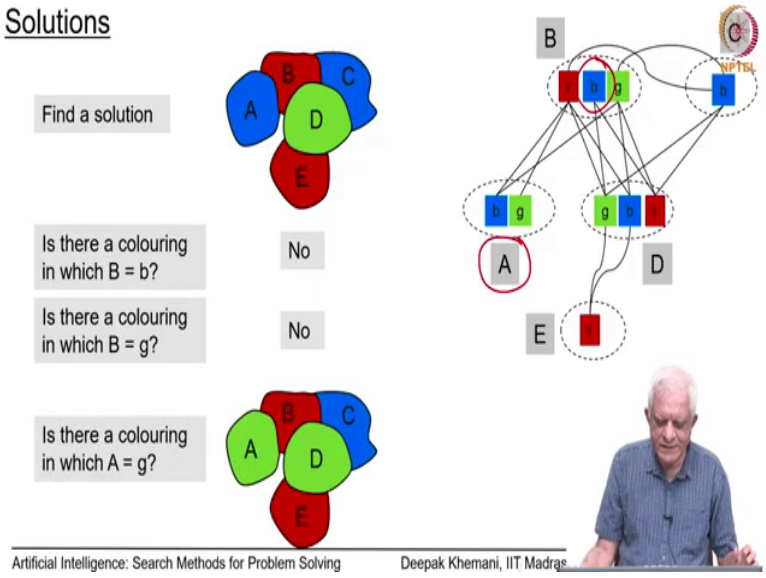
The constraint graph is a graph which tells you what combinations are allowed what variables are related to what variables and the matching diagram tells you as to which values are allowed which combination of values are allowed essentially. Now, there is one thing which is kind of implicit here and it is that if there is no edge between 2 variables in the constraint graph, it basically means that they do not constraint each other.

In the matching diagram that translates to saying that any combination of values is allowed. So, for reasons that are not connected the matching diagram has a implicit universal relation. So, look at A and E for example, not having an edge between A and E in the constraint graph is equivalent to having these both edges in the matching diagram. It says that you can use these two combinations and there is no constraint between them.

Even if we had let us say blue as blue in the domain of E, we could have also connected blue to blue there and green to blue here and the reason for that is that you can color both A and E in the same color, there is no constraint that you should not.

So, implicitly there is a universal relation and any combination of values is allowed essentially. The reason we mention this is that the first approach that we will see is going to be a search approach as you can imagine is that you try out different values for variables till you found the solution I think.

(Refer Slide Time: 31:10)



So, we are talking about solutions now. So, what are solutions? Solutions are as we said a value to each variable. So, you let us say use will choose a value for A then you will maybe this business of which order to do is itself an interesting problem. Unfortunately, we do not have time to go into that, but there is a lot of study which people have done on the properties of graphs which help you decide which value to choose first, also on the domains of the variables.

So, for example, as you can see here E has only one value in the domain. So, that is the only choice you have. So, you might as well look a look at it first. So, if you are choosing that that strategy or that heuristic which is called dynamic variable ordering then you would choose the variables with the smallest domains first.

In this case you would choose E then you choose C and then you would you know choose the others. We will see a little bit about how choice also metals a little bit, but in general what you expect is that you state a constraint satisfaction problem which is triple of a set of variables domains for each variables and constraints on each variables which can be represented either by constraint graph or a constraint or more explicitly in matching diagram as we have done here and we say give us a solution essentially.

So, if you have a solvers then that is what you would say. Simply you might say find a solution and solve our would find a solution and say that oh ok, here is a coloring of the graph. You might specify that certain regions might have certain colors.

So, you might say is there a coloring in which the region B has color blue and it tries its best and says, no, that you cannot color it blue even though blue is in the domain of B. Then you say ok, is there a coloring in which B has a value green? It comes back and says no again essentially. If you ask is it a coloring in which A has a value green?

It says yes, I can give you a solution to that essentially. So, that is what you expect when you are working with constraint satisfaction problems that you pose a problem as a constraint satisfaction problem. You call some solver to solve the problem for you and you may have specify additional constraints like we have done here in the bottom those 3 cases and this system will look for a solution and give you one.

So, let us take a tiny break and we will come back and look at some more examples.