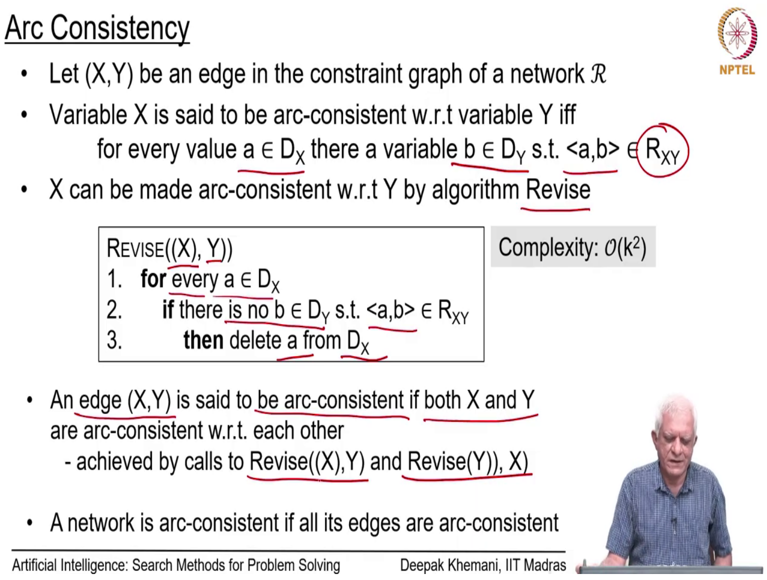
Artificial Intelligence: Search Methods for Problem Solving

Prof. Deepak Khemani
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Chapter – 09
A first Course in Artificial Intelligence
Lecture – 93
Constraint Processing Arc Consistency

(Refer Slide Time: 00:14)



Arc Consistency

- Let (X,Y) be an edge in the constraint graph of a network $\mathcal{R}$
- Variable X is said to be arc-consistent w.r.t variable Y iff
  for every value $a \in D_X$ there a variable $b \in D_Y$ s.t. $<a,b> \in R_{XY}$
- X can be made arc-consistent w.r.t Y by algorithm Revise

```
REVISE((X), Y))
1. for every $a \in D_X$
2.     if there is no $b \in D_Y$ s.t. $<a,b> \in R_{XY}$
3.         then delete a from $D_X$
```

Complexity: $\mathcal{O}(k^2)$

- An edge (X,Y) is said to be arc-consistent if both X and Y are arc-consistent w.r.t. each other
  - achieved by calls to Revise((X),Y) and Revise(Y)), X)
- A network is arc-consistent if all its edges are arc-consistent

Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Madras

So, welcome back. We just looked at the algorithm backtracking, and we observed that it suffers from the problem of running into combinatorial explosion like, blind search that we have seen earlier does. And we have decided to look at a couple of ways of elevating that problem. And the first thing we want to look at is consistency enforcement.

So, we will look at a basic simple notion of consistency to start with and we will, then talk about higher orders of consistency. The simplest kind of consistency is called arc consistency and the idea is as follows, remember that we have a constraint graph which we are trying to solve. So, let X Y be an edge in the constraint graph of a network R essentially.

We say that a variable X is said to be arc consistent with respect to a variable Y. So, X and Y is an edge remember, if for every value a of a in D of X D of X is a domain of X, there is a value b in D of Y such that this pair or this tuple belongs to the relation between the two variables X and Y. Remember that constraint graph, represents relations essentially in edge says that X is related to Y.

So, if for every value of a in D of X, there is a value of b in D of Y it can be the same value, it does not matter all we are saying is that from that value a there must be a corresponding value b in the other domain. Then we say that the variable X is arc consistent with respect to Y, X can be made arc consistent with respect to Y by a simple algorithm which is traditionally called revise.

So, what are you doing here? You are revising the domain of X with respect to the domain of Y essentially. So, this notation kind of emphasizes the fact that X is being revised, with respect to the variable Y essentially. And the algorithm is what we just said above that for every a belonging to D of X, if there is no b belong to D of Y in the domain of Y. Then you delete this a from D of X essentially.

So, you look at all the values of a look at all the values of X and for every value a in the domain of X. If you cannot find a corresponding value in Y, then you delete that a from the domain. So, we can see that we are basically pruning the domain essentially and we have pruning the domain of variable X.

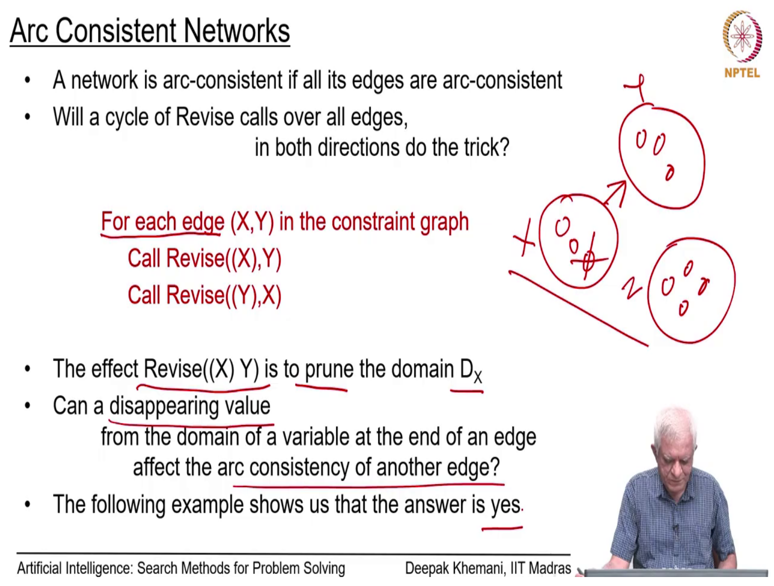
So, that is revise an edge X Y is said to be arc consistent. So, we are talking now about an edge being arc consistent. If both X and Y are arc consistent with respect to each other. And; obviously, we can achieve this by calls to revise X Y and calls to revise Y X.

So, X Y will delete values from the domain of X, Y X will delete values from the domain of Y essentially so, there is an edge. A network is said to be arc consistent if all its edges are arc consistent essentially. So, now, we have a recipe for pruning the domains by looking at a particular kind of consistency called arc consistency.

And arc consistency says, that if you take any arc or any edge in the constraint graph. And you take any value from one end you will find a corresponding value in the other end essentially.

So, you can take a value in this X and you will find a value in Y. So, we say X is arc consistent with respect to Y or you can take a value from Y and you will find a value in X, then we say Y is arc consistent to X. And then if both are the case, then we say that the edge is arc consistent and if all edges are arc consistent then the network is arc consistent.

(Refer Slide Time: 04:48)



Arc Consistent Networks

- A network is arc-consistent if all its edges are arc-consistent
- Will a cycle of Revise calls over all edges, in both directions do the trick?

```
For each edge (X,Y) in the constraint graph
    Call Revise((X),Y)
    Call Revise((Y),X)
```

- The effect Revise((X) Y) is to prune the domain D_X
- Can a disappearing value from the domain of a variable at the end of an edge affect the arc consistency of another edge?
- The following example shows us that the answer is yes.

Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Madras

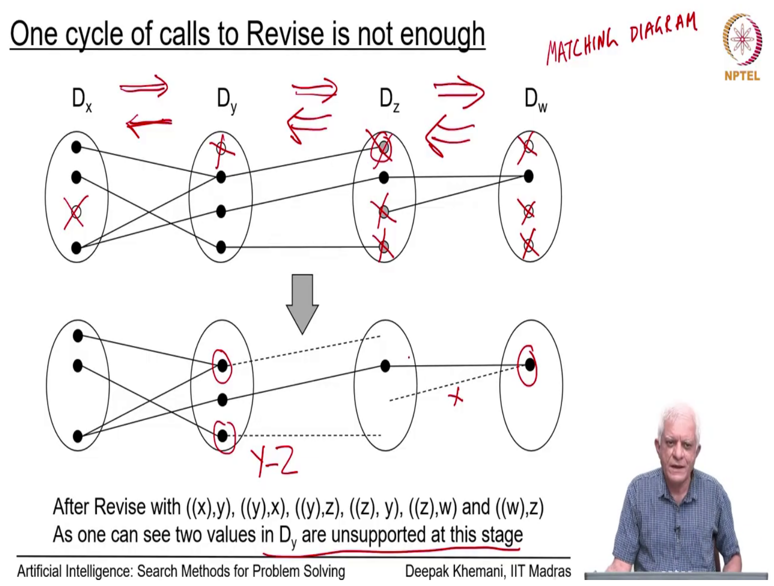
So, let us look at an algorithm for making a network arc consistent. So, we just said that a network is arc consistent, if all its edges are arc consistent. So, will a cycle of a revise calls over all edges in both directions do the trick. So, can we do the following and that is a algorithm that I have written here or at least the snippet of an algorithm. It says that for each edge X Y in the constraint graph called revise X with respect to Y and call revise Y with respect to X.

So, clearly you are addressing all edges, because we are seeing for each edge and we are looking at it in both directions. I would like you to pause here and try to imagine this algorithm, maybe write a small constrained graph may be use a map coloring example or something. And see whether this algorithm will work or not essentially. The question that I would like you to ask is the following that we know that the effect of revise X Y is to prune the domain of X.

The question is that, if a value disappears from the domain of X, will it affect the arc consistency of some other edge essentially. So, what is happening? Here we have X it has some values we have Y, it may have some values we may have other variables like Z it may have some values.

And if we remove a value from the domain of X, when we are doing arc consistency with respect to Y. Will it affect the consistency of X and z? That is the question we are asking. And the answer is yes I hope you came to the answer, but let us see why that is the case.

(Refer Slide Time: 07:10)



One cycle of calls to Revise is not enough

After Revise with ((x),y), ((y),x), ((y),z), ((z), y), ((z),w) and ((w),z)
As one can see two values in D_y are unsupported at this stage

Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Madras

So, let us look at a constraint graph, we have spoken about the matching diagram earlier right. And so, this is a matching diagram for the constraint graph and we have four variables x y z and w x is related to y y is related to z and z is related to w. And these are the values each domain has four values in this example. And each edge is a edge in the matching diagram. So, this is the matching diagram that we have seen earlier. So, an edge says that these two values are related essentially.

So, the edges which are filled the nodes which are in black, they are arc consistent with respect to the other nodes and that is why we have colored them black. The nodes which are unfilled which are blank or white whatever you want to say are nodes which are not arc consistent essentially and so, they clearly will need to be removed essentially. The nodes which are grey for example, this node in the domain of z is arc consistent with respect to y, but it is not arc consistent with respect to w essentially.

So, eventually of course, it will have to be removed that is why. So, supposing this is the four nodes given to us, and this we said that for every edge in the constraint graph do revised in both directions. So, supposing we do it in this order, we first do revise in this direction, then we do revise in this direction. So, we have done two then you do this one.

So, what happens when we do the first one? When we do the first one, then we end up removing this from the domain, when you do the second one we end up removing this from the domain, when we do the third one y with respect to z, we do not do anything because every value of y has the corresponding value in z. When we do the reverse, we find that all that there is one value which is not does not have a value in y. So, we delete that value from that essentially.

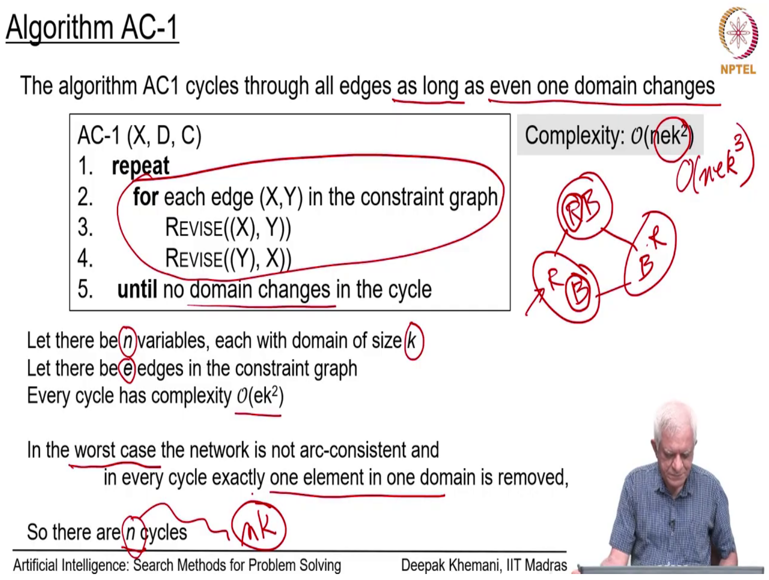
Then, when we do this one we will be forced to delete this because, it does not have a value in w and also we will be forced to delete this one, because it does not have a value in w essentially. And finally, when we do in this direction we these things anyway was not arc consistent. So, these three we will delete. So, if you do this whole thing in this direction, then this is what will happen.

We will get network which looks like this. So, the values that have been deleted on the top have vanished from the bottom part of the graph. And you can see that there are two nodes in the domain of y, which are not arc consistent with respect to z.

So, which I have written as there are two values in y which are unsupported at any at this stage actually. So, clearly this one cycle of doing revise x with respect to y, y with respect to x, then y with respect to z and z with respect to y. And then z with respect to w and w with respect to z will not work, because at the end of it the network is not arc consistent.

Why is it not arc consistent? Because this edge Y Z is not arc consistent and this edge is not arc consistent. Because, there are values in Y which do not have corresponding values in Z, observe that this edge which is also unsupported does not matter because, this value has a different support essentially. So, the only inconsistency is not inconsistency the only arc inconsistency is in the variable Y.

(Refer Slide Time: 11:39)



Algorithm AC-1

The algorithm AC1 cycles through all edges as long as even one domain changes

```
AC-1 (X, D, C)
1.  repeat
2.      for each edge (X,Y) in the constraint graph
3.          REVISE((X), Y))
4.          REVISE((Y), X))
5.  until no domain changes in the cycle
```

Complexity: $O(nek^2)$

Let there be n variables, each with domain of size k
Let there be e edges in the constraint graph
Every cycle has complexity $O(ek^2)$

In the worst case the network is not arc-consistent and
in every cycle exactly one element in one domain is removed,

So there are n cycles

Artificial Intelligence: Search Methods for Problem Solving — Deepak Khemani, IIT Madras

So, we have to this simple algorithm of just cycling through all the edges once will not work. And you have to try something more exhaustive, and that is a well known algorithm called AC 1, which stands for arc consistency 1 and it works as follows.

It cycles through all the edges and it repeats the cycle as long as even one domain changes, even if one domain changes we go back and do all this cycle again. So, as you can see that it is clearly a brute force algorithm and we will see in a short while an improvement upon that.

But, that captures the notion of making a network arc consistent its a correct algorithm, it may not be an efficient algorithm will try and make it efficient as we go along. So, what does this algorithm do? It simply says what we said earlier, which was this that for each edge X Y in the constraint graph, call revise X Y and call revise Y X essentially. But, now we have put this in a loop in a peak loop, which says that until no domain changes in the cycle essentially.

So, you want to make sure that after every cycle of all the revise calls, there is every node has a supported value which means, some value has not disappeared from some domain essentially. So, what is the complexity of this very quickly? Let there be n variables and each which the domain of size k.

So, the variables are n the sizes are k and let there be e edges. So, in every cycle we will do it order ek square times k square, because look at all the values of Y. So, that is where the k square comes from essentially.

So, in the first if you are looking at the first value of X, you look at all the values of Y, you do not get a value then you look at the second value of X look at all the values of Y. So, you will do k this times and for every k you will k here so, its k square and because there are e edges it is ek square. Now, in the worst case we are talking about the worst case of AC 1 the what is the best case? The best case of AC 1 would be when the network is itself arc consistent to start with essentially.

And so, here is an interesting example. Supposing we have a map coloring problem, we have three regions connected to each other and 2 colors red and blue. You can see that this network is arc consistent to start with itself, because you choose any value here. So, supposing you choose red here, you can choose blue in the other two. If you choose blue here, you can choose red in the other two and likewise for the other three variables.

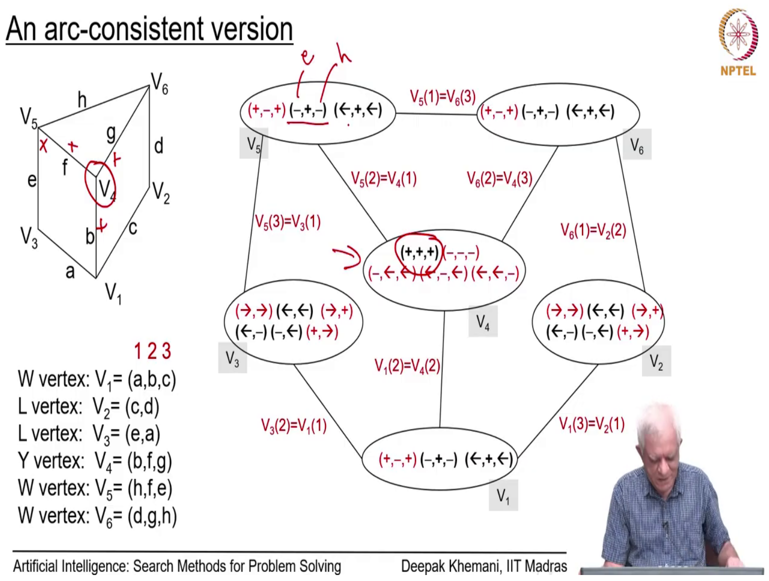
So, clearly by definition this is arc consistent, but if you observe this network closely you will see that it does not have a solution. In fact, even though it is arc consistent. So, constraint satisfaction is an interesting field, I think we are talking about consistency we are talking about arc consistency.

But, we will also talk about higher order of consistencies in the passing only in this course. And then we will say that you know the amount of consistency required depends upon certain things, which we will mention only in the passing essentially.

In this particular example the level of consistency required is one level higher which is called path consistency, which says that if you choose two values from 2 any 2 variables. So, supposing you choose B from here that is allowed, you choose red from here that is also allowed. You should be able to find the value in the third domain, but as you can see if you have chosen B and R, you cannot choose b you cannot choose R. So, this network is not path consistent.

So the best case is when the network is arc consistent to start with the worst case is, and if there are in every cycle if one element is removed, then you can see that there are n into k value. So, this should be actually order n e k cube, why because there are n cycles n sorry, they will go there are going to be n into k cycles. Because in every cycle one element will be removed so, that is why we get ne k cube here essentially. So, that is the algorithm AC 1.

(Refer Slide Time: 17:05)



An arc-consistent version

1 2 3

W vertex: V_1= (a,b,c)

L vertex: V_2= (c,d)

L vertex: V_3= (e,a)

Y vertex: V_4= (b,f,g)

W vertex: V_5= (h,f,e)

W vertex: V_6= (d,g,h)

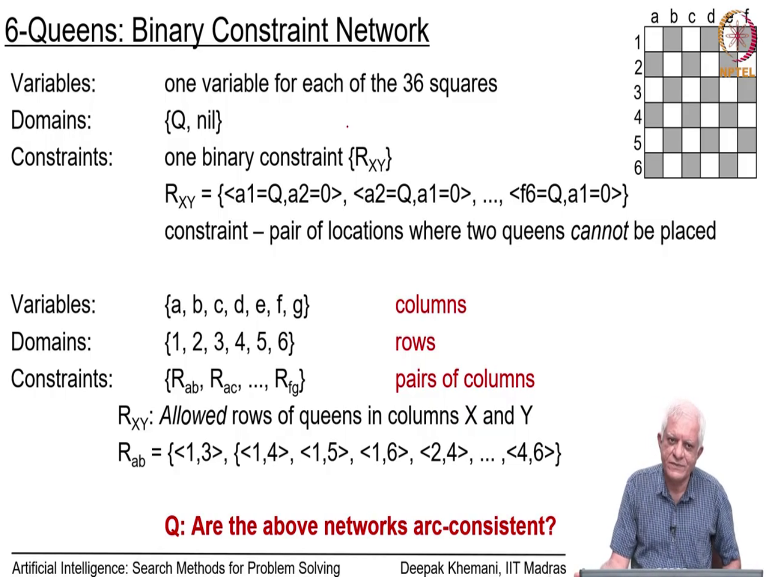
Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Madras

Here is an example that we had seen earlier of line labeling problem it sometimes called as Huffman Clowes labeling. And, we had seen a simple object and we had constructed the matching diagram from for that object. And I will leave it to you to verify that after you make this network arc consistent.

Some of the values which we have shown in red will vanish and only the values that we have shown in black will remain essentially. So, for example, for this vertex which is vertex V 4, we can only say that all the three values are positive. And there is no other consistent value of assignment. Whereas, if you look at V 5, we have given two options here that this of course, the middle one must be positive of course, but the other two which is e and h.

So, this is e and this is h they can either be arrows which means that there is these kind of objects hanging in the air, or they can be minus which means that they are placed against a wall or they are part of a larger object, which we cannot see in this figure essentially. But, all these arc consistent assignments and you should try out the algorithm of arc consistency and see that it is indeed the case.

(Refer Slide Time: 18:41)



6-Queens: Binary Constraint Network

Variables: one variable for each of the 36 squares

Domains: {Q, nil}

Constraints: one binary constraint {R_{XY}}

R_{XY} = {<a1=Q,a2=0>, <a2=Q,a1=0>, ..., <f6=Q,a1=0>}

constraint – pair of locations where two queens *cannot* be placed

Variables:	{a, b, c, d, e, f, g}	columns
Domains:	{1, 2, 3, 4, 5, 6}	rows
Constraints:	{R_{ab}, R_{ac}, ..., R_{fg}}	pairs of columns

R_{XY}: *Allowed* rows of queens in columns X and Y

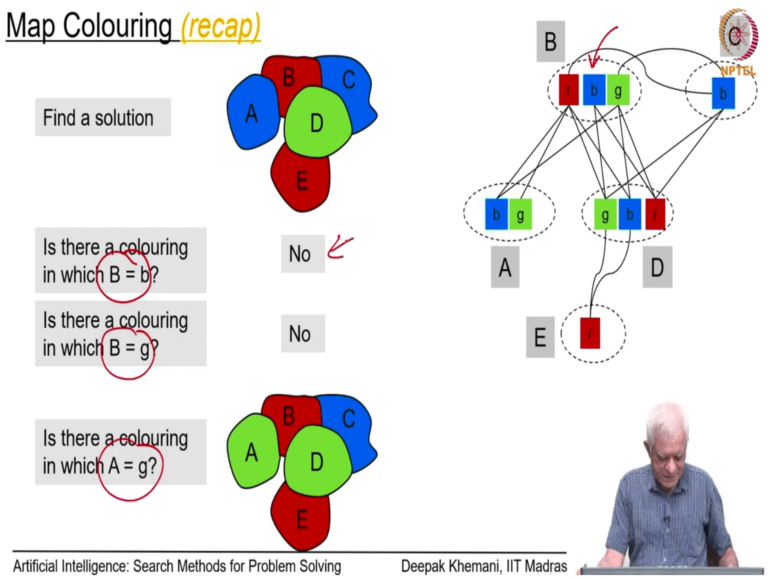
R_{ab} = {<1,3>, {<1,4>, <1,5>, <1,6>, <2,4>, ... ,<4,6>}

Q: Are the above networks arc-consistent?

Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Madras

We had also looked at the n queens problem in particular the 6 queens problem. And we had given these two formulations of the n queen problem or the 6 queen problem. And I will leave it to you to look at these networks and answer whether they are arc consistent. And if they are not make them arc consistent. So, that I believe it is a small exercise for you to do.

(Refer Slide Time: 19:12)



Map Colouring *(recap)*

Find a solution

Is there a colouring in which B = b? No

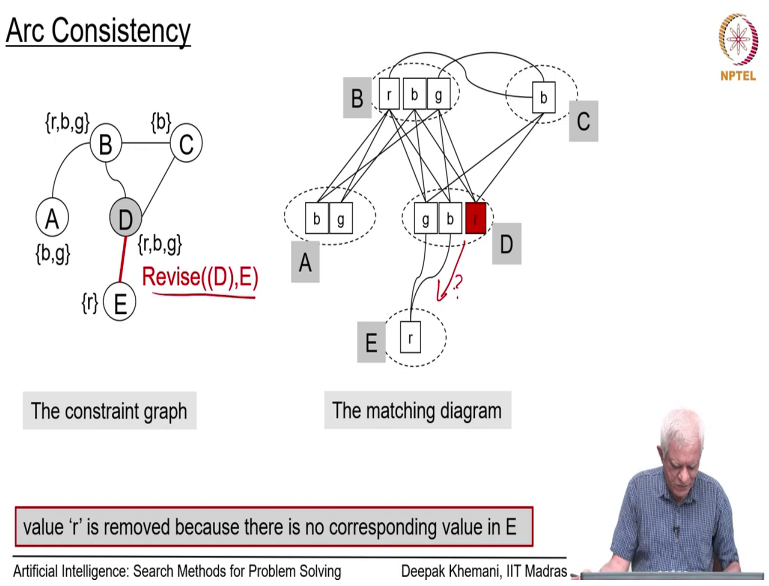
Is there a colouring in which B = g? No

Is there a colouring in which A = g?

Look at this map problem that we had seen earlier, we had asked various questions that you know give us a solution or give us the solution where for example, region b has a value blue. And the system had come back and said no and so on and so forth. So, this is what we had done earlier, you asked for a solution, you get a solution you ask for a specific solution with B equal to b this is the algorithm should come back and say no.

You asked for a solution will B equal to g no or you asked for a solution with A is equal to g and you get a solution. Now, when it answers this no remember that it would have searched the entire tree. And then, eventually when the search failed, then it would have come back and said no essentially. Let us see what happens when we make this network arc consistent.

(Refer Slide Time: 20:16)

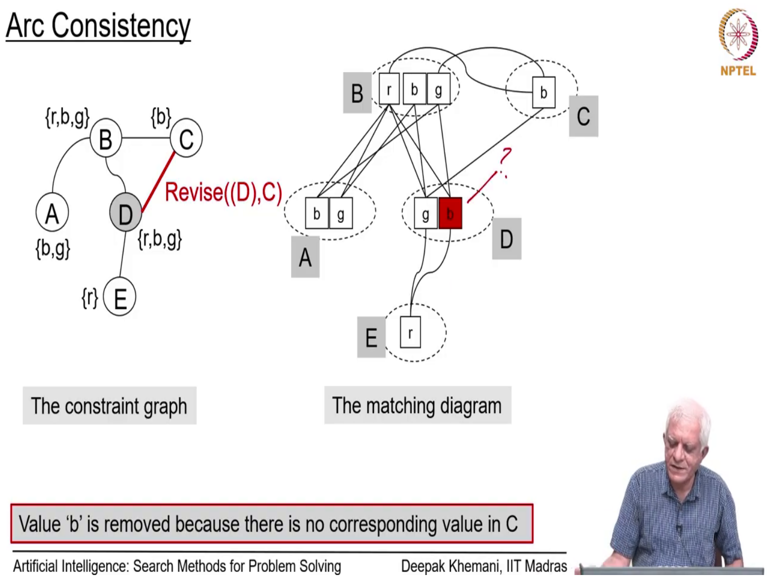


Arc Consistency

value 'r' is removed because there is no corresponding value in E

So, we will process the edges in some order. So, first the diagram here that we have going to be drawing is the grey nodes are the ones which are going to be processed. And the red edges are the tell you as to with respect to which other node you are processing. So, first we revise D with respect to E and you see that in the matching diagram, there is a value r in D shown in red, which does not have a corresponding value in the domain of E. So, we so revise will remove that value r from there essentially.

(Refer Slide Time: 21:00)



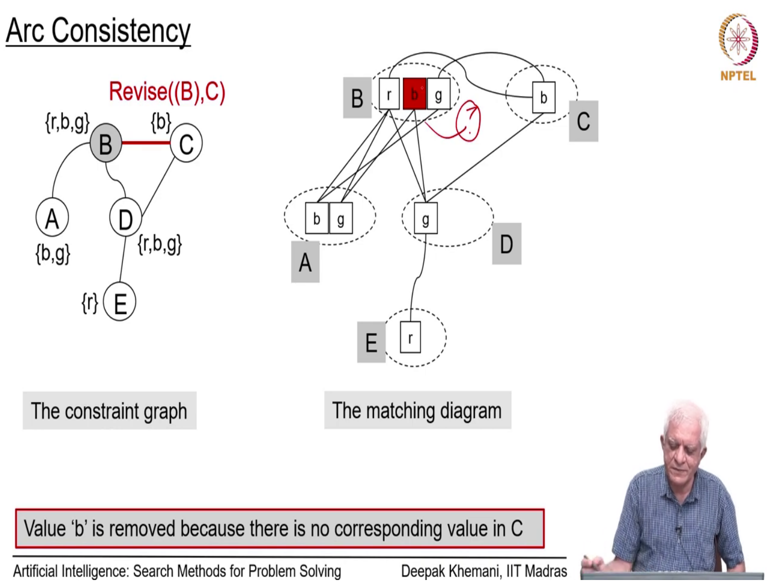
Arc Consistency

Value 'b' is removed because there is no corresponding value in C

Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Madras

Then, we revised D with respect to C and you can see that this value b does not have a value b in the domain of C.

(Refer Slide Time: 21:10)



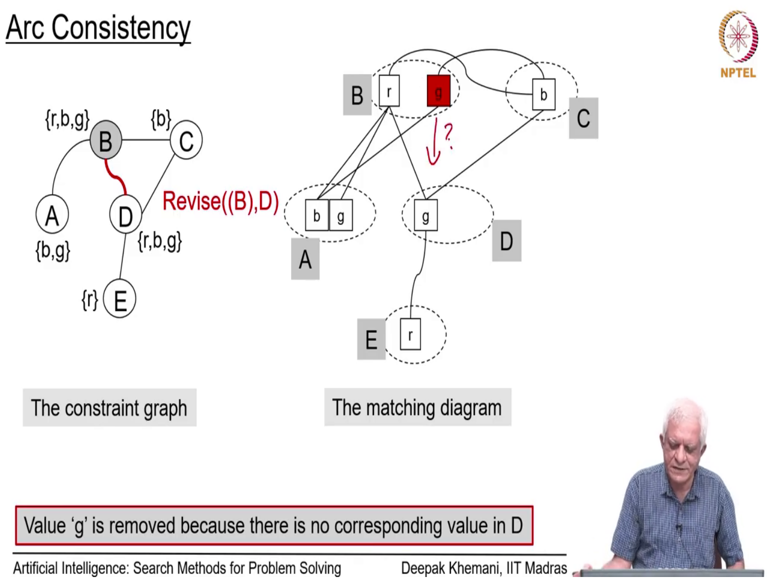
Arc Consistency

Value 'b' is removed because there is no corresponding value in C

Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Madras

So, b will go from this domain, then you revise B with respect to C and you can see that this value b does not have a corresponding value in C. Remember that the value must be different essentially. So, if you are going to color region b as blue C must get some other color, but this problem says it does not have.

(Refer Slide Time: 21:37)



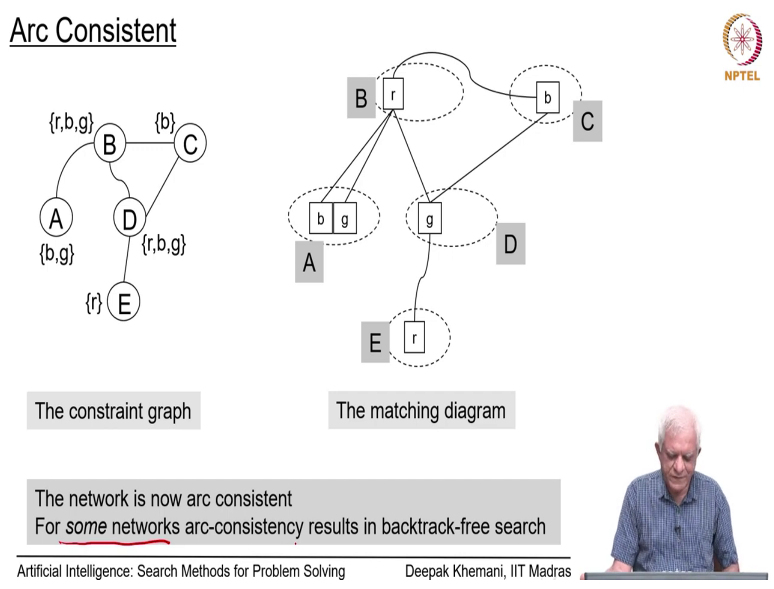
Arc Consistency

Value ‘g’ is removed because there is no corresponding value in D

Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Madras

So, you move B from there and when you revise g with respect to D again there same problem occurs.

(Refer Slide Time: 21:45)



Arc Consistent

The constraint graph

The matching diagram

The network is now arc consistent
For *some* networks arc-consistency results in backtrack-free search

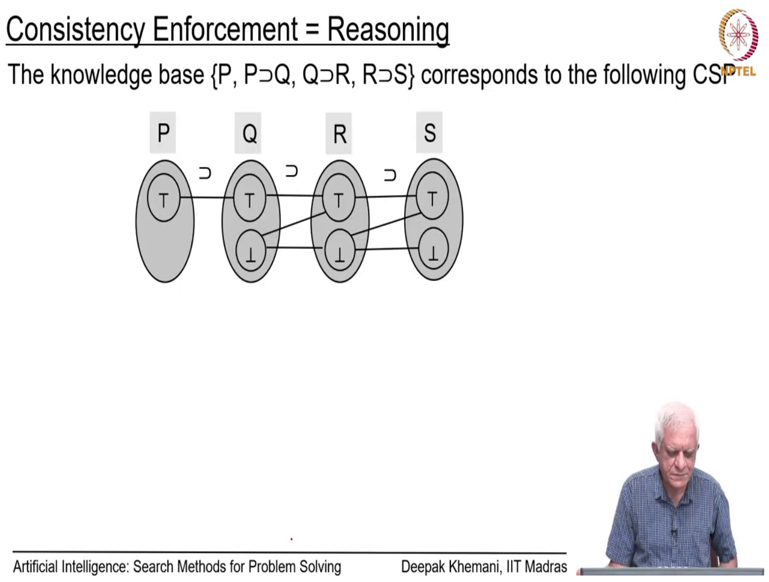
Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Madras

And you moved that essentially by the time you made the network arc consistent here, you can see that you can do the search algorithm in any order and you will reach a solution without back tracking essentially. For some problems, for some networks, arc consistency results in backtrack free search essentially.

We will not get into the formal discussion on how much consistency is enough to make the search backtrack free, but just remember that consistency reduces backtracking. And if you do enough consistency, then you may eliminate backtracking all together.

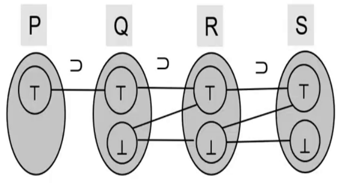
Now, arc consistency is enough if your network has a topology of a tree of course, this network that we have drawn is not a tree. But, for networks which are trees it can be shown that arc consistency is enough. And perhaps, you can take it up with a small exercise in verify for yourself if that is the case, but we will not go into all those details here.

(Refer Slide Time: 22:51)



Consistency Enforcement = Reasoning

The knowledge base {P, P⊃Q, Q⊃R, R⊃S} corresponds to the following CSP



Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Madras