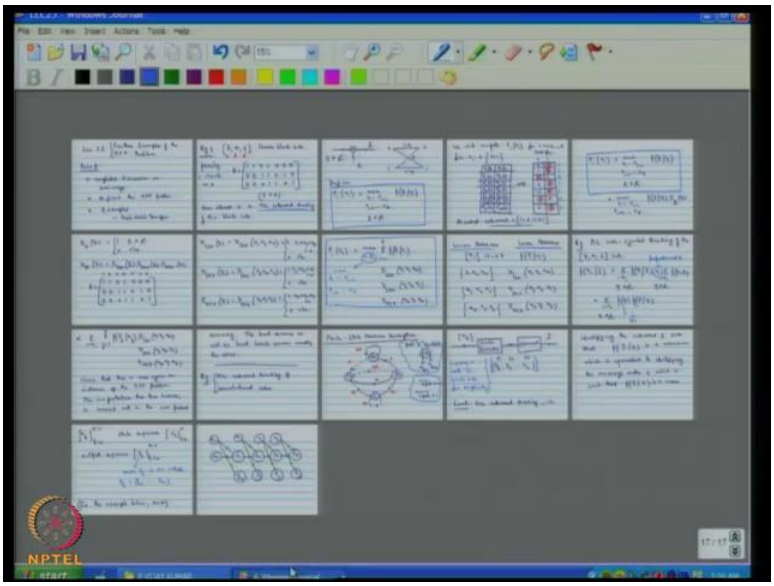
Error Correcting Codes
Prof. Dr. P Vijay Kumar
Department of Electrical Communication Engineering
Indian Institute of Science, Bangalore

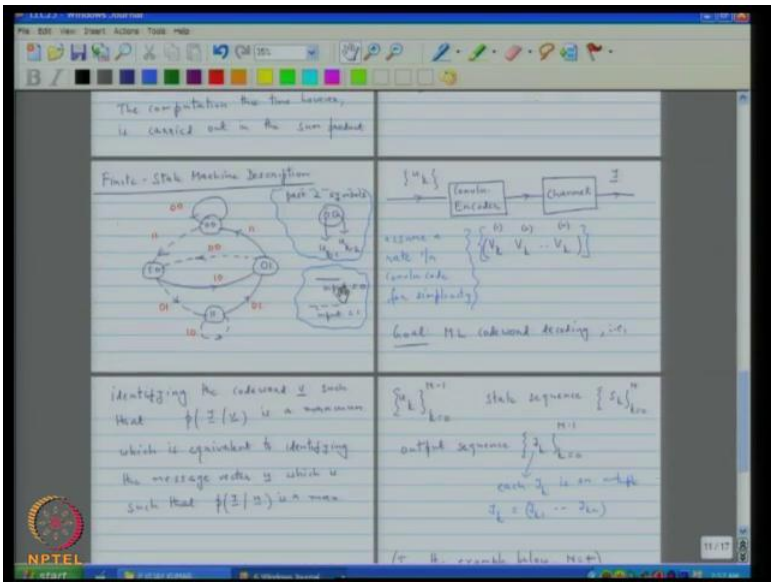
Lecture No. # 24
Junction trees recap

(Refer Slide Time: 00:23)



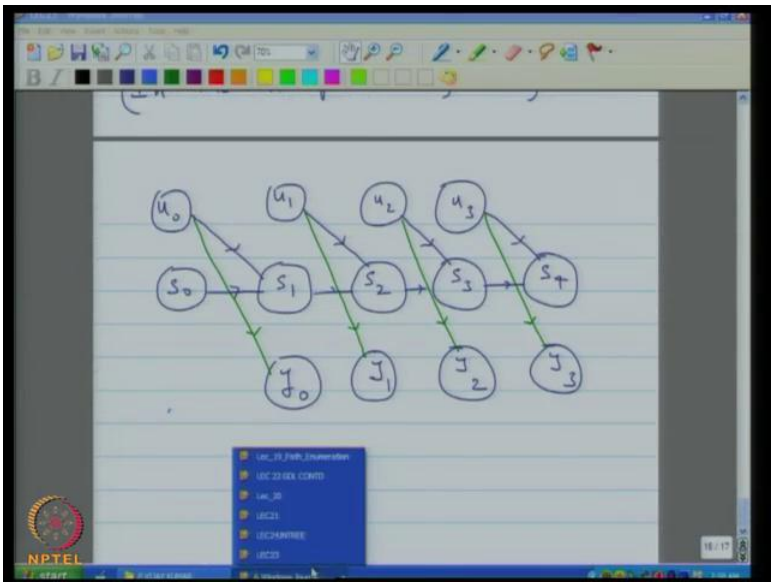
Good afternoon; welcome back, this is lecture 24. And as always I begin with quick recap what we were discussing last time; in the last lecture, we had, we were looking at other examples of MPF problem and the examples that we were looking at related to the decoding of codes; and our first example had to do with maximum likelihood decoding of block code. So, you can see that here that, we showed how you can formulate as an MPF problem.

(Refer Slide Time: 01:07)



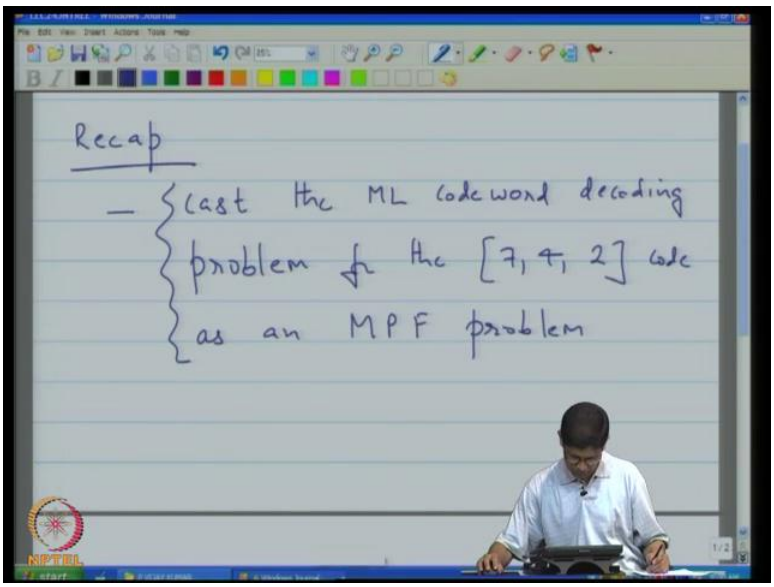
Then after we were done with that then, we shifted our attention to convolutional codes example; here I started off by saying that, we had many descriptions of convolutional codes along the way in this course, but the one that is relevant here is the one that is in terms of the state diagram. So, I just so this is the example state diagram that I provided for you, just to remind you. Then we introduced some notation, and I am restricting myself here convolutional codes have the rate k by n, where k is the number of inputs, and n is the number of outputs. Here, we are restricting attention to the case, when there exist one single input, and then there are n outputs. After introducing notation, I actually introduce this diagram.

(Refer Slide Time: 02:02)



We will actually continue from there, but let me get to our current lecture. The current lecture is entitle junction trees junction trees; so, our goal is actually provide an algorithm to solve the MPF problem; and it turns out that you may or may not always be able to solve the problem using this approach that will outline. But in the cases, when you can in the first step is to construct a graph, which is a particular kind of a graph known as a junction tree. We will come to that into course and I will skip that.

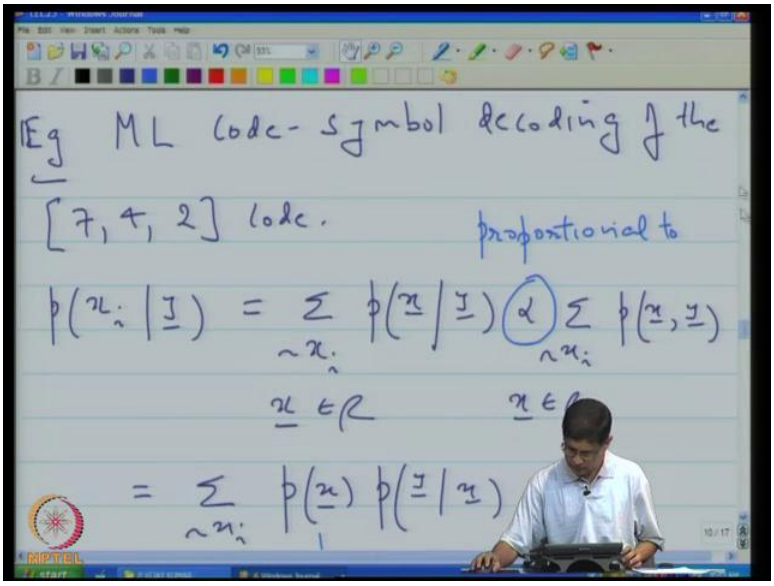
(Refer Slide Time: 02:49)



So recap, the cast the maximum likelihood codeword decoding problem for the 7 4 2 code as an MPF problem.

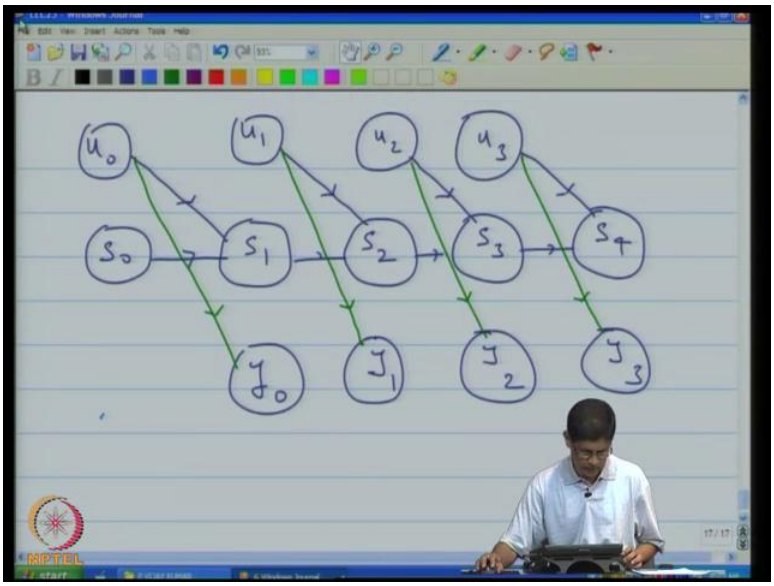
Let me quickly pull that of once; and then, very quickly we followed that up by showing that for the same code, you can have a second decoding algorithm, which is called the maximum likelihood codeword decoding algorithm.

(Refer Slide Time: 04:33)



For the same code, we have two different decoding algorithms; one is, maximum likelihood codeword decoding, the other one is, the maximum likelihood code symbol decoding. And the difference in the two cases was basically in one we were operating in the max product semantic and in other way operating with the sum product semantic. That is one of the nice things about the approach that will outline there it gathers together in a unified way, some different algorithms for different codes. The next thing that we did was did the same for maximum likelihood code symbol decoding of the same 7 4 2 code. Then I have just got started in trying to formulate began maximum likelihood codeword decoding of the, of a, of a rate of a rate 1 by n convolutional code.

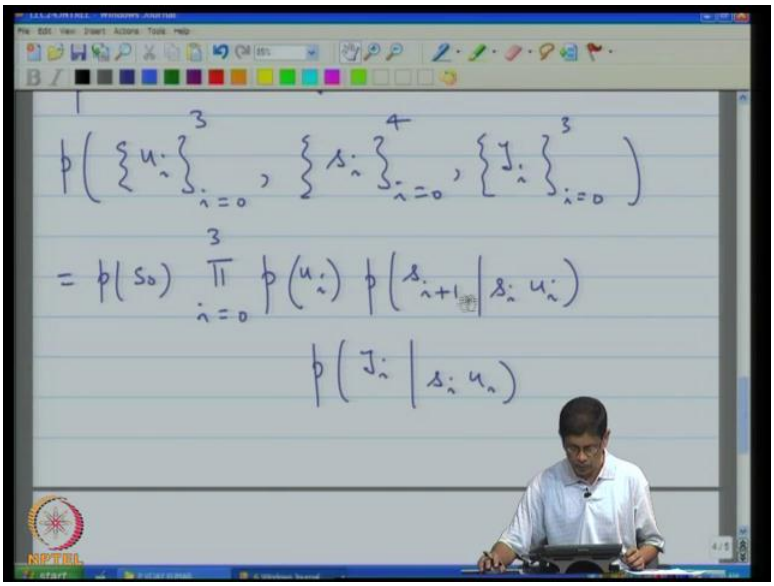
(Refer Slide Time: 06:55)



We will continue on that. Let me as a starting point pull up, I think moreover less last slide from the last lecture. We will take this, and copy it over... Here what I am actually doing is I am looking at, convolutional code with small number of information symbols. There are in fact four information symbols, which are u 0, u 1, u 2 and u 3, and then the convolutional code as initially state is 0, and subsequently goes through state s 1, s 2, s 3, and s 4. The corresponding received symbols are y 0, y 1, y 2 and y 3, given y 0, y 1, y 2 and y 3 they are asked to be covered u 0, u 1, u 2 and u 3. I pointed out last time that y 0, y 1, y 2, y 3 are in general n tuples, because your convolutional code being rate sometime for every input symbols there are n output symbols. So, each of these is a vector actually.

I want to write down an expression for joint probability distribution based upon inspection. So, the way you should interpret this diagram is designed at, if this is what is known as the detective acyclic graph associated to a probability, joint probability function. If you thing about the joint probability function of all of these variables then, this graph is actually telling you how it will fact? It will fact according to the product of p u 0, p u 1, p u 2, p u 3, p s 0 and then, p s 1 given u 0 and s 0, because there are arrows leading into it p of y 0 given u 0 s 0, p of s 2 given u 1 s 1, and so on. This graph suggests how you factor the probability mass function.

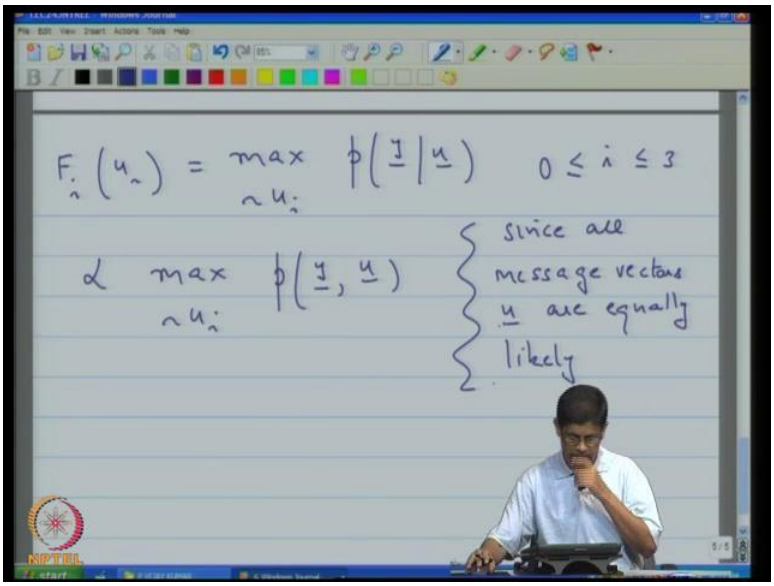
(Refer Slide Time: 10:35)



The above graph known as a directed as a directed acyclic graph and abbreviated as DAG, tells us how to factor the joint probability distribution function. So, factors according to p of u i, i is equal to 0 to 3 comma s i, i is equal to 0 to 4, and y i, i is equal to 0 to 3.The joint distribution function of all of these factors according to p of s 0, the product i is equal to 0 to 3 p of u i times p of s i plus 1 given s i u i times p of y i given s i u i. This is how it factors, and you can see that what the figure above showed us this is that, it is a graph that reflects this factorization, because there was in the arrow leading from s i and u i to s i plus 1; indicating there was dependence of s i plus 1 on the values of s i and u i.

You could see that here. For example s 2 is depending upon u 1 and s 1.These probability effect the probability of this. Now are your interest is in maximum likelihood codeword decoding of the convolutional codes.

(Refer Slide Time: 12:25)



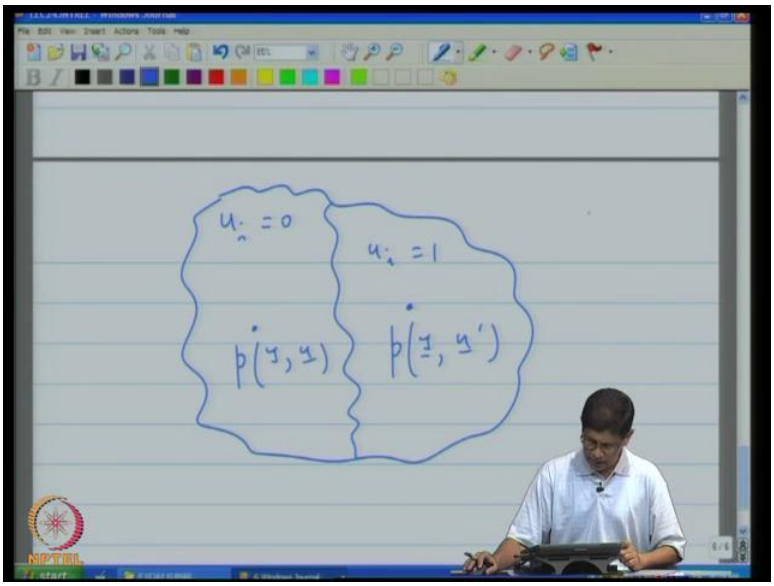
So, what will do is, will defined, will defined F i of u i will defined F i of u i to be the max over not u i of p of y given u. I think I already explained in the last lecture that there is a 1 to1 correspondence between codewords and necessary vectors, and which is y this is can also be regarded as maximum likelihood codeword decoding. In trying to understand how this complexion is augmented is same as we did last time, I will repeat it quickly. You compute this quantity for each i, there are three symbols will compute this for i ranging between 0 and 3 and you will actually compute it for each of them. Then, you will compare F i of u i equal to 0, and F i of u i for u i equal to 1, whichever is larger, you choose corresponding symbol, and the way to understand that this is maximum likelihood codeword decoding is to pretend that there is one message vector u, for which this quantity that this quantity here is the largest possible.

Then what you doing is you are uncovering the symbols of that 1 by 1, because when you do a comparison for u i equal to 0, let say that most likely codeword had all its symbols to be 1. When you put u i equal to 0 and u i equal to 1, that codeword will show up in the list corresponding to u i equal to 1, but it will not show up in the list u i equal to 0. So, since getting the max here you will actually by comparing larger quantity for the case u i equal to 1. So, in this way, you will actually 1 by 1 the stripping of the message symbols of the most likelihood codeword and we saw detailed example last time.

Once you understand there are goal is actually compute this for u i equal to 0, u i equal to 1. For each value of i in this range, that is our goal. Then the rest, now we need to formulate this is a MPF problem. So, we will do that. The first step is actually say where, since all message symbols are equally likely. I will write this is equal to... by the way, I think I explain last time that this tilda u i means that you are taking a max over all the symbols u 0 to u 3 other than u i. It is the not, and will see that before so this is the max over tilda u i. I can write p of y comma and actually it is proportional to this, it is proportional to this and the reasoning being that all message symbols, all message vectors are equally likely.

So, that justifies this step. Now what we are going to do is, we are going to take the max equal to the max over tilda u i of p of y comma u comma s.

(Refer Slide Time: 17:15)

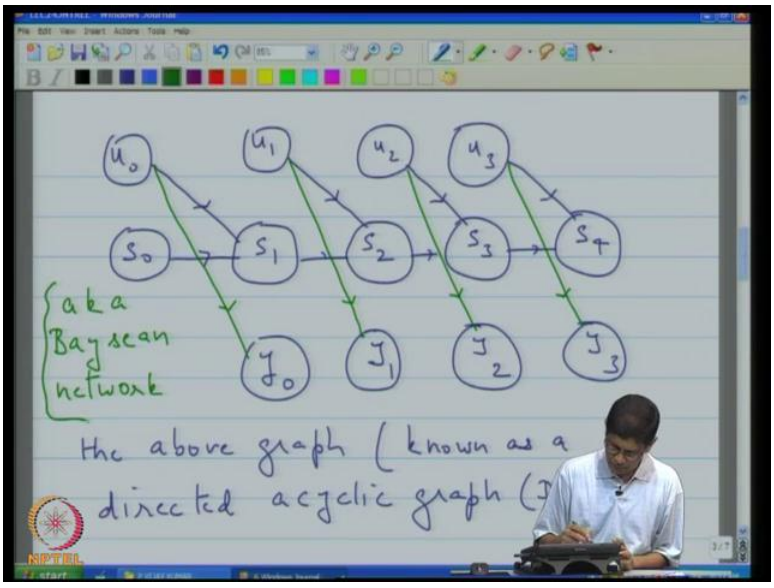


The reason for this, I will try to explain is just that then your, because what you are doing is maybe I should draw small figure here. You can divide the set of all message vectors into those in which, let say u i equal to 0, and u i equal to 1. Here, you will get, you will get for example, you will be computing p of y, u for some vector u, which is such that u i equal to 0. Here we will be computing p of y comma u prime, for some u prime such that u i equal to 1. You will be computing these are example quantities here.

Now, what this s does? s of course stands for the state. Now given message sequence u, s is completely determined. What that means is that, when you are computing the max over here you are you are also taking a max excuse me, you are taking a max not only that u i. But you are taking the max also, over the max over all vectors s. You might say that how can you justify that, how can you actually arbitrarily increase the variables said over which you taking the max, because basically you are taking a max of the set of real numbers. In here what you done is that to this original set all that you done is you added some 0s, because for any given message sequence and any given state sequence, this exact for any given message sequence is exactly then state sequence which is consistence with it. For a large majority of pairs u and s, this quantity will be 0. It will only be 1, for those which are comparable.

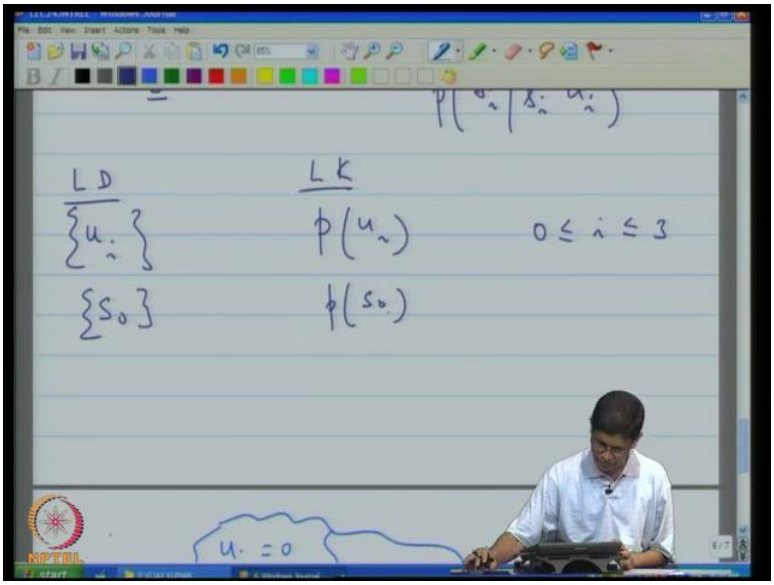
What you going to do is, here you are going to when you actually put in s, you are going whatever see when you trying to decide between 0 and 1, you are looking at all these values taking the largest value here, the largest value here, and listing them under 0 and 1, and choosing 0 or 1, according to which was large? Now what is going to happen is that, when you actually expand to include this new state variable s. You are going to have the same values in here with a lot of 0s thrown in. But that presently changed fact that, because when you computing the max of all the all these quantities are make it, you still recover the same value you still recover the same value here, and for this reason this is actually justify.

(Refer Slide Time: 20:16)



Now, I am going to actually expand the joint distribution function given r the directed acyclic graph here. This is also called a Baysean network sometimes. This is also called search try perhaps that it is also a k a Baysean network, because it tells is how the probability distribution factors.

(Refer Slide Time: 20:52)



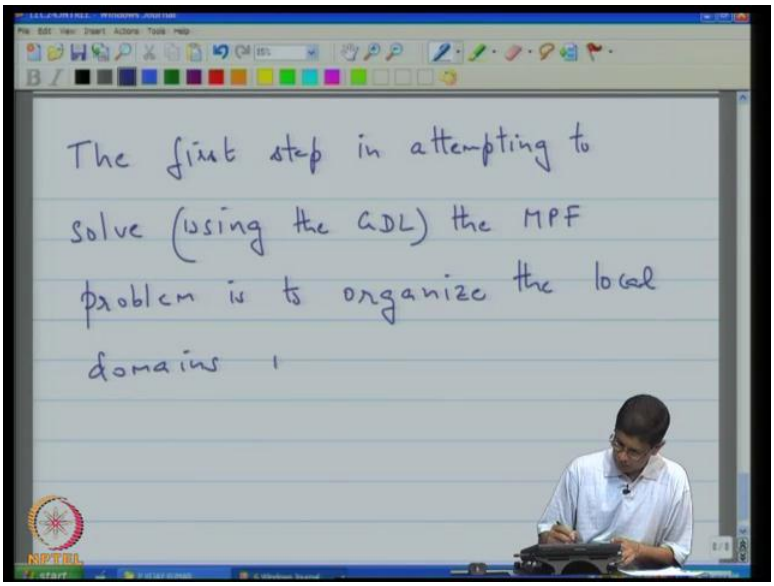
Using the Baysean network and the corresponding factorization, I can write this as the max tilda u i over all s of the exactly the expression we had earlier, p of s not i is equal to 0 to 3, p of u i, p of s i plus 1 given s i u i, p of y i given s i and u i. This is where we stop, because now we formulated this as an MPF problem, because you see that you have you have all these quantities here, which can be regarded as local kernels, and you have a global kernel which is the product of local kernel. You have marginalization with respect to certain set of variables. So, only thing that remains for us is to identify the local domains and local kernels.

The local domains are in this case the u i, i going from 0 to 3 with associated local kernels p of u i. Then you have the aparts to decrease the chance of confusion. Let me write this not is a set, because each u i is individually a local domains. So, perhaps let me modify this slightly. This is u i associated with local kernel p of u i, and this is for i ranging between 0 and 3, and then you have, you have s 0 associated to p of s 0, and you have s i plus 1, s i u i associated to p of s i plus 1 given s i u i, and then finally y sorry, finally just s i u i associated to p of y i given s i u i. Then,

these are the local domains, and these are a kernels; you can see that you multiply this and I marginalizing.

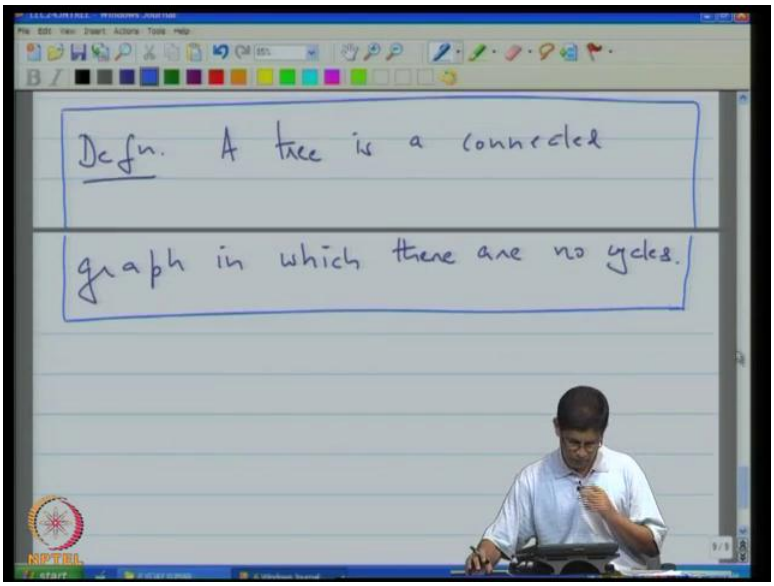
The conclusion is we have thus we have thus formulated the maximum likely hood codeword decoding of a convolutional code as an MPF problem. Now, we will introduce a new sub topic namely, we introduce the motion of junction tree. So, we outline the problem, the class of problem call the MPF problem. We have shown how the decoding of some example code can be formulated as an MPF problem. Now, we want to actually try to go solving the problem and what will actually doing is will be using the distributive law to give us an algorithm, which is of reduced complexity. The first step in applying the distributive law or the generalized distributive law, the GDL; the first step in applying the GDL is to attempt to organize the local domains into a so called junction tree.

(Refer Slide Time: 26:22)



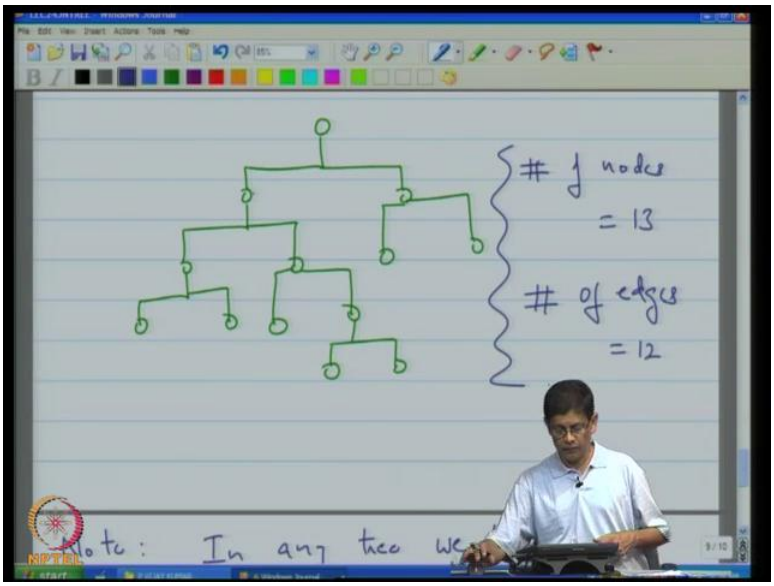
Let us make a note of that, the first step in attempting to solve using the GDL, the MPF problem is to organize the local domains into a form of graph known as a junction tree.

(Refer Slide Time: 27:55)



What is a junction tree? Definition, a tree is a graph, in which a tree is a graph is a connected graph, in which there are no cycles.

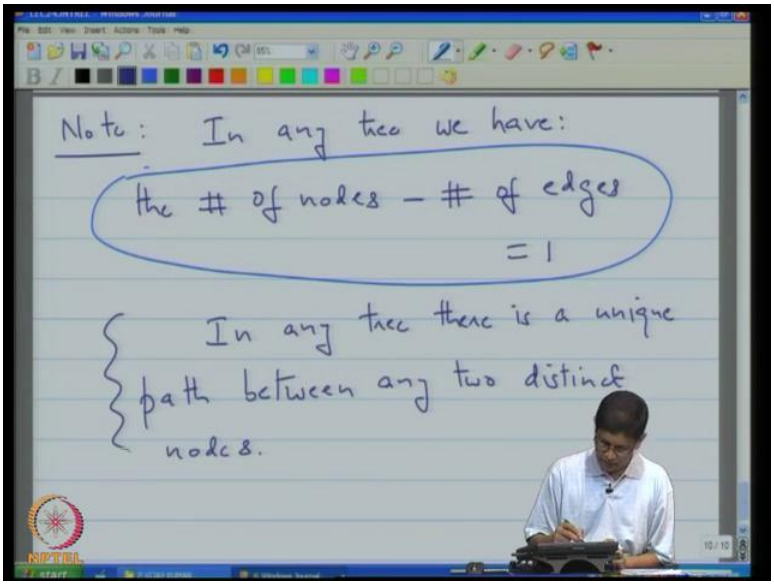
(Refer Slide Time: 28:58)



And couple of so, the generator, example of tree might be a graph that looks like this.Tree matrix like this, you can see that in a graph like this. These are nodes, and these are the edges. You can see that it is connected in the sense that, you can go by traversing the sequence of edges from any

node in this graph to any other node. There are no cycles in the sense that you cannot travel without retracing your steps around the graph, and land up with the same node at which you started. Example, if you start here, there is no way to actually comeback to this node without retracing your steps in some point. That is the picture, you should have in your mind of a tree.

(Refer Slide Time: 30:07)

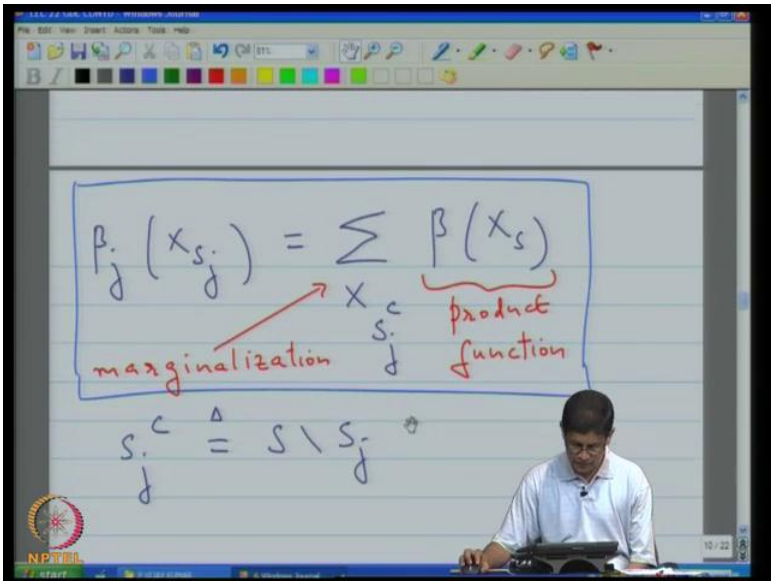


Guess some observations about trees. We call them nodes. Note, in any tree the, in any tree, we have we have that the number of nodes or what is sees, minus the number of edges is equal to 1. For example, if you look here. In this example, the number of nodes we have 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13. You have 13 nodes, and you defined an edge as a link between two nodes. So, you have 1 between these 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12. So, the number of edges is equal to 12. You can see that the number of nodes exactly exceeds the number of nodes, number of edges by 1. It is not have to proof. But we want do that. Then, the other, observation about trees is that between any two nodes there is a unique path. For example, if I wanted to go let say from let us call this node A here to node B. Then this exactly one path which will take me, that path is to follow like this. It would lead me this node which would lead me this node; so, this exactly one path that I have to follow.

In any tree there is a unique path between any two distinct nodes. Let us not have to see, because supposing you have more than path; then would actually have a cyclic, because you could go

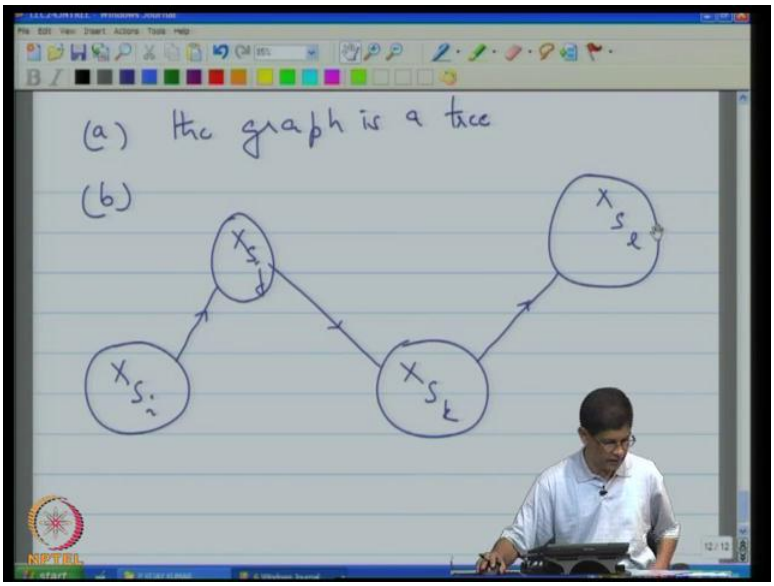
from node A to node B, along one path, and return by the other that would create a cyclic. Having defined tree, what is a junction tree? So, we are going to define a junction tree in a certain setting; definition, in the setting of the MPF problem; so let us quickly review what we mean by that.

(Refer Slide Time: 34:38)



The setting of the MPF problem this was from a couple of lectures ago, was where you have a universal set variables and you had subset which we call local domains and you have local kernels, global kernels and so on. So, there are these local domains. The idea is that we are going to call upon them. In the setting of the MPF problem, a junction tree is a graph whose nodes are in 1-1 correspondence with the local domains X of S i, i going from 1 to M. There are capital M nodes.

(Refer Slide Time: 36:33)



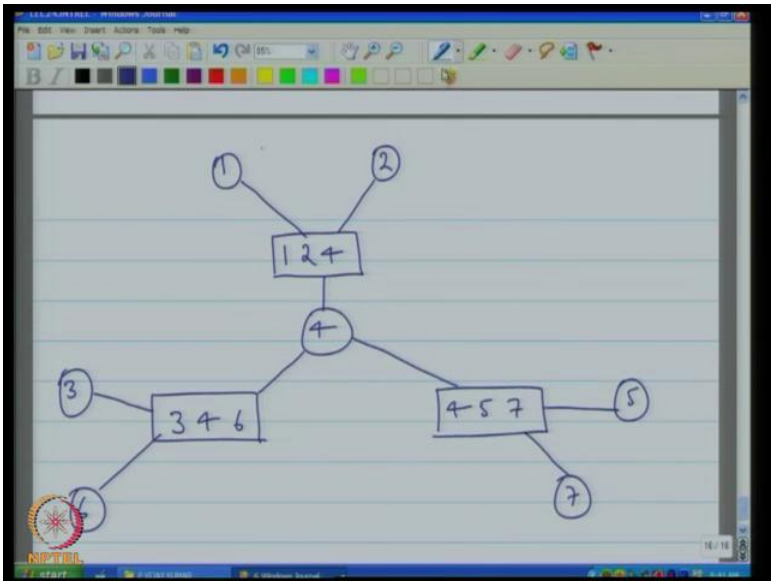
Each one corresponding to a certain local domain; and where edges are drawn edges are drawn between nodes in such a way that 1, the graph is a tree and 2, I will put down a picture...

That is that in the graph, you see that these are somewhat sees that are associated with the local domains. There actually going from X S i to X S l. We have the sequence of edges. What the second point says is that if there is a variable that is in the intersection of S i and S l, and that variable must be present along every one of the nodes that lead that lie on the unique path between S i and S l. We will put that nodes. The second property is if X, if I am erase that it will be better. For every node X S j on the unique path lying between nodes X S i and X S l, it must be that X it must be that X sorry, it must be that S j is the subset of S i of S i intersect S l. Look at example in just a second.

But just quickly getting back to this graph what we were saying is that on this graph. Here is S i, here is S l. So, nodes S j, S k happens to lie on this unique path. And, what we are saying is that, take a look the intersection S i intersect S l; whatever variables at present must be present in the variables corresponding to each of the intermediate nodes. These variables must belong to S j and S k. The meaning is not so clear from the definition, but we look at one example. Consider will look at the 7 4 2 code which we had example earlier let me see if I can pull up something from our earlier discussion on that.

Here we go. This is the base that I want, I am interested, then the decoding of our code let me put let me see let me carry this slide, I am going to pick this slide. You select this page and copy it... that is fine. And then, I am also going to copy a second page from our last lecture. Which was vary a listed the local domains. This is the page, let us select the page copy the page here. Cut so in our example we looking at the 7 4 2 code again. Here are the associated local domains.

(Refer Slide Time: 43:35)



So, the local domains for the x i(s) individual x i(s) x 1, x 2, x 4; the question is, can you organize these into a junction tree? Well, then the question that comes up is, how do you actually organize these into a junction tree? So, we will look at that sub sequent lecture, but I just want to show you what a junction tree looks like. I am going to draw so, I have to organize the local domains and i choose to organize them in vectors.

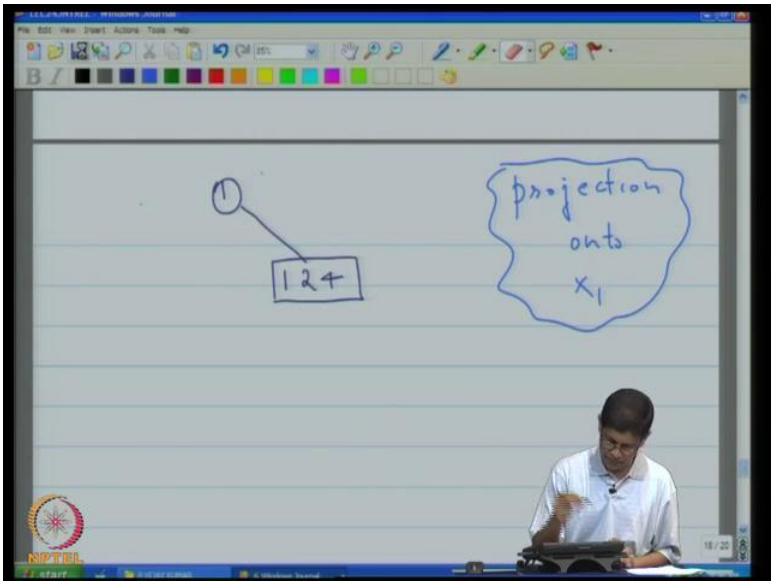
You can see that, what I have done to organize that all the local domains. By the way whether I write, when I write 124, and put that down is the local domain this is completely prominent you writing down X 1, X 2, X 4. These two are equivalent, because there is representations of the same local different representation of same local domain. Now, it is true that I actually form the graph out of them. It is obviously looking at this graph there are what I actually have is a tree. So, the question is, is this a junction tree? Now it shows a junction tree after pick for every pair

of nodes for take let us say, I take for example for take this one here, this node here and this. So, here the S i is 346, the S j is 124. The intersection between the 2 is 4.

So, what I need to make sure is that on the unique path between these two nodes which happens to be this, 4 is also include which is this. Similarly, if I look at 124 and 457 on this unique path, you must have every variable which appears in their intersection. But their intersection is 4 so, 4 also appears here. If you check this now, for the other conditions it is very easily checked, because 2 if you look between this node and this node for example. Their intersection is empty that we replaces no constrain. This then is a junction tree is an example of a junction tree. However there is an alternative definition of the junction tree which is important both for the theory as well as for very find the something is a junction tree.

Note, an alternative definition of a junction tree is a tree which when projected. I put this in quotes since this needs some explanation. Projected onto each individual variable also yields a tree. Example, let us go back over here, take this graph and paste it. Now, what I mean that projection is, you take this graph and supposing you wanted to project onto the variable 1 then, you would retain all the nodes in which 1 appears, and any edges between two nodes, which are connected by an edge.

(Refer Slide Time: 49:25)

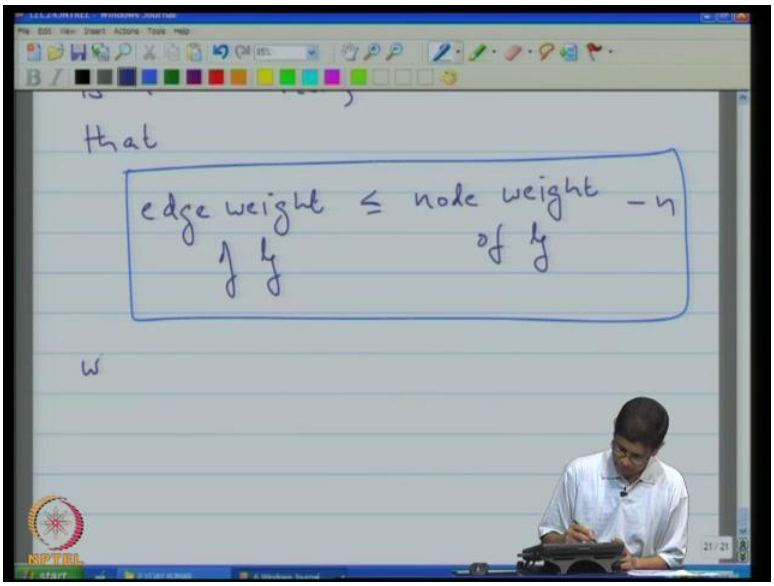


Here you would simply let us do one think let us copy this page over few times, because will use it couple times. The projection, let me put down that what we going to do here is the projection onto variable X 1. That is what we are interested. We ignore all the nodes we should do not have one and then none of these do. There are only two nodes left. We also removed the variables other than 1. Left, this kind of a graph, and this graph is a tree. We required that in every such projection, the end is all of be a tree. So, let us look at a second example. Here...

The it is a anti symmetric projection not into X 1, but rather onto X 4. I am interested in projection onto X 4. So what I do is, I do the same think, I ignore all nodes in which X 4 does not appear. Then, once in the remaining nodes, I actually delete the variables other than sorry, I should have kept X 4. I delete all the variables other than X 4 or 4 that appear. So, I am left to this. I can see that this is a tree. This is an alternative definition of the junction tree, their requirements is not had to prove, because after all in this graph what we required was that for example, we required that if 4 is present here and 4 is, if 4 is present in this node, is present in this node.

Then, it must be present in all the intermediate nodes that lie on the unique path leading from here to here. But, that is equivalent excuse me, that is equivalent is saying that when you project this entire graph onto 4, that since 4 is presented in this node the projection will have a node corresponding to this node. It will have a node corresponding to this node. These node must be connected this is why the definition the projection be connected is equivalent to the definition that, 4 be present on the node of every in the variable set of every node that lies on this part. The two definitions are in fact, are equivalent. Now, having done the then, connects question that raises. You defined a junction tree but, how do I go back constructing a junction tree.

(Refer Slide Time: 53:17)



This is the question how does one construct a junction tree? The answer is that there are algorithms in graph theory, which tell you how to construct a maximal weight spanning tree. What we are going to do is, we are going to adapt one of those algorithms. But first we have to actually a make connection between the two between spanning trees, maximal weight spanning trees and junction trees. Answer, by extracting or rather by constructing a maximal weight spanning tree. It is not little check typically see references to algorithms for connect, for constructing minimal weight spanning trees, but it is not have to see that you can actually adapt a construction for a minimal weight spanning tree into a construction for a maximum weight spanning tree.

What is the connection? The connection is this. Theorem, if g is a graph whose nodes correspond to the local domains of an MPF problem, and which is also a tree is also a tree, then it must be that the edge weight of g must be less than or equal to the node weight of g minus n, where n is the number of variables which is also the size of the universal set S. We just have had enough time to state the theorem. We will actually provide it in the next class, but just to very quickly summarize what we done today is, we formulated the problem of maximum likely hood decoding of a convolutional code as an MPF problem. Then I have said the next step in our discussion on the GDL is to show you how the GDL attempts to solve the problem, the MPF

problem. The first step is to construct what is called a junction tree? So, we will continue in the next class. Thank you.