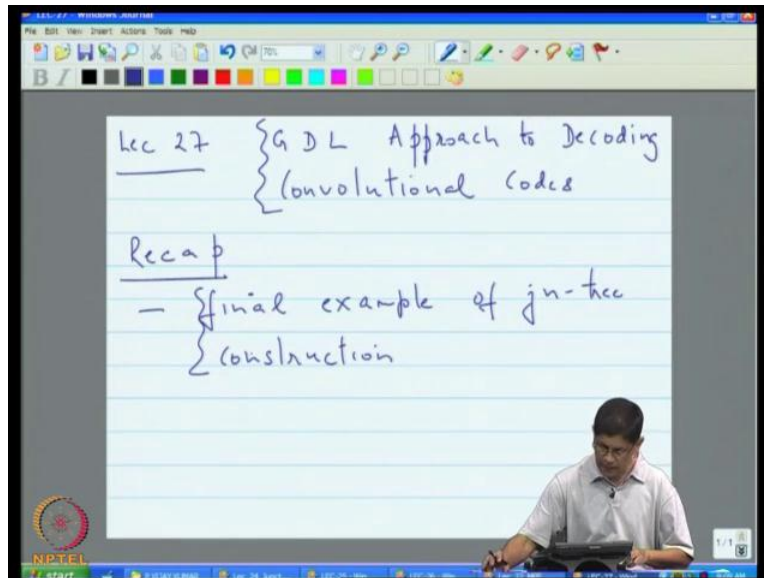


**Error Correcting Codes**  
**Prof. Dr. P Vijay Kumar**  
**Department of Electrical Communication Engineering**  
**Indian Institute of Science, Bangalore**

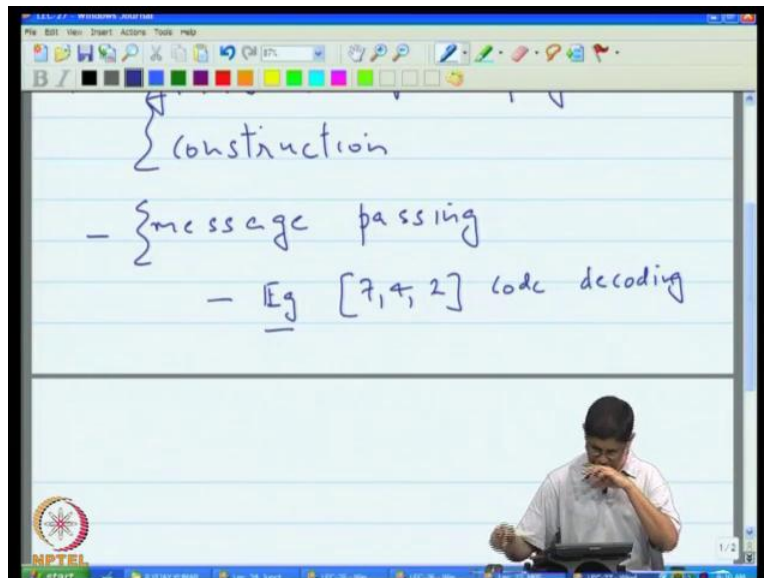
**Lecture No. # 27**  
**GDL Approach to Decoding Convolutional Codes**

(Refer Slide Time: 00:16)



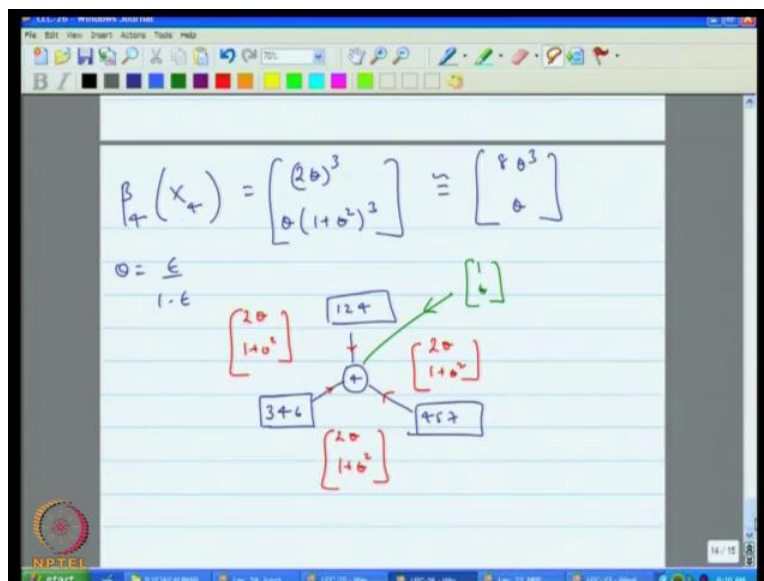
The previous lecture was I gave you our final example of junction tree construction. Then I moved on to the topic of **moved on to the topic of** message passing, because that is our next step in solving the NTF problem.

(Refer Slide Time: 01:00)



And in the last lecture, I gave you formula for message passing, and then we looked at an example relating to decoding of 7 4 2 code decoding. And let me just take you through, walk you through some of that once again. So the first part of the lecture have to do with the example of junction tree construction, and that was concluded here.

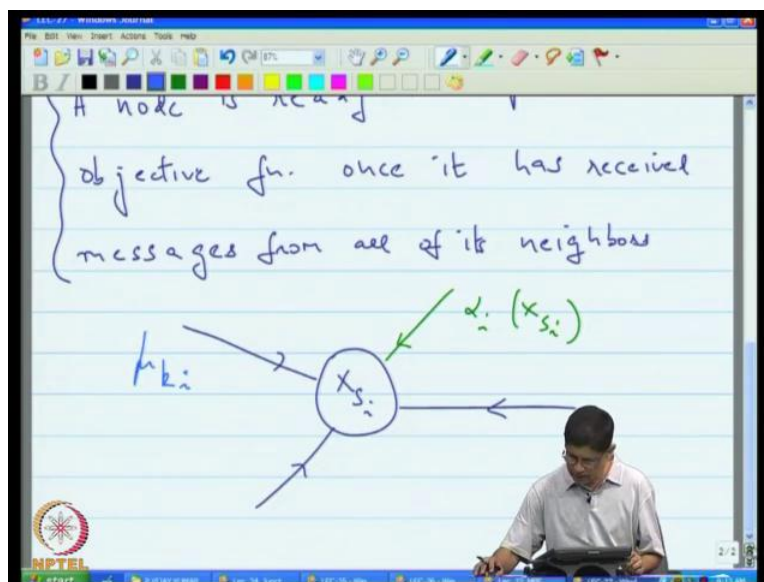
(Refer Slide Time: 02:00)



Then I give you a formula for message passing, and I we applied it to the 7 4 2 code and we pass messages on the 7 4 2 code, I shown here and eventually entered up. Now all I introduce another notion that is that of a schedule, so you pass messages in accordance with some schedule, and the schedule is the function of how many objective functions you want to compute, if there is just a single objective function that you want to compute, then the right thing to actually do is to orient all the edges in a network towards the particular that local domain associated with that objective function, which is  $x_4$  in this case. And then you pass messages and we did that.

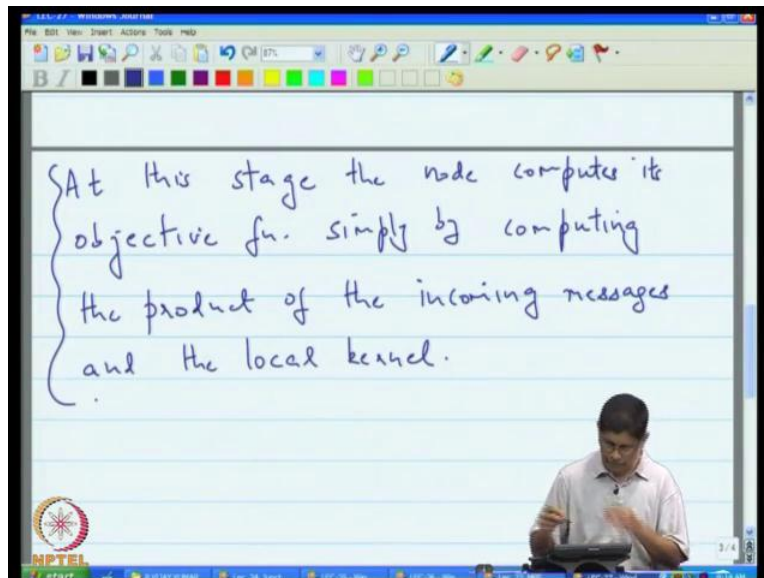
And then eventually end up with the point where you have all the messages that are incoming to your objective function and then the question is well what to you do here, and I said you just multiply them and that is true, but let me just do the following. I will come back to this, I just want to state a general principle.

(Refer Slide Time: 03:25)



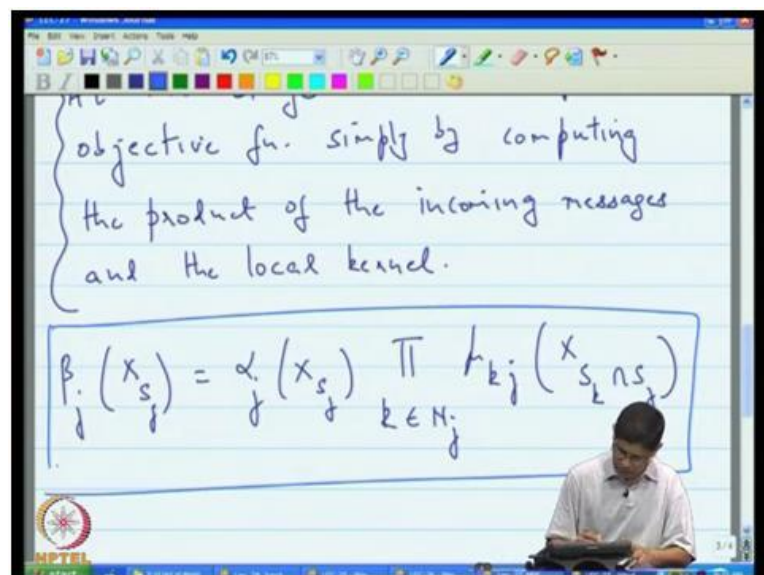
So, a node is ready to compute its objective function, once it has received messages from all of its neighbors. So, in the general situation that we are node here, let us say  $x_{s_i}$  and there will be messages coming from all of its neighbors, and will all be its local kernel.  $\alpha_i(x_{s_i})$  and these are all the  $\mu_{k_i}(s)$  that are actually coming in. These are all the  $\mu$  messages,  $\mu_{k_i}$  that are coming from the other nodes. And then all they were means to do is simply to multiply all of them.

(Refer Slide Time: 05:00)



At this stage, the node computes its objective function simply by computing the product of the incoming messages and the local kernel.

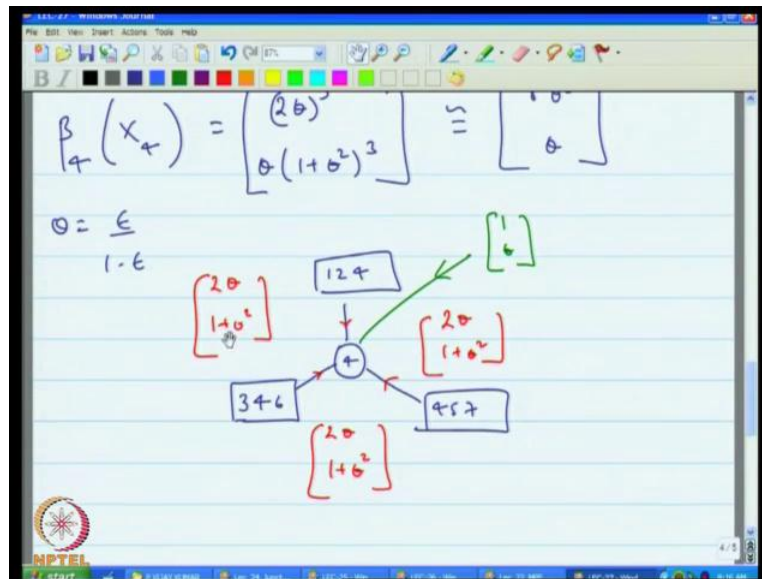
(Refer Slide Time: 06:00)



So, in other words  $\beta_j(x_{s_j})$  is  $\alpha_j(x_{s_j})$  times the product of  $k$ , the neighbors of  $j$  such that  $x_{s_k} \in s_j$ . So, the objective function is simply equivalent to this product computation, take all the incoming messages multiply them and list. We think is to keep

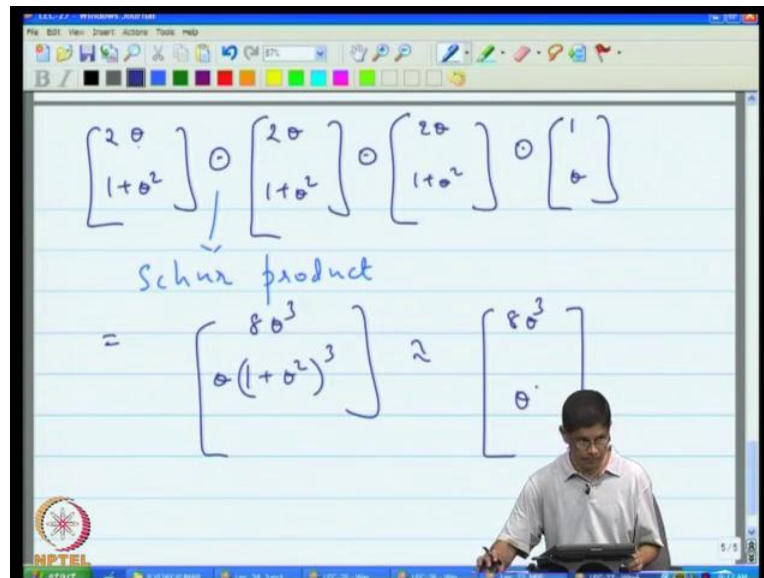
in mind is that all of these are functions when we actually multiply these functions and then when you represent functions as vectors multiplying functions is equaling to multiplying vectors component by component and that is all called the show product of a branch of vector. So, towards the end that is exactly what we were doing, we were computing the shoot product of all the incoming vectors, let me just select this page and copy it for our current lecture.

(Refer Slide Time: 08:00)



We have this, so you can see here what I meant, because the incoming messages are  $2\theta$  plus  $\theta$  square from each of these nodes, so you multiplied these together the local kernel is this; so what you are doing?

(Refer Slide Time: 08:20)



$$\begin{bmatrix} 2\theta & 0 \\ 0 & 1+\theta^2 \end{bmatrix} \odot \begin{bmatrix} 2\theta & 0 \\ 0 & 1+\theta^2 \end{bmatrix} = \begin{bmatrix} 8\theta^3 & 0 \\ 0 & \theta(1+\theta^2)^3 \end{bmatrix} \approx \begin{bmatrix} 8\theta^3 & 0 \\ 0 & \theta \end{bmatrix}$$

Schur product

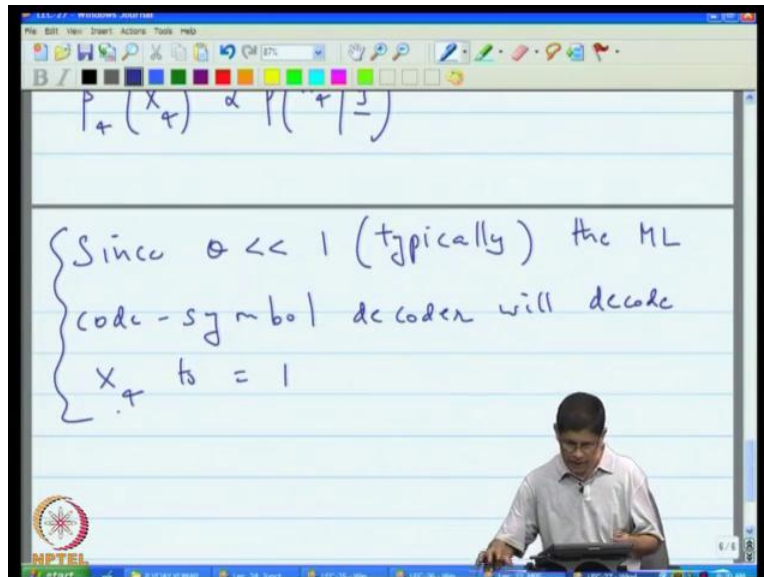
You could also view this computation as being  $2\theta$  plus  $\theta^2$  schur product, which is the component by component product  $1$  plus  $\theta^2$  component by component product and then final, we have the local kernel which is one  $\theta$ , you also put that in and you have one  $\theta$ , if you take the component by component product this is the schur product, so it shows the component by component product. So, this is equal to  $8\theta^3$   $1$  plus  $\theta^2$  the hole cubed into  $\theta$  and because  $\theta$  is small typically, the application of interest, you can approximate this by  $8\theta^3$ , and then  $\theta$  at the bottom; that is what we ended up in last lecture.

Now, remember that your objective function was really, so the objective function, which in this case is  $\beta_4 \times 4$  was really proportional to the probability of  $x_4$  given there is the vector. And given that your vector was part of a code; that was how we started out that was a problem we started out with in the lecture you see here, we had this quantity as the quantity that we were interested in computing here, this one here. And so that is this, and now what this is saying is look this is saying that the probability that, and then the way we always I mean, there is a choice of ordering when you are listing the values of a function as vector, so this first component corresponds to  $x_4$  equal to  $0$ , and this is  $x_4$  equal to  $0$  and this is  $x_4$  equal to  $1$ . What this is saying is that the likelihood of  $x_4$  being a  $0$ , given that you see that was  $8\theta^3$ , but the likelihood there will actually a  $1$  given by  $\theta$  and since  $\theta$  is small, this number is much



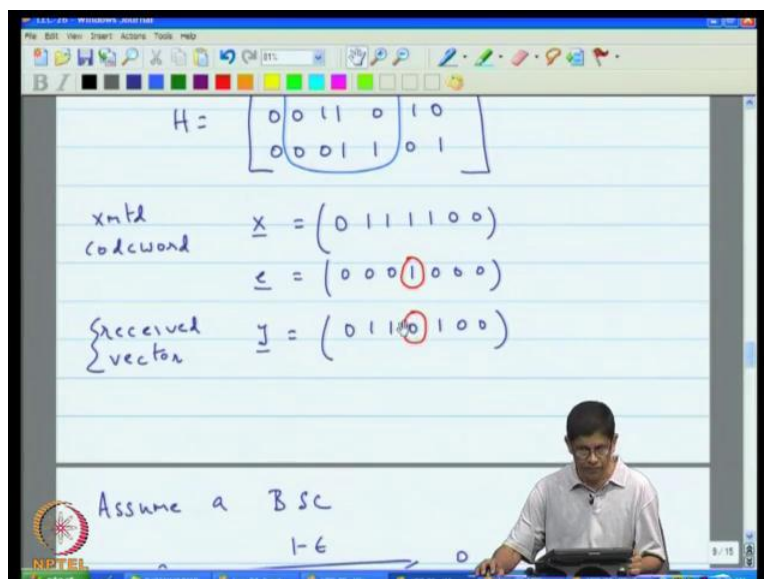
larger remember, what maximum likelihood code symbol decoding will do at that stage is to declare  $x_4$  to be a 1.

(Refer Slide Time: 11:00)



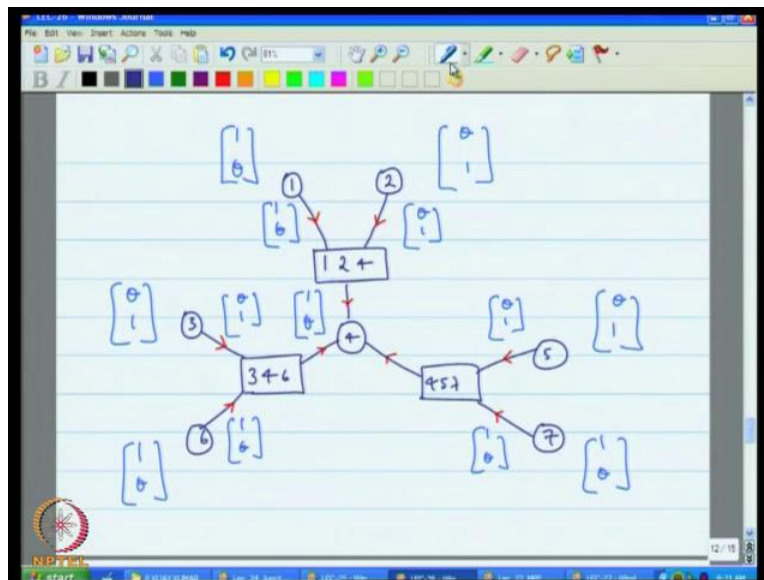
Since, Let me do it on the next page; since, theta is much smaller than 1 typically, the maximum likelihood code symbol decoder will decode  $x_4$  to equal 1. And now, if you go back to our last lecture, we will see that, we will have corrected the error that we introduce across the channel.

(Refer Slide Time: 12:00)



So, here in this example, this was the transmitted code word so indeed the fourth symbol was equal to one, but we are introduced an error here, so that the received vector was this what, so we have enable to do using the single vertex passion of the GDL is to actually that restore the correct value.

(Refer Slide Time: 12:30)



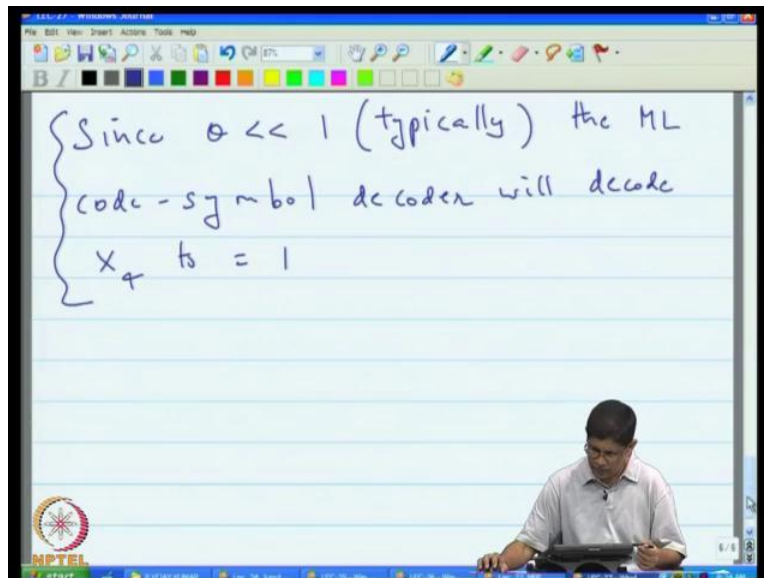
Now a comment, in a way I am going back and forth between lectures, last lecture and the present lecture hope that is okay, I just want to actually make one point here. Let, me just perhaps I should, let me here is make that point with respect to this figure here. Again this is from here this is the twelfth slide from your last lecture, lecture twenty six. The question is well, what is the justification for what you have doing, I mean you are telling some steps, you are passing me messages, but what is the justification? The justification is simple this that the objective function that you wish to actually compute, which is this one here, is here if you look at it you will see that what you want to compute is the product of all the local kernels, and then followed by marginalization, which means you want to get rid of the variables there are not interest in some sense, which in this case are  $x_1 \times x_2 \times x_3$  and then  $x_5 \times x_6 \times x_7$ . The only thing that GDL a saying well, I understand that is what you want to do and then, but I am trying to be a little clever in terms of when I marginalizing, because node one looks and says look I have information about node one, and it passes this here.



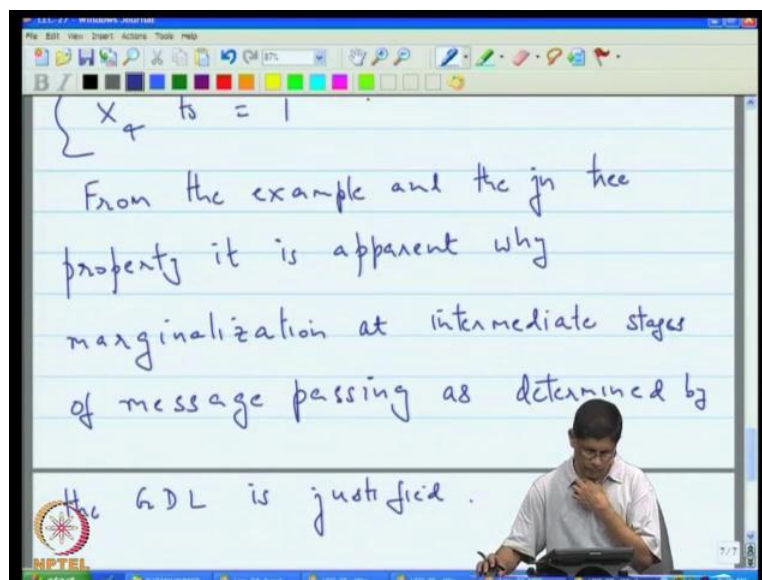
This node looks at node four and says look I have information caring concerning  $x_1$ , but that is of no interest to this. The brute force we have doing the computation would be to collect all the local kernels have them passed, and then do the marginalization, finally and only finally at the objective node. But the smart thing to do is to do the marginalization where ever possible, because then it reduces the over head in terms of both competition as well as how much of message you pass, because the message first passed across here is the function of a smaller number of variables. And all your computation will actually a therefore be simpler, because when you are multiplying you are involving a smaller set of variable. That is the read why, that is exactly what GDL does; so that all at the same time justifies the procedure. It tells you that nothing much is really going on, the key property that you are using is that of the GDL. Now, so for example, supposing instead of actually trying to compute at the objective function at node 4, I was trying to compute the objective function at node six let us say, when I would orient all my edges towards node six.

And then here what would happen is that I would marginalize with respect to  $x_1$ , because I do this node looks across this gap and says, I have information which is a function of  $x_1 \times x_2 \times x_4$  but on the other side, I see only  $x_4$  from then marginalize with respect to  $x_1 \times x_2$  and the question minds wells in your mind well if you got rid of all the information of  $x_1 \times x_2$ , how do you know that it is not going to pop up somewhere here. What if our node here, a node here has information about  $x_1$  and  $x_2$ ? And may not in trouble, but that is a beauty of a junction tree because if  $x_1$  is present here and not present here, and present somewhere else, then it has to be present along all the nodes along the unique part joining these two nodes. So the very fact that  $x_1$  is here and  $x_4$  is in  $x_1$  is not here you once you see that you can rest you sure that the  $x_1$  will not appear and this part of the tree. So, that is what justifies your marginalization at this stage that is another point.

(Refer Slide Time: 16:15)



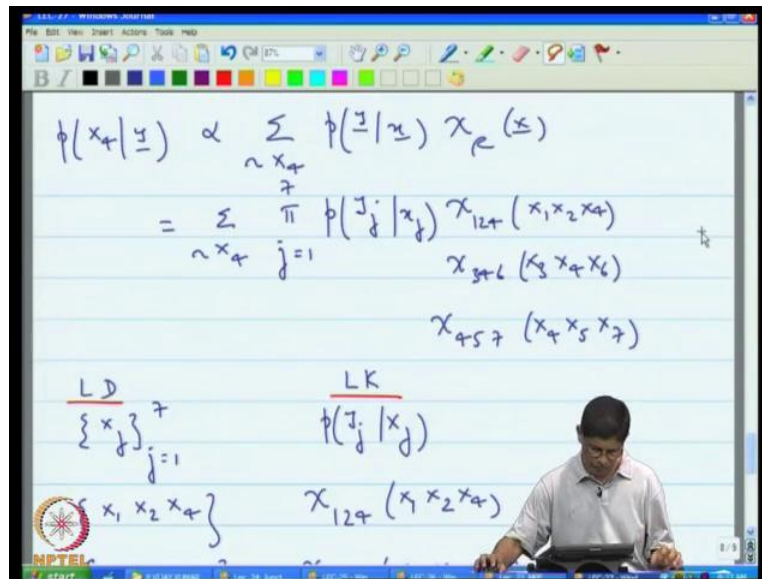
(Refer Slide Time: 17:00)



So now, the next question is, well now we justify how a maximum likelihood code symbol decoding is achieved, so I just make a very short note about this from the example and the junction tree property, it is apparent why marginalization at intermediate stages of message passing.

As determined by the GDL is justified. Now, I want to actually repeat the same example that we did here with the difference that I want to consider in place of maximum likelihood code symbol decoding. I would like to actually carry out maximum likelihood code word decoding. Take this end code, but now I am see maximum likelihood code word decoding. So the question is what changes would be required in order to enable us to do that.

(Refer Slide Time: 19:00)



$$p(x_4|y) \propto \sum_{x_4} p(y|x) \chi_{\mathcal{C}}(x)$$

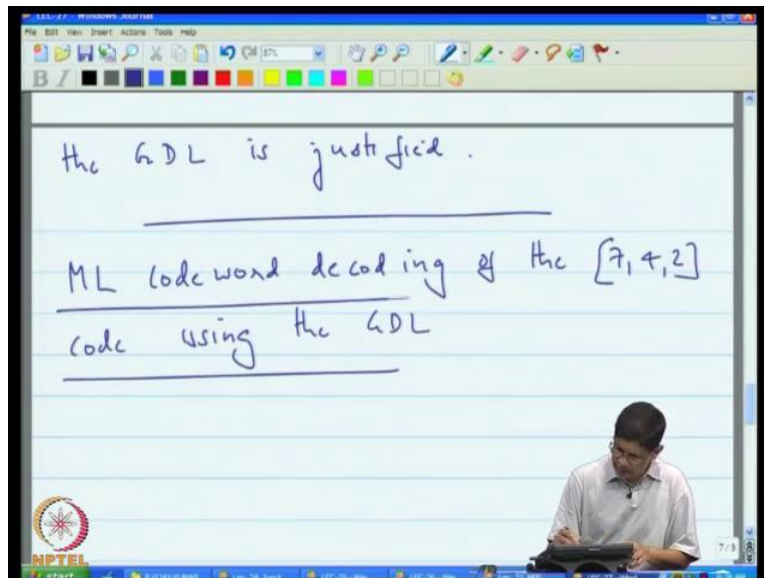
$$= \sum_{x_4} \prod_{j=1}^7 p(j_j|x_j) \chi_{124}(x_1 x_2 x_4) \chi_{346}(x_3 x_4 x_6) \chi_{457}(x_4 x_5 x_7)$$

LD  $\{x_j\}_{j=1}^7$   $\{x_1 x_2 x_4\}$

LK  $p(j_j|x_j)$   $\chi_{124}(x_1 x_2 x_4)$

And as I pointed out, what you could do is, you could the only change that is really required is that instead of the sum product the summation sign here, you would carry out marginalization using the maximum operator. So, let us just copy this page on to the current lecture.

(Refer Slide Time: 20:00)



And talk about maximum likelihood code word decoding of the 7 4 2 code using the G D L and I think in a earlier lecture we had in fact set up the objective function, let me see if I can actually fill that up.

Here we go so in our twenty third lecture, we had there is an example this linear code and our interest was in maximum likelihood code word decoding of this particular block code. And towards this end, what we have done was, we had introduce this quantities  $F_i$  of  $x_i$  and we have define it like this, and if you break this it turns out to an end up with an expression like this, you end up with this objective function.

(Refer Slide Time: 21:00)

$$\hat{F}_i(\hat{x}_i) = \max_{\hat{x}_i} \prod_{j=1}^7 p(\hat{j}_i | \hat{x}_i)$$

$$\begin{aligned} & \max_{x_1 \dots x_{i-1}} \chi^2_{124}(x_1, x_2, x_4) \cdot \\ & \max_{x_{i+1} \dots x_7} \chi^2_{346}(x_3, x_4, x_6) \cdot \\ & \chi^2_{457}(x_4, x_5, x_7) \end{aligned}$$

Let us copy this page as well on to our current lecture.

So, with the same code we now have this I guess I can actually with this is here for just comparing purpose what we speak, perhaps I will just move this up so that we can readily compare these two quantities. So, if you look at this expression on this page. This is our objective function may needed to carry out maximum likelihood code word decoding barrowed from a previous lecture, and you can see that all the local kernels are the same, you are doing the same computation except that you have max here; whereas in the case of maximal likelihood code word symbol decoding, what you actually had here, let me removed this; is the same thing except that you have the summation. So given that, its decoding the change in strategy immediate to carry out maximal likelihood code word decoding is almost immediate, and what I will do now is given this are I will (( )) are all going to be changed accordingly. What do I mean by that?

(Refer Slide Time: 23:00)

$$h_{ij}(x_{s_i}, x_{s_j}) = \sum_{x_{s_k} \in N_j} d_i(x_{s_i}, x_{s_k})$$

$$\prod_{k \in N_i, k \neq j} h_{ki}(x_{s_k}, x_{s_i})$$

$N_i = \text{set of neighbors of } x_{s_i}$

I mean that in the last lecture, we put down an algorithm, we put down an algorithm for message passing, which was given here. Now this message passing algorithm assumed that this that the semi ring in which you are operating is the sum product semi ring. So, in fact this notation summation really stands for the first of the two operation in the semi ring; in the sum product semi ring, this would be a summation and that is a product. But on the other hand, if you looking at max product semi ring, then this summation would be replaced by a max, and this product would be stay as it is. So, that is how you should interpret this. And similarly, we said in the beginning of the current lecture at the end, when you are computing the global kernel, what you do is you just take the product of all of this. So, this product is really stands for the second of the two operators in the semi ring. For example again since we are talking about the sum product semi ring, here what we have is the product.



(Refer Slide Time: 24:00)

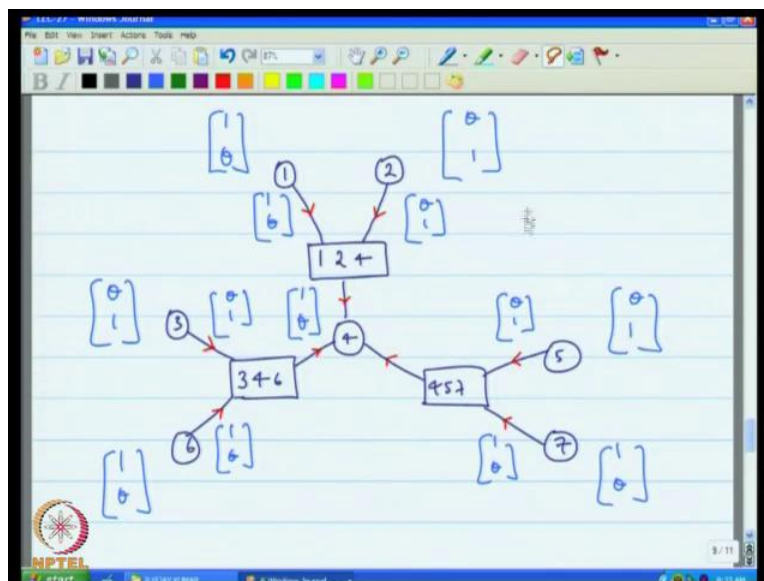
(and the local kernel.

$$\beta_j(x_{s_j}) = \alpha_j(x_{s_j}) \prod_{k \in N_j} h_{kj}(x_{s_k} n_{s_j})$$

$$\beta(x_x) = \begin{bmatrix} (2\theta)^3 \\ \theta(1+\theta^2)^3 \end{bmatrix} \approx \begin{bmatrix} 8\theta \\ \theta \end{bmatrix}$$

But there is another semi ring called the min sum semi ring, in which case the second of the two operation is the summation; so it actually be adding instead of multiplying; please keep that in mind. So, the notation here is with respect to the sum product semi ring. Once we are clear on that, it becomes clear how to actually carry out message passing, we can go back to this example.

(Refer Slide Time: 25:00)

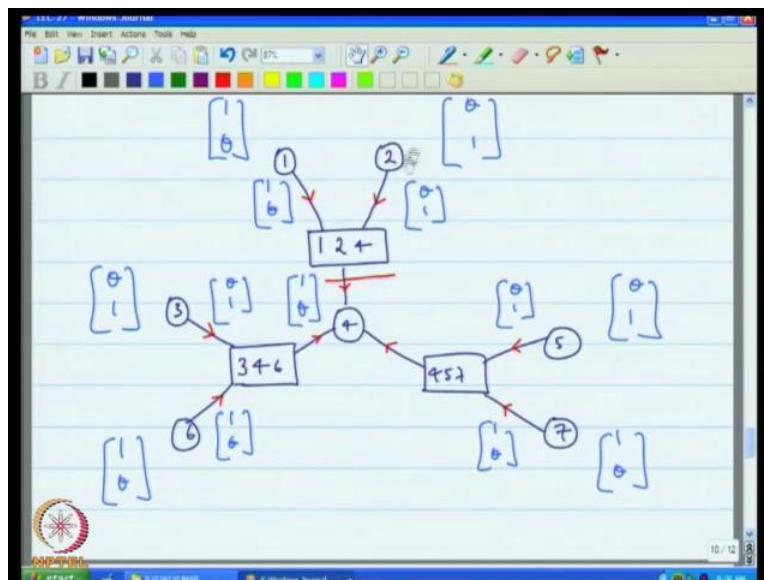


Let us go back to this example and carry on from there.

Here, we have the junction tree, and this time I want to tackle exactly the same example, the same setting exactly as before. And see, what difference comes about, since we have employing the max product. So the first thing that is clear is that here initial messages are all the same; so I can actually restore them I guess, I did not need to delete them, since the initial messages are all the same. So, the messages that are incoming from each of the leaf nodes are exactly the local kernels, and these do not change. And again let us assume that so the setting is the same as before; here we will do decode the 7 4 2 code once again, for  $y$  equal to 0 1 1 0 1 6 7.

We had remembered we had introduced an arrow in the fourth symbol, so will be still carry that over. So, for that same example with the same notation as earlier that initial wave of message is the same coming in from the leaf nodes. And the first marginalization that carried out is actually here; so the question is what is a marginalization that you would carry out between these two nodes; in the sum product summing you would sum over the variables  $x_1$  and  $x_2$ , here would actually take the max. So, what you are going to do is, I always feel more comfortable, when we have a little bit must space to write let me again copy this over.

(Refer Slide Time: 28:00)



So here, we want to focus on the computation, the computation indicated by this red line over here. I am going to delete me of this are the stuff, which we do not need.

So, the question is what is a computation, let us actually being carried out, and that marginalization with your actually carrying out here is the max over  $x_1 \times 2$  of  $p$  of  $y_1 \times 1 \times p$  of  $y_2 \times 2$  times  $k_{124}$  of  $x_1 \times 2 \times 4$ . And this is our function  $g$  of  $x_4$ , exactly like we had last time we had the summation, so that things of  $x_4$  is 0, then this is going to force there  $x_1$  plus  $x_1$  agree with  $x_2$ . And when you take the max, what you are going to do is you are going to take the max, so here for example, you will multiply 1 with theta one by theta and take the max of these.

(Refer Slide Time: 30:00)

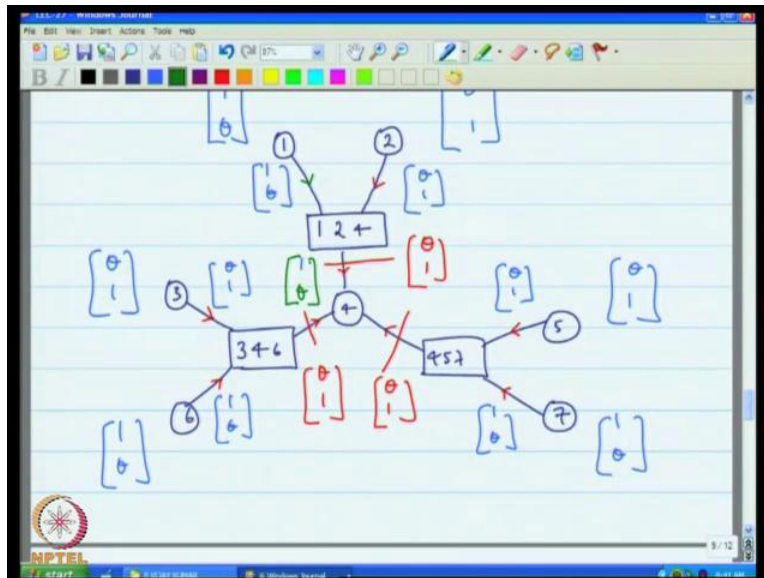
$$g(x_4) = \max_{x_1, x_2} p(y_1/x_1) p(y_2/x_2) k_{124}(x_1, x_2, x_4)$$

$$g(0) = \max \{ 1 \cdot \theta, \theta \cdot 1 \} = \theta$$

$$g(1) = \max \{ 1 \cdot 1, \theta \cdot \theta \} = 1$$

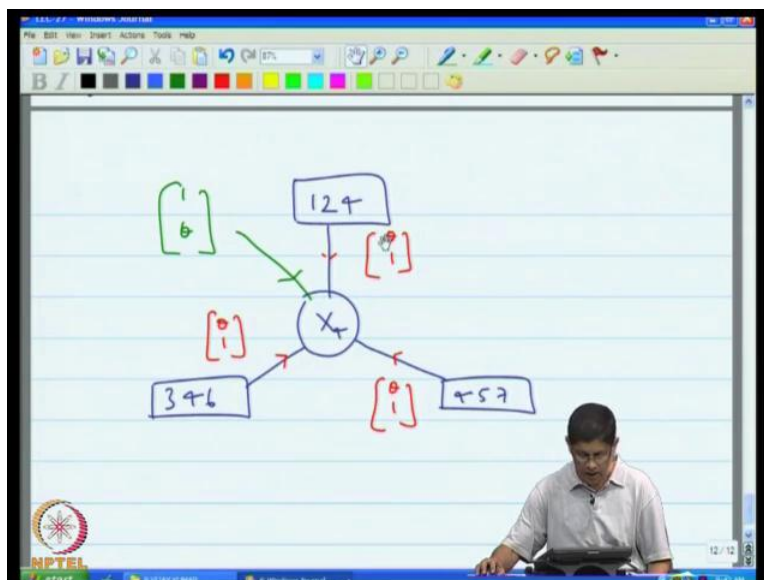
So here, what that would mean is that remember these quantities in vector format 1 theta 1 and theta 1, so you are going to multiply these component by component, and then you are going to take the max. Now,  $x_4$  here dictates that if  $x_4$  is 0, it dictates that if  $x_1$  is 0  $x_2$  is 0, then you would multiply 1 and theta and theta and 1 and take the max if  $x_4$  and theta and 1 and take the max if  $x_4$  on the other hand is a one, when  $x_1$  is 0  $x_2$  is a 1, when this is a 1 that is a 1 when this is a theta this is a theta it takes the max of that you will actually end up taking in this case The max of  $g$  of 0 therefore is the max between one times theta and theta times one which is theta;  $g$  of 1 is the max of and this time coke fast, because one here force one here it is one times one and theta times theta, this is actually equal to 1. So, the message that is passed here is result is theta 1, the message that is passed is theta 1.

(Refer Slide Time: 32:00)



And now, we can go back to this figure here, and fill in all the messages, we seen here that the message that is passed in here let me try to put a different color here; perhaps, I will try red itself, this we would already seen is theta 1, the message coming in here again for exactly the same reason is theta 1, this is also theta 1. And this quantity here is the local kernel. This is the local kernel, and you are going to take the product of all of these.

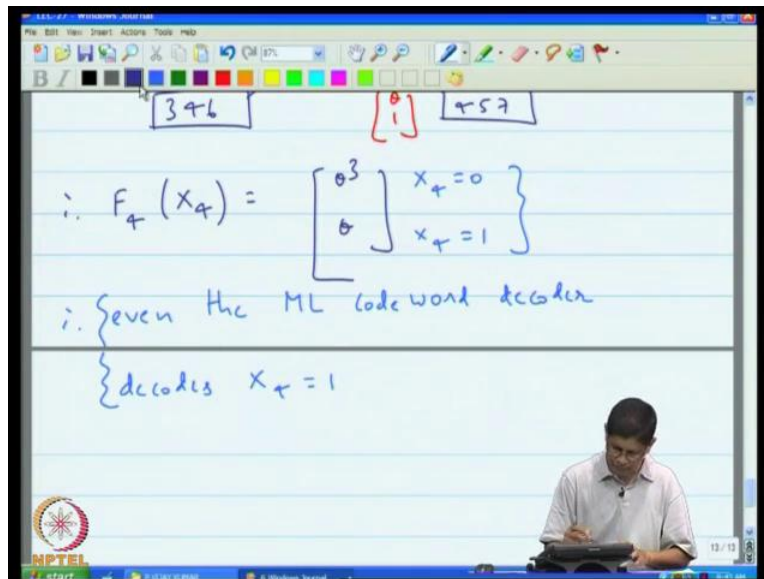
(Refer Slide Time: 33:00)



The end what you going to left with is the following computation; you have x 4 and from 1 2 4, 3 4 6 and 4 5 7 all of the incoming messages are the same and correspond to theta 1. Your local

kernel however is 1 theta, so the way you should interpret this is that it saying that the local kernel seems to suggest that you know likely to be a 0, because this number is larger. However the messages that are coming in from all this check nodes tell you otherwise that telling you are more likely to be a 1, and that is then what do you do? Well, the GDL tells you need to multiply these quantities.

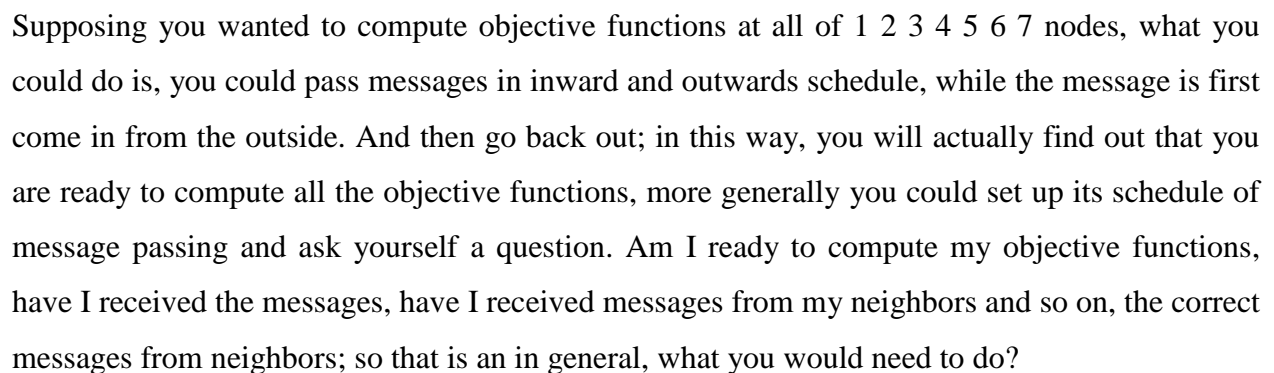
Refer Slide Time: 35:00)



So, in this case your  $F_4$  remember objective function we had called we denoted that by  $f_i$  of  $x_i$  say 4 of  $x_4$ ; therefore  $F_4$  of  $x_4$  is equal to product of all three, which will then be theta cubed and theta. Therefore, since huge compare in decoding what you do is, compare between the two possibilities; the possibilities corresponding to 0 and 1, so again I remainder this here stands for  $x_4$  equal to 0. This is  $x_4$  equal to 1. And when you do the comparison, you see that  $x_4$  equal to 1, so therefore even the maximum likelihood decoder; even the maximum, likelihood code word decoder, decodes  $x_4$  equal to 1; so in both cases end up correcting the error.

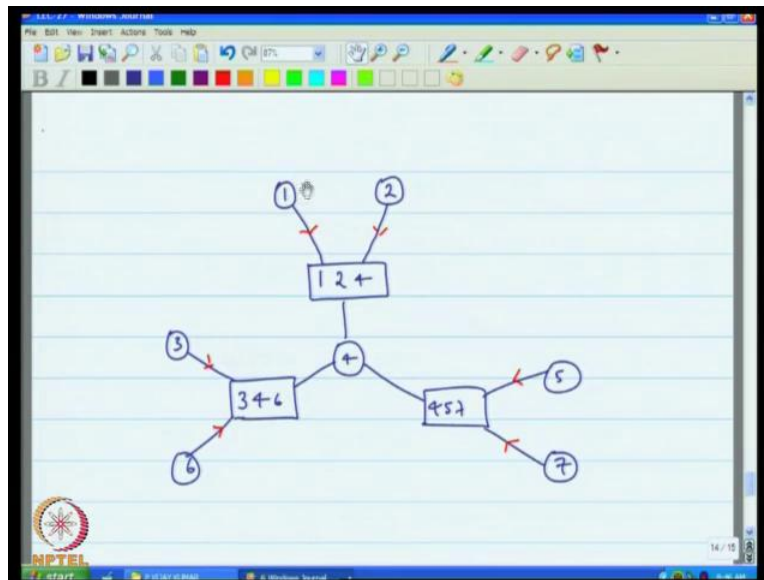
Now, the other question you might have on your mind is well. You tell me how to decode  $x_4$  on this graph, but how what if I have wanted to do decode all the others. Well, this node difference fundamentally, because node is ready to compute his objective function, when it has received information from all its neighboring nodes. So, you can work out whatever schedule you want,

(Refer Slide Time: 36:00)



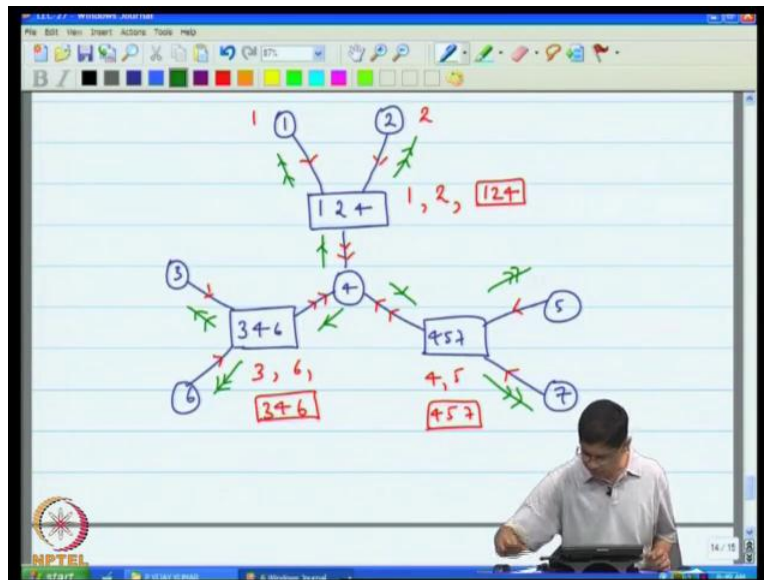


(Refer Slide Time: 37:00)



You could and you could keep track of in sequential message passing procedure, so perhaps I should do this. Let us go back to lecture twenty six. So here, in general if you wanted to know, when it is exactly that a node is ready to compute its objective function. What you could actually see is what inward outward schedule let say that you first pass all messages inward like this; and on the graph, you could keep track of who has heard from whom? So, let me get rid of this right here. So initially, it is true that every node knows its own local kernel, so this knows this has knowledge pertaining to 1, this has knowledge pertaining into 2 and so on. But after these messages have been passed, we can keep track of what who has what messages, perhaps if should put downing red again. So, this knows all information pertaining to 1 and to 2; once these messages have been passed, this has all the information pertaining to 1, 2 and of course 1 2 4.

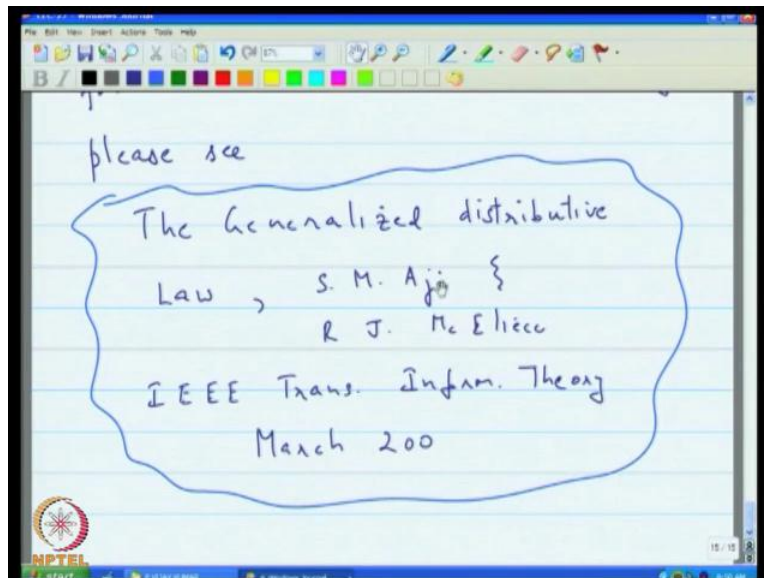
(Refer Slide Time: 39:00)



It knows 1 2 and 1 2 4 likewise this knows 3 6 and 3 4 6, because 3 4 6 is a node by itself, similarly this knows 4 5 and 4 5 7. Now, once these nodes have had a chance to pass on their information to 4. Then 4 has information from everywhere and that is when it is ready to compute its objective function. Now, so that is exactly what we did just now, but supposing at this stage, we were actually to pass messages in the backward direction. So, let us say that next my next step is to pass a message like this, a message like this and a message like this. And you see that I am able to pass a message in whichever direction I choose, because I have heard from all my neighbors me my precision to pass messages in these directions. Now at that stage I have all the information.

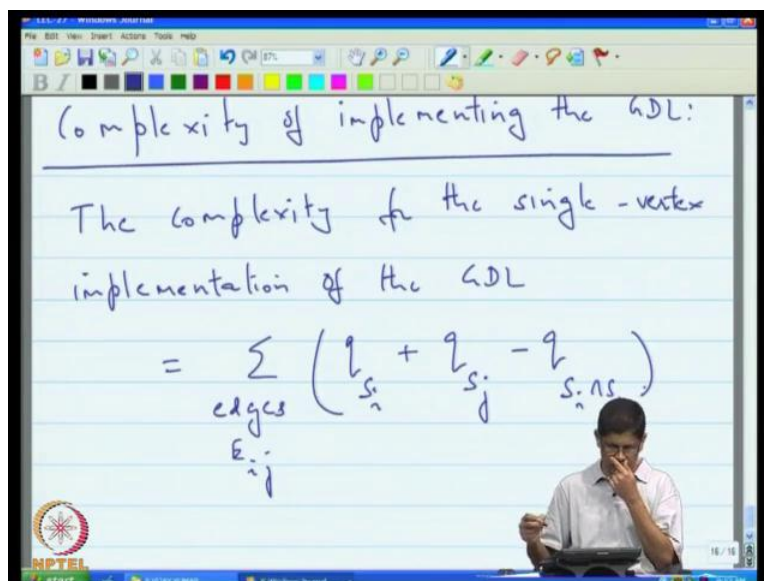
Therefore, I am able to pass all information to these nodes. So, on the backward wave every node is it here receives a message it has it has complete knowledge, this already heard from to this here is from everything else to which its connected. And so, you actually go back and then computed that is on the first and on the second iteration, you can compute the objective function at the leaf nodes.

(Refer Slide Time: 41:00)



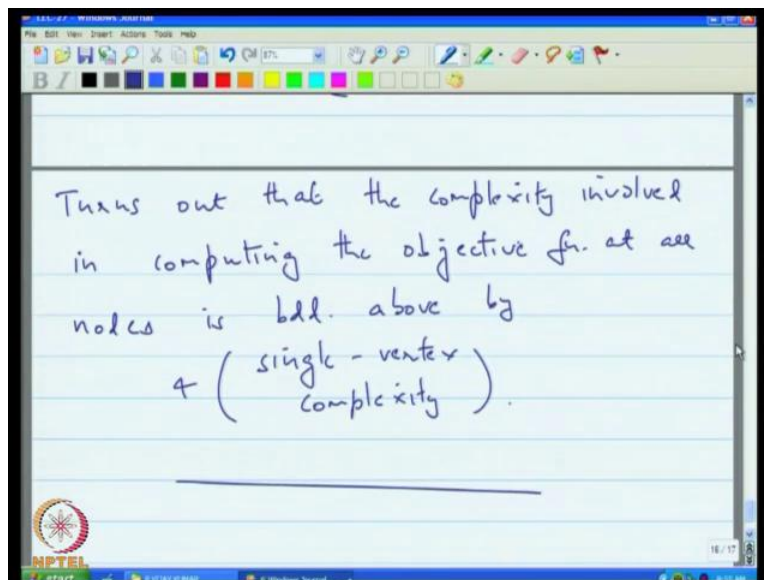
So, you can find out more details about this schedule from the paper, which is our reference for more details on this scheduling please see. The generalized distributive law by S M Aji and R J Mc Eliece IEEE transactions information theory this is march two thousand issue alright with that, although there may be small gaps here in there hopefully you understood. How you can actually use the GDL to decode a code when there is a junction tree.

(Refer Slide Time: 42:56)



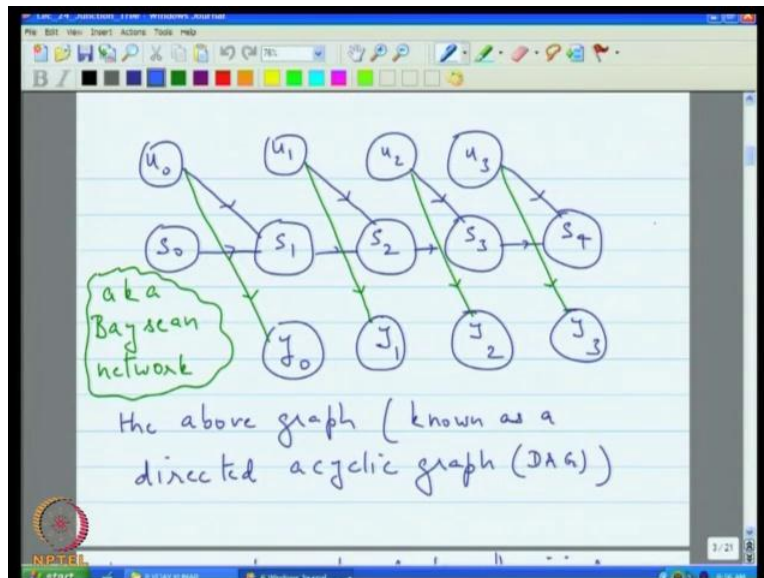
Just a quick note again without proofs, note on the complexity at the GDL of implementing the GDL; this it can be shown that the complexity for the single, a single vertex implementation of a GDL is equal to the sum over all edges  $E_{ij}$  of  $q$  of  $s_i$  plus  $q$  of  $s_j$  minus  $q$  of  $s_i \cap s_j$ . And I mention this theorem earlier and this is why when we were actually setting up the junction tree we wanted keep this as an objective function to be minimized, where an opportunity errors. It can be shown so this is counting additions and multiplications as operations. Again I use the terms addition and multiplications but these really are reference to the two operations in the summing which in the sum product summing turns out to be additions and multiplications.

(Refer Slide Time: 45:27)



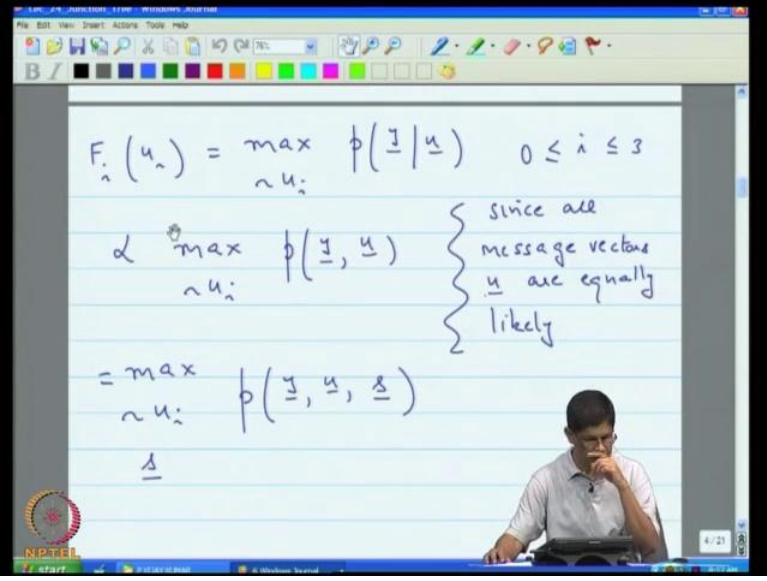
And it turns out that, the complexity involved in computing the objective function at all nodes is bounded above by four times the single vertex complexity. So, it tells you, gives you a reasonable handle on how many operations are required to implement the GDL. Again for the proof, you can actually, can solve the paper I think that will take as no time then we really have to actually go through the proof. So, I somewhat ambitiously put down the title of our lecture, the GDL approach to decoding convolution codes. Let us move on to consider convolution code next.

(Refer Slide Time: 47:14)



So, maximum likelihood code symbol decoding of a convolutional code; and we did actually introduce convolutional codes in the context of the MPF problem in lectures twenty 3 and also twenty four. So, let me go to twenty four here and I actually, we looked at this figure here this actually tells you that in which really is right is a graphical means of actually depicting the factorization of the joint probability function of the message symbols, the output symbols on the state of the convolutional code, and we said that this will factor like this, and that is what this figures is all about this is not a junction tree it is what is called Bayesian network. It is a directed acyclic graph that reflects the factorization and nothing more, and that is actually given here.

(Refer Slide Time: 49:00)



The whiteboard contains the following text:

$$F_i(u_i) = \max_{\sim u_i} p(\underline{y} | u_i) \quad 0 \leq i \leq 3$$
$$\propto \max_{\sim u_i} p(\underline{y}, u_i)$$
$$= \max_{\sim u_i} p(\underline{y}, u_i, \underline{s})$$

Since all message vectors  $u_i$  are equally likely

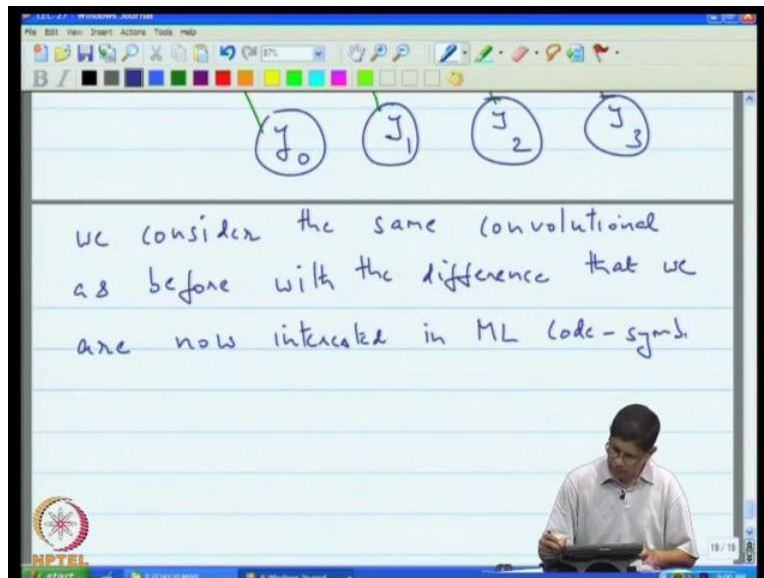
The lecturer is a man with glasses, wearing a light-colored shirt, sitting at a desk and looking towards the camera.

And just as we did maximum likelihood code word decoding at the 7 4 2 code, you can also maximum likelihood code word decoding of the convolutional code in a similar fashion by introducing objective functions such as these, and if you work through it will actually get this, and this maximum likelihood code word decoding is what the viterbi algorithm actually does. However I will what I we will come to viterbi decoding and I actually show you how viterbi decoding can all be viewed from the perspective of the GDL. What I would like to do today however is actually introduce a different decoding technique, which is decoding technique for maximum likelihood code symbol decoding of the convolutional code.

So for that, let me just, so I think that is easier than perhaps to just try it out again simply. So our interest here is in actually in but I should clarify that we will do this for example, and the example view the same one that we considered earlier, namely this convolutional code. What do I mean by this? Well its sufficiently general except that just to keep thing have manageable I have restarted my attention to this four message symbols here; short and in some sense of short length convolutional code. So, we will restrict attention to this perhaps I should reproduce this; so let me say we consider the same convolutional code, I think I am writing too close to the edge of the paper.

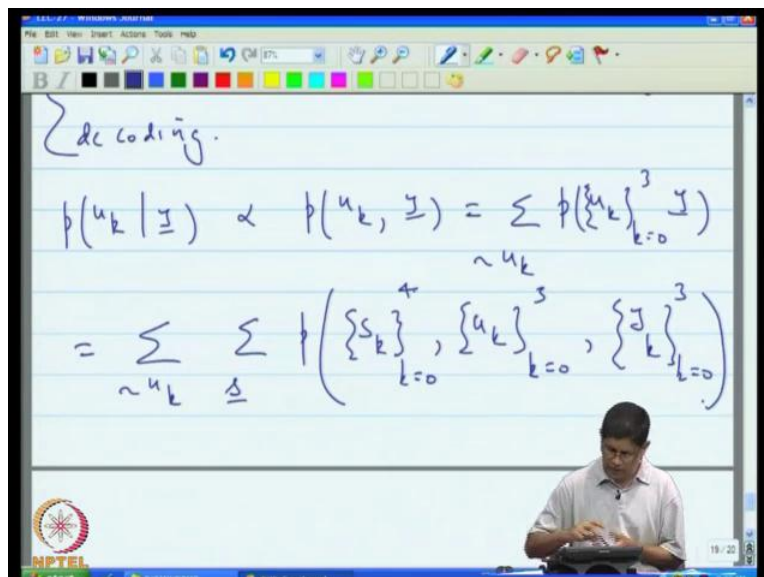


(Refer Slide Time: 51:20)



We consider the same convolutional code as before with the difference that we are now interested in maximum likelihood code symbol decoding, so that leads as to an expression that is  $p$  of  $u_k$  given  $y$ .

(Refer Slide Time: 53:00)



I can write it as being proportional to  $p$  of  $u_k$  comma  $y$ , which is a marginalization of  $p$  of  $u_k$ ,  $k$  is equal to 0 to 3, and  $y$  which is a marginalization with respect to  $u_k$  of and further

marginalization over the entire state space of the code of  $p$  of  $s_k$  the state sequence going from  $k$  is equal to 0 to 3. Our  $u_k$  there are four states excuse me  $k$  is equal to 0 to 3 and  $y_k$ ,  $k$  is equal to 0 to 3. This corresponds exactly to our convolutional code here with there are these message symbols this is the state sequence. We start from state 0 end up in state 4 and these have a corresponding receive symbols and that graph tells us how this actually factors.

(Refer Slide Time: 55:00)

$$= \sum_{\mathbf{u}_k} \sum_{\mathbf{s}_k} p(\{\mathbf{u}_k\}, \{\mathbf{s}_k\}_{k=0}^3, \{\mathbf{y}_k\}_{k=0}^3)$$

$$= \sum_{\mathbf{u}_k} \sum_{\mathbf{s}_k} p(s_0) \prod_{k=0}^3 p(u_k) p(s_{k+1} | s_k u_k) p(y_k | s_k u_k)$$

We will have probability of  $s_0$ . This factors like this and once again we see that we were ended up with a problem that has a form of marginalize a product function problem and this is where since we only have two minutes left this is where we take up our discussion in the next lecture just to recap what we did today was, I started out with the GDL approach to decoding were we want to do maximum likelihood code word decoding at begin with, I just know the couple of nodes, I explain that how would node under the GDL computes its objective function. We completed our discussion of maximum likelihood code symbol decoding. Before moving on to maximum likelihood code word decoding, we made the observation that, you simply replace the summation operator by the max operator and we carried out decoding that was quit straight forward and then towards the end we started discussing decoding of the convolutional code though is also a short note on the complexity of the GDL. Thank you we will see you next time.