Artificial Intelligence: Constraint Satisfaction Problems
Module 4: Directional Consistency
Lecture 4: Directional Path-Consistency and Directional i-Consistency
Professor: Deepak Khemani
Department of Computer Science and Engineering, IIT Madras

Keywords: directional path-consistency, directional i-consistency

Okay, so let's continue our study of directional local consistency. In the last class, we saw that trees which are essentially width one graph in the sense that you can always find a width one ordering for a tree can be solved effectively by doing directional arc consistency. You produce simply the minimum width ordering, which will also be the minimum induced width ordering and since each node would have exactly one parent, you do DAC on that ordering and you would have produced a graph which will have backtrack free search. If there are orderings which are more than width 2, then what do we do? The next step obviously is something which is when there are two parents, so let's look at that notion and the notion that we will need is directional path consistency. So let's as usual start with a definition, a network is, so this is, we will call it DPC, is DPC with respect to an ordering d if for every, it should be if and only if $k > i, j$, actually greater than is enough, the pairs $<x_i , x_j>$ is path consistent with respect to x_j. So this definition like before is similar to path consistency except that it is not enforced in all directions, but only from past variable to further variables. So past variables should be consistent with future variables. That is the general idea of directional consistency, and in this case we are talking about pairs of variables since we are talking about path consistency or three consistency. We want that every pair to which we assign a value, we should be able to find a value for a future pair essentially. Now we had seen when we were looking at path consistency algorithms that they introduce new edges essentially, so that was something that we had seen, change the constraint graph or may change, not necessarily changes and the specific change it makes is that it adds edges, and we will see that this leads us to the difficulty that you may start off with a certain ordering of a certain width and then suddenly you find that you are adding more edges so the width is gone up essentially. So that is the main difficulty we will find in higher order consistencies. We had seen earlier that enforcing i-consistency in general induces a relation of arity $i - 1$, and this is just a special case of that in the sense that inducing three consistency induces a relation of arity two, which is basically an edge in the constraint graph essentially. So that leads to certain complications, so if you look at an example that we had seen earlier where you have map coloring and these four countries are kind of connected to each other and there are two colors: red, blue, and the relation is not equal to essentially, and now let's impose an order on it. So A, B, C, D, so if I now do in this order, I will have A which is related to B which is related to C which is related to D, which in turn is related to A. So the first graph is an unordered graph and the second graph is a particular ordering essentially. Now if we do full PC, full path consistency adds, introduces two edges to the graph and these two edges are you can see like this and like this and the relation there is equality. We had seen this when we were looking at path consistency algorithm that if you enforce full path consistency then you will end up introducing these two edges essentially, but directional path consistency adds only one edge, and the edge is between A and C. So if you look at the first graph, you can see that the relation between A and C is equality. In the second graph also we are saying the same equality relation. The other relations, the solid relations are not equal to relations because it is the same graph that we have redrawn in a given order. We are saying the same thing that A and C must get the same value. It is just that we are not bothered about B and D, which was there if you did full arc consistency and the reason for that is now let's say that you are doing this map coloring, you start with A and you start giving it a color red, that’s fine and then because A is related to B, you have to choose a different color for B and the only different color that is available here, or let me write this here, you choose blue for B. Let me add one color to blue, let's say g is also allowed here, but then that thing will disturb the whole problem, so let's just leave it like this. In this example it turns out that the only choice you are left with for C is red because B and C are also related by the not equal to relation, but what path consistency is saying that you can choose a value for A and a value for C in such a way that I can choose a value for D. That is the basic idea of path consistency, which is why we are inducing a new relation here between A and C, which says that only certain combinations of A C values are allowed essentially. But the point about I am trying to make here is that directional path consistency will introduce only one edge as opposed to full path consistency would have introduced two edges and in that sense you are doing less work than the other algorithm.

So let's look at the algorithm DPC, so it takes a network $R=<X, D, C>$ and it has a constraint graph $<V, E>$. So the first thing you do is you make a copy of the set of edges that you have and then like before, so all this directional consistency we always, as you would have noticed, start from the last node and sort of proceed towards the earlier nodes, because you want to make earlier nodes consistent with later nodes essentially. So here again, we do something similar for A going from n down to let's say 2, you do two things. We had discussed this when we were talking about directional path consistency. We had said that just or when we were talking about path consistency that enforcing path consistency will not guarantee the fact that there is a solution at all essentially, and we had seen that you must enforce path consistency as well as arc consistency to guarantee that there is a solution. In other words, you must enforce strong path consistency. So in fact that is what we are doing here. We are talking about strong path consistency, which basically means that in this algorithm we will do path consistency as well as arc consistency, but both will be directional in nature. So let me use the notation used by Dechter here. So $\forall$ i, j < k, such that $x_i, x_k \in E'$ and $x_j, x_k \in E'$, we do Revise3, I will again use Dechter's notation here x_i x_j with respect to x_k, and then we add edges. So this we are familiar with that when we are enforcing path consistency, we end up adding edges to the constraint graph, and we are doing this explicitly here essentially. So this part is in some sense doing the DAC part or DPC part and then we do the DAC part, which is second part of algorithm, which says that for all $i < k$ such that $x_i, x_k \in E'$, you call Revise x_i with respect to x_k, and so that is an algorithm. So you are looking at a constraint graph and for every pair of edges or every pair of parents that the node has we are connecting them together and imposing a relation such that you know that revise relation will prune the edges or delete some tuples or in other words add a new edge. If the edge does not exist, if the edge exists of course this union operation will do nothing essentially. So let's first discuss the complexity of this. You can see that this will introduce a k^3 component if you remember Revise3 had a k^3 component and then this part will introduce n^2 component because we may end up looking at all n^2 variables and this part will introduce an n component. So between the three of them, we will have order n^3 k^3 essentially.

Okay, let me just quickly talk about the generalization to directional i-consistency, we will not write the algorithm in detail. I want to get back to that if you enforce DPC on certain problems, is it enough to guarantee backtrack search or not, but this generalization is something that is natural. So every $i - 1$ past, or I should say parents consistent with that variable, i^{th} variable. So this is the only little bit of thing that we have to worry about here is that if the number of parents is less than or equal to $i - 1$ then just call Revise-i which is an appropriate algorithm, which will induce a relation of arity $i - 1$, else and the else part says that the number of parents is more than what you are trying to, you are trying to do i-consistency and the number of parents is larger than that so what you do is that call Revise with each subset of P of size $i - 1$. So that is a small small difference in that, but this Revise will induce a relation of arity $i - 1$, and this we have seen before. The only thing here is directional essentially, that you are moving only only from the left hand side to right hand side, essentially. So we are not doing in multiple directions. So we will do only one pass as before, and adds many edges to the constraint graph. So you can just read this topic. It is just a straight forward generalization of moving from arc consistency to path consistency to i-consistency, from directional arc consistency to directional path consistency to directional i-consistency.

So let's go back to this direction path consistency and ask the question when is DPC enough for backtrack search. And the relation that we are talking about is that, is the following that if the induced width, remember, that when we were looking at this algorithm called DPC, this algorithm, we had this step where you were adding edges. So you are going to add edges. As you enforce directional path consistency starting with the last variable to the first variable you may end up adding edges. Now, if the induced width of the graph, that is just the resultant graph is equal to 2, then DPC will make it backtrack free. What is the reasoning behind this or what is the argument here? That if the induced width of a graph is 2, that means every node has at most two parents and directional path consistency would have made sure that those two parents will only get such values that you can get a value for the current node essentially. So the induced width of the graph is 2, then DPC will be enough. So this, again you can see it is a generalization from the previous statement that we made that if the graph is a tree, or the induced width is 1, or the width is 1, then directional arc consistency was enough to solve it essentially. So just order it in such a way that the width is one and then it turns out in that case that induced width will also be one because if width is one, induced width will remain one because there is only one parent in the ordering and then directional arc consistency would have been enough. But if you are working with graphs of width 2, what happens essentially. So if you have a problem whose induced width is 2, whose width is 2, so for example this graph that we had looked at like this, you do any ordering of this and it is width 2 graph. Every node in fact has any two parents in any ordering essentially. For this particular example we had seen that this had become like this that one edge was added, I don’t remember which one, any way, depending on the ordering that we choose, one edge will get added to the this thing, or if you recall the diagram that we had drawn, in this particular ordering we had added one edge. Here of course induced width is 2 essentially, and therefore DPC is sufficient for backtrack free search. But this is the case for all problems of width 2. In this particular example it turned out that after, so this is the induced graph, right? And the induced graph has also width 2, because if we look at all the nodes, each one of them has exactly 2 or less parents. The last node has 2 parents, the second last node has 2 parents, the third last node or the second node has one parent and the root has no parents anyway essentially. So the induced width of the graph was 2, essentially. So that was kind of fortuitous, so a counter example. So I will just show you a small example where if you start with the following for some graph okay in some ordering essentially, let me give names to them, let's say A, B, C, D, E, so this is a graph whose width as you can seen is 2, but what happens when you do DPC, so when you do DPC, I will add edges in blue, E is related to D and E is related to A, so I will end up connecting A with E induced, but now you can see that here width equal to 3. You started with a width 2 graph and when you do directional path consistency at the very first step in this once you made A and D consistent with E, you ended up adding an edge between A and D, and now D has three edges, three parents, A is a parent of D, C is a parent D, and B is a parent D essentially. So which means that all those three must be consistent with D and for that I have to ensure four consistency. So the problem with doing this is that not necessarily because the induced width, so let me write induced width itself or which is the width of G^* may increase. So it is not enough essentially. So that is the difficulty you see that happens that as you impose higher and higher order of consistency you are imposing relations of higher arity so with path consistency you are adding relations of order 2 or you are adding edges and as a result the induced width of a graph may go up in this example I have shown you that it goes up to 3, that you may be even construct an example where it goes up to 4 and so on. So there is nothing you can really say about that essentially that if you have starting with a graph whose mean width ordering is 2 then it is not guaranteed that if you take the min induced width ordering also that its width will remain 2, which means it is not guaranteed that directional path consistency will ensure backtrack free search essentially. But just the last comment here, is that is it hard to, so we said that induced width is NP complete to compute and so on, to construct a minimum induced width graph is NP complete, but we can check that okay I have taken this width 2 graph to start with and I have done directional path consistency, is the new graph of width 2 essentially or I can do like this that I can do min induced width or something like that or min fill algorithm, I run the min fill algorithm on the graph and then and remember that is an greedy linear time algorithm, at the end of running that algorithm I will know what is the width, induced width essentially. So rather when I am doing min induced width algorithm which is a greedy algorithm, at some point if I find that the number of edges, number of parents for a node is more than 2, I know that is no longer induced width 2, so I know that is no longer induced width 2, so I can no longer do directional path consistency. So the next thing that we would like to do is to why decide in advance how much consistency to enforce essentially that is it, is directional arc consistency enough or path consistency or four consistency or five consistency, Why can't we have a flexible algorithm which will do an appropriate amount of consistency for every node, you know, why look at the whole graph and make a decision, each node may require difference degrees of consistencies. We will look at that in the next class and that is called adaptive consistency, and there is a related algorithm called bucket elimination. We will look at these two algorithms in the next class and after that we will move on to search essentially. Okay, so we'll stop here.