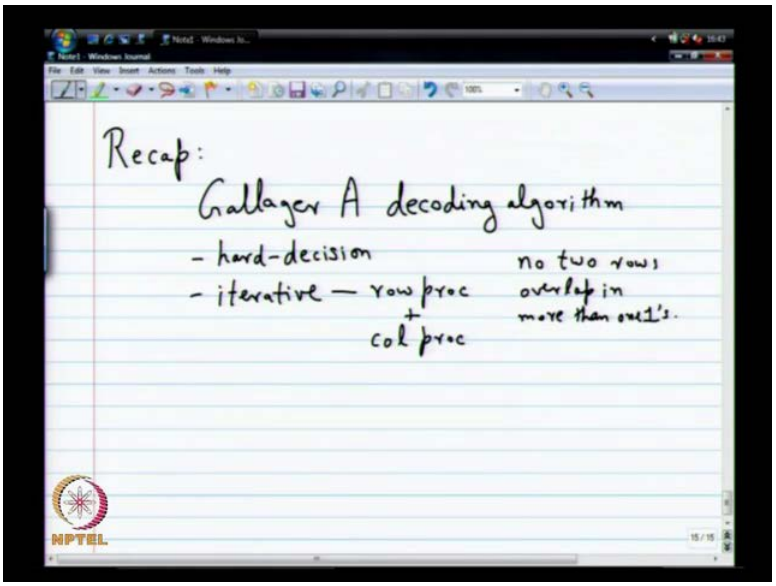
Coding Theory
Dr. Andrew Thangaraj
Department of Electronics and Communication Engineering
Indian Institute of Technology, Madras

Lecture - 24
Message Passing, Density Evolution Analysis

(Refer Slide Time: 0:11)



So, let us once again quickly recap where we were. So, we are looking at the Gallager A decoding algorithm, it is a hard decision decoding algorithm over the binary symmetric channel. So, what are the various features if you think about the way it is working, it is hard decision decoder. It works over the binary symmetric channel, it is iterative the iterations are in two steps. First step is row processing, each iteration is split into row processing and column processing. The basic idea is to use the rows of the parity check matrix that are connected to a particular bit to get estimates of that bit. So, there is an incentive directly from this algorithm to make sure that no two rows overlap in more than two places in 1's, right more than 2 1's right.

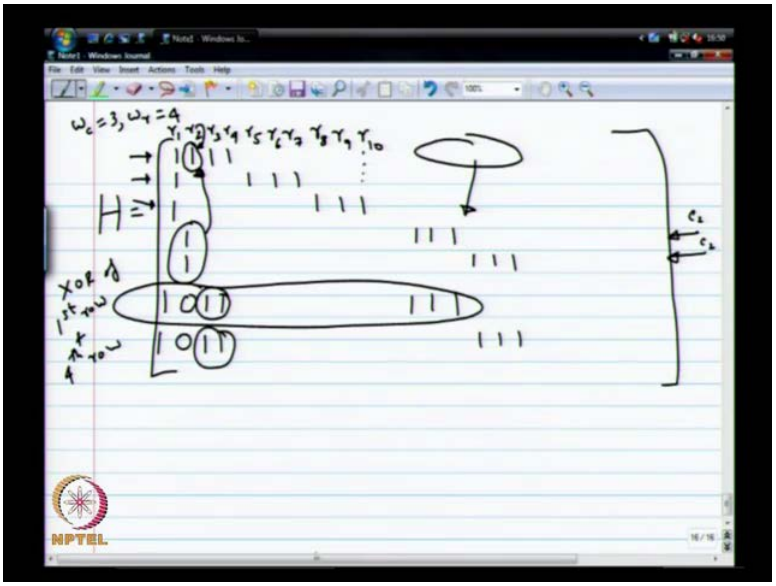
So, if you do that you see that your row processing information is independent at least for the first iteration. So, beyond that it is, it just becomes a little bit more complicated to figure out all that was involved in a particular estimate. So, because what you do is you iterate on this estimation, you first get some initial estimate then you use those better estimates to re derive the estimates and you keep continuing. So, it becomes a more

complicated process to visualize what is happening, iteration after iteration. So, I will make brief attempt at it and then we will move on to the tanner graph, yes.

Student: More than 1, should there be 2?

More than 2, more than 1 you said, more than 1 I am sorry more than 1, is that correct? So, I will, let me just briefly show you one picture with a parity check matrix to see how this, how multiple code words of the dual, more code words of the dual than that are there in the parity check matrix itself are being used at the decoding, just to illustrate real quick. And then will move on to the tanner graph description, which is, which is much more standard and much more useful, interesting.

(Refer Slide Time: 3:03)



So, let us, let us look at the parity check matrix and maybe I will rearrange the rows so that, the 1's of the first column show up in the beginning and maybe the 1's that are here, I will use, let us say w c equals 3 w r equals 4, just to make it little bit more. So, everything else is 0 so, it is a huge matrix ((Refer time: 3:32)) some very big matrix. So, this could be, this could be one way in which we could permute the rows and columns and make bring everything up here. So, I can do this there is no problem. So, what happens in the first iteration, you get an estimate for. So, this is of course r 1, r 2, r 3, r 4, r 5, r 6, r 7, r 8, r 9, r 10. So, r 10 comes...((Refer time: 4:00))

So, in the first estimate r 1 the received values involved are r 2, r 3, r 4 and the code word of the dual involved is simply the first row. In the second estimate the code word that is involved is the second row. In third estimate code word that is involved is the third row, what happens in the next iteration? So, it is a bit more difficult to imagine, but let us say I will look at the column corresponding to, I will look at the second column corresponding to this guy. Maybe I put the 1's,1's write here, right and may be the remaining 1's for this, maybe I mean I will just draw it far away. I mean it could, it could overlap maybe with here, but I will draw it far far away so that, we see it a bit more clearly. 1, 1, 1 and then next one is will be 1, 1, 1. So, these are additional values here.

Now what would happen in the first iteration for c 2, what would have happened? It would have got estimates from the first row and then, yeah 4th and the 5th row. This is for c 2, c 2 in the first iteration, what is happens for now c 1 in the second iteration? Imagine what is happening see so, you are going to update that depending on the, these two things. So, the estimate for r 2 now so, c 2 that you got in the first iteration, they were three different estimates. One from the 1st row, another from the 4th row another from the 5th row, when I update the estimate for c 2 that is going to be used in the second iteration, in the first row what will I do? Based on these two guys, right these two guys will play a role and r 2 also will play a role, alright.

Now when this updated value estimate is being used in the second iteration for c 1, what is actually happening is indirectly you are using the fact that is the first row x or with the fourth row is also a code word of the dual. So, what is that thing that you are using 1 0 1 1, which is the x or of first row and fourth row, right. The second estimate that you had, the estimate from this row, from the fourth row that went into r 2 will make its way to r 1 and that is almost like, I am sorry 1 0 1 1 that is also a same word. So, it is almost as if you are getting the estimate from this code word, but it is happening indirectly in an iterative way. The first round you only get estimate from fourth row for r 2 and then in the next iteration, you are using that new estimate in updating the estimate for c 1.

So, indirectly the code word that is involved is 1 0 1 1 1 1 1 notice this 0 here is crucial so that, you do not use the same r 2. So, you cannot use it again so of course, in the hard decision decoder it is not very easy to make this connection explicitly. In the soft decision decoder you can make an explicit connection like this, we will see a soft decision decoder later, there you can make a very explicit connection between these x or

code words and what is happening in the iteration. But essentially this is what's happening so, you are using more code words from the dual indirectly through the other bits that are getting updated in each row.

You are using this row, but then the value is used for those rows, the other bits are getting updated through other code words and essentially are using the x or finally. So, this is what I briefly meant when I said even though you are doing using only the rows in the parity check matrix explicitly, there is implicit use of xors through this connection and think about it, it is, it is, it is, it is kind of a loose connection at this point. Maybe when we do soft decision decoding later, we will write down some explicit expressions, at that point you will see this code word is also involved, is that fine. Think about it, it is a bit rough I know it is, it is, but it is important to keep in mind, yes.

Student: Did Fourth and fifth row affect together also, it is not x or of first and fourth row.

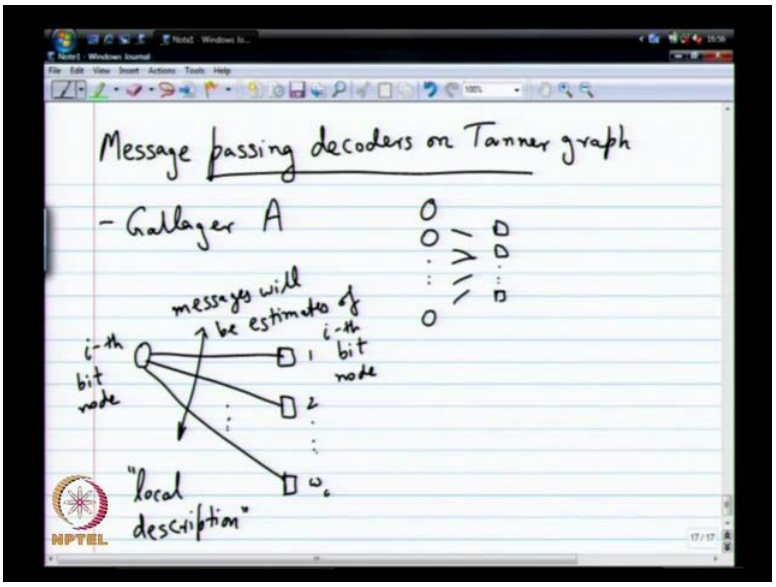
Exactly, so this, this other code word is also involved you are right, yes. So, all these code words are involved in that decision, that is what that is all, I am saying. So, all these other code words are indirectly involved through r 2. So, when you write the soft decoding expression you will see every every extrinsic information will get added. So, it will be much clearer. Here we are not writing any soft LLR, right so, it is only just estimates. So, more and more code words are involved that is all, it will not actually be 1 1 I put 1 1 here, but this is a bit questionable because, again you are using more code, more code words that r 3 is connected to, right.

r 3 is connected to two others, those also are giving some estimate, that is what they are actually using, but r 4 is connected to other so, using a lot of things at the same time. So, even though it looks like in the first iteration I am only using very small number of code words, the number of code words I am using eventually from the dual just increases, in fact exponentially with iterations. And as you do more and more iterations, you are using more and more extrinsic information and all that gets added up. Of course, the independence assumption is crucial we are using the independence assumption, it may not be valid, but even then it is something.

So, this is a brief idea with the tanner graph this will become a little bit more clear. So, maybe for now it is a bit more bit hazy. So, now let me move on to the description using

tanner graph. So, the algorithm that I described Gallagher's A, Gallagher A algorithm of course, Gallagher invented it, he came up with it in his PhD thesis without any reference to the tanner graph. So, all these ingredients were already there, how you should not use this information from the same row in the next iteration etcetera, all that was crucially there.

(Refer Slide Time: 10:37)



So, now the modern way of describing this algorithm is of course you think the tanner graph, there you use this notion of what is known as message passing decoders on tanner graph. So, the same Gallagher A algorithm can be interpreted as what is known as a message passing decoder on a tanner graph. So, all this confusion I had in describing to you the estimate of this based on that. And then, in you have several other estimates for each thing, you have different estimates etcetera etcetera, can be very nicely phrased in the, in the tanner graph language and you will see is very simple. All you have to do is describe is what are known as local operations, then the global operation kind of gets defined automatically without any, without any problem.

So, that is what is nice about the tanner graph you can have some local view, global view etcetera. So, we will we will write it down will analyze it we will finally, slowly get to the main result. So, what are these message passing decoders? So, let me just, let me have the right page up there, once again remember its BSc and the received vector is over a binary symmetric channel and we are doing hard decision decoding. So, I am only

going to describe Gallagher A right now. So, you might there are, there are, there are of course, message passing decoders are very general. There are several ways of doing it, but what am I going to describe is the Gallagher A decoding algorithm, has a message passing algorithm.

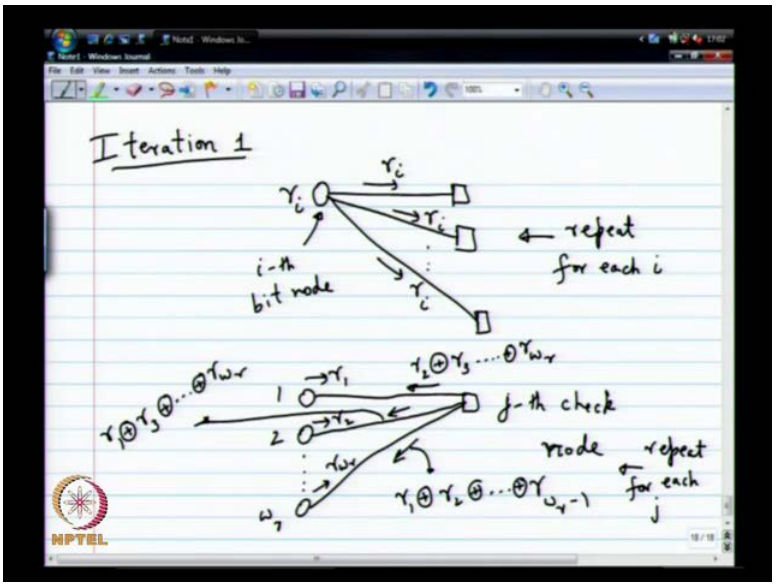
So, remember the parity check matrix I drew has a tanner graph representation, it has bit nodes n of them and it has check nodes as many as the number of rows and all kinds of connections. So, if you, if you think of what is happening in the in the in the first step in the Gallagher A decoder and multiple iterations. Essentially what is happening is, you have a bit node and it is connected to w c check nodes, right. So, you have a particular bit node and let us say the ith bit note, it is connected to w c check nodes, right. Any processing that happens, the row processing or column processing, row processing for the ith bit node for instance, involves these w c check nodes, right. And the column processing for the ith bid node also kind of involves this same picture, right.

So, it is not, is that correct, is it clear, you see what I am saying. So, essentially all the estimation and iteration that is happening is mostly local, it is around this bit node. You do not really do any operation that is far away from the bid node, you only see closed to the bit node and use the rows directly connected to it, by updating these messages and then passing, then passing, passing them around. You indirectly use the remaining part of the graph so, directly you use only the local things. So, you will see the entire Gallagher A algorithm can be given a simple local description, through this message passing idea.

So, we will always think of messages flowing from bit nodes to check nodes, then there will be that flow from check nodes to bit nodes. These messages will basically be estimates of the ith bit node, that any message that flows on this will be estimates of ith bit node. So, this crucial idea and a message passing interpretation which is, what happens even in the Gallagher A decoding algorithm. You are mostly concerned with estimates for a particular node and when describe it in the tanner graph language, we will talk of messages that flow from bit node to check nodes. And messages that flow from check nodes to bit node, any message flowing out of the ith bit node will be an estimate for the ith bid node, ith bit. Any message that is flowing into the ith bid note will once again be an estimate for the ith bit node.

So, that is the main idea to remember in this description, when I describe it if you remember that idea many of the things will be very simple. You can also describe it the matrix language, but it is a little bit more involved so, best thing is to do use the tanner graph, is it alright.

(Refer Slide Time: 15:40)



So, let us first do the first iteration, iteration 1, it is a little bit special because you do not have any extrinsic information. I will draw the ith bit node okay, this is the ith bit node, remember this is the first iteration, the only thing, estimate I have for the ith bit is r i. So, this r i is simply associated with the ith bit node and now this guy is connected to, let us say as many check nodes as you want. In this case we have been looking at w c check nodes so in the, in the first half of the row processing step or let me say, may, not let me not say that. In the, in the first to initiate the decoding some messages to flow in this tanner graph, right.

So, you have estimates for bit nodes on the left side which is just r i or if I want to simply pass this message or pass this estimate out to the check node. What is the only thing I can do from the ith bit node? I can pass r i so, what will happen in the first iteration is simply r I will flow out of each of the ith node into the connected check nodes each of the ith bid nodes to the connected check nodes. So, it is a very local simple description of message passing or estimate passing from one side to the other. So, this is like the first half of the

first iteration where, messages are flowing from the bit node to check node these are estimates, alright?

Now in the second half, what will happen is you have a check node let us say the ith or jth check node, the jth check node will now be connected to how many bit nodes? There will be w r bid nodes, right. So, it will be connected to all of that and let us say just to make the notation simple, I will number these bit nodes as 1, 2 all the way to w r. It will not be numbered from 1, 2 to w r of course, it will depend on j right. So, if you want you can call it 1 of j, 2 of j etcetera, but people do not like such things or may be j 1, j 2 etcetera, whatever you want to call it. But to simplify notation I will simply say 1, 2 all the way to w r these are the bid nodes that it is connected to. What would have happened in the previous step?

Yeah, r 1 would have flown from, would have not flown, would have been passed from 1 to this check node, r 2 would have been passed from this thing and so on, right. So, this is what would happen in the first iteration, r w, r would have passed from this guy to this guy, is that okay. So, what will happen in the second step of the first iteration is the check node will now pass back estimates for each of these bits. So, remember what has to flow on this edge, the first edge here it has to be an estimate of bit 1, it cannot be an estimate if bit 2, 1 has to flow, what about this edge? Estimate of bit 3, but this check node now has two different estimates for bit node 1, what are the two different estimates?

One is r 1 itself the other is x or of all these other things r 2 xor r w r, what do you think it should send back? It should send the xor, no point in sending r 1 back, right r 1 already bit 1 has why, why, why will I send r 1 back. There is no reason why you should send this, so that is some kind of, that is what is known as extrinsic principle in message passing. Whenever you receive something and whenever you want to send something back, you do not repeat that information, you do not resend that information that you got from a particular node. So, this is kind of a principle that is followed in most message passing algorithms in general and that is something we will again repeat here and it is very similar to the estimate that we did in Gallagher A matrix description also.

So, what will be passed back here is r 2 xor so, maybe this arrow is kind of misleading, what will be passed back in this is r 2 xor x 3 xor r 2 xor r 3 so on till r w r. What will be

passed here in this direction basically is r 1 xor r 3 xor so on till r w r. What will be passed here is r 1 xor r 2 xor so on till r w r minus 1. So, the nicest thing about this tanner graph picture, you can see what is nice so, all you have to do is give local descriptions, you do not have to remember this huge matrix and worry about what is happening at the same time everywhere.

And simply say, you simply, you give the local description and say you repeat this vary check node, it is very easy to do, right, the first step you repeat for, here you repeat for each j. So, that is enough to specify what is happening here and that is exactly what happens in the Gallagher A decoding algorithm also. So, kind of, the first step maybe was little bit hidden so, I did not state it explicitly, but this is what, this is what you do in the first step, is it clear, any doubts or questions,

Student: ((Refer time: 21:46))

x or of everything to what?

r 1 to r omega.

Oh, you sent back to 1 and like 1 figure out the, xor subtracting the r 1's, yeah you could do that also. So, of course so it is, it is point is where do you want to send the w r different things all you have to send is r 1 xor r 2 back, yeah it is perfectly correct and then let the each bit remove its own thing and then figure out what, that is also perfect so that is fine. So, those are ways of implementing this in a more efficient way, that you can. So, now comes the description for iteration l so, I will give one description for iteration l which is a general iteration, you can think of l equals 2, 3 so on. For 1 of course, you have to use the previous description, for iteration l you have to give you have to do, this is a description that I want to give.

So, you have so, so, what has happened before. So, if you look at the particular bit node, the ith bid node it has its intrinsic information r I, it has connected to w c check nodes, which I will, for convenience number simply as 1 to w c. It is connected to that, what would have happened in iteration l minus 1 at the very end messages would have been received from each of these w c check nodes. So, I need some notation for it so, I will simply denote them as, cannot remember if you used, if we should use u here v here, what should I use here? I should use u so, I will use u so, u i l minus 1. So, I do not need

I’s, it is just, it is the ith node so, simply say u 1 l minus 1, u 2 l minus 1 so on u w c l minus 1.

So, those are estimates that would have been received in the, from the previous iteration right, I can always assume that some w c estimates are being received, each of them remember r estimates for the ith bit. They are extrinsic estimates, in addition I also have the intrinsic estimates r i, what I have to figure out is, what I will send back. So, what is send back maybe I will denote it as v 1 l v 2 l v w c l and how do I figure out v 1 l, I will tell you how I figure out v 1 l. If u 2 l minus 1 equals u 3 l minus 1 so on till u w c l minus 1 equals some b then, u set b 1 l to be equal to b, else v 1 l equals r i ,that is the rule for b 1 l, what will I do for v 2 l? If u 1 equals u 3 equals so on till u w c equals b, then you set v 2 l to be equal to that b.

Otherwise, you set v 2 l to be equal to r i, that is it. So, as simple as that, it is the local description of course, you repeat this for each i. I am writing it only for v 1 so, hopefully everything else is also clear, everything else you have to do the same thing. So, once again the principle is, we are following the principle the principle is to not send known information back to the same node, you got u 1 l minus 1 from the first check node. No point in using that in sending anything back. So, you do not do that because, you have to maintain the extrinsic nature of all messages that are passed, you do not pass known information back, it only cost some kind of feedback, negative, positive feedback and blow up thing, etcetera.

So, you can use the systems description to motivate that. So, hopefully that is that is clear, this part is clear. The next step what happens, you have to now look at it the j jth check node, which has w r bit nodes connected to it. Let us say this is the jth check node so, it is very simple, and the check node operation is much much simpler here. This would have been, this will receive so with some major abuse in notation ,I will simply say this is v 1 l, this is v 2 l, this is v w r l. So, of course,, the index is not the same, you do not have to worry about which bit node is connected to etcetera etcetera, just for simple description I will simply say, the messages that are received from the corresponding bit nodes is this.

What are these messages once again, remember these are all estimates for the corresponding bit. So, v 1 l is an estimate for bit 1, v 2 l is an estimate bit 2 etcetera

okay, that is been relieved in the lth iteration. Now, this thing will have to send back u 1 l, u 2 l so on u w r l, what will this be, what will u 1 l be? It will be the xor of v 2 l, v 3 l so on till v w r l. So, likewise for u 2, u 3 and all that u 2 will simply be v 1 l xor with v 3 l so on till v w r l.

Student: ((Refer time: 28:51))

So, then that is Gallagher A prime, maybe it is Ashwinram A, something. So, this is Gallagher A so of course, I mean like he is pointing on, his question is can you not change any of these things. Of course, you can change any of these things, as long as you can get good performance for the decoder, that is correct and there are all kinds of research. Believe me in the last 15 years there have been 10,000 papers on this. So, every single thing has been tried so, very unlikely that you can try something new, I mean unlikely only, still possible. But unlikely that you will come up with something new, so many researchers have lived at it, it is in the standards, right.

So, people have tried everything literally yeah, but of course, you can try variations here. So, this is the rule that it is called Gallagher A, any questions? So, let me just ask general question now, which one do you like better the matrix description or the tanner graph description. Tanner graph description is much more natural simple and it is, it is easy to implement for instance, so many things we can do with the tanner graph and it is more difficult to visualize what is happening in the matrix. So, this is much be better.

Student: ((Refer time: 30:16)) convergence to a particular code word.

So, that is the next question, next question is about analysis it is saying we described the decoder how do, how could do it be, what does it do for instance, it is not even clear what is does. But it seems to be doing sensible things locally, will globally anything good happen? I mean it is an important question, right. So, you have to ask that question, there are partial answers, partial very very successful answers, these answers can be used for designing codes and they work very well. There is no complete answer, that I can tell you very easily and anybody can tell you, there is no complete answer saying these are the received errors that can be corrected, these are the errors that cannot be corrected, no such answers.

And since the partial answers are very very successful not many people are looking at complete answers also. Anything else, any other question, alright. Now we will slowly move towards analysis, but let me reiterate once again in a Gallagher A message passing algorithm, messages are flowing on the edges between the bit nodes and the check node. Any message that is flowing on an edge is an estimate of the bit node that it is connected to, in both steps right whether it is the bit, whether the message is flowing from bit node to check node or from check node to bit node, it is always an estimate of the bit node that is connected to. Of course, there is no point in estimation the check node because, it is anyway it is not, it is not transmitted right.

So, it is not, it is not desired so always estimate the bit node, that is one thing to keep in mind. The other thing to keep mind is the extrinsic nature of the messages that are passed. So, you, as much as possible locally at least you avoid resending information from the node that you received, locally you avoid it. Globally of course, there is no way to avoid such things, right. Finally, that message would have made its way through some other loop and finally come back here, you cannot avoid that and you do not care about it, you do not care about it. Locally you avoid sending back information so, I will try and depict that in some plot here. So, that is the other thing.

The third point if you want to look in terms of code words of the dual, right, if you want to think about code words of the dual, the way, the what, the message passing algorithm is doing is it is specifying some kind of a sequence in which you process code words of the dual. So, that is what it is doing at the end of the day of course, you do not process them ideally, you process them approximately assuming they are independent. But it gives you a sequencing of the code words by which you process the code words of the dual, right. Sequencing of the code words of the dual by which you process for information and it is not the same sequence for every bit, right. Every bit depends on its neighbors and then it depends on the neighbors there, so on.

So, the neighbors play a big role in defining which, how the code words are processed. So, for first iteration the immediate neighbors bit node, what do I mean by neighbors? Yeah so, there are, the nodes that they are connected to immediately. So, only those nodes are involved in the first iteration and then the neighbors of the neighbors are involved in the second iteration. Then their own neighbors are involved so, you are exploring this entire tanner graph, one neighborhood at a time and then o to the

neighbourhood, neighbourhood of the neighbourhood, neighbourhood of that neighbourhood. And you are assuming there will be no dependence in any of these neighborhoods, as you keep proceeding.

Of course, there will be I mean there will be might be some dependence, then you will make error, but you will live with that. So, fine I make errors I do not care we will see how it works. So, that is the general idea. So, one thing I will strongly urge you to do is to write some simple MATLAB programs, it does not take more than lines for any of these programs. One program is to generate a 3 comma 6 for instance matrix for a given block length n, for instance if 6 divides that n then, it is very easy to do the Gallagher construction. It is almost trivial to do it, just pick an arbitrary permutation try your best to avoid overlapping of rows, but if you even if you cannot avoid it, let it overlap it.

It does not matter, you just have one realization of the parity check matrix, it is very important unless you do this you will never believe it. Because, I will never be able to show that it really works, it can only give you a partial results, based on that if you want to really believe it, you have to work with it. So, you have come up with parity check matrix then, what can you do? You can simulate the b s c very easily, write its very trivial to write a MATLAB program simulate the b s c and then you run this decoder. This decoder is very easy so, the fact if you remember the parity check matrix, all messages correspond to positions of 1's on the parity check matrix, remember all messages are being exchanges on the edges.

So, all you have to remember is this big sparse matrix then all your messages will be on the sparse matrix, it is only those locations will be passing messages. So, it is very easy write the program, you can write within like 10 lines in MATLAB and wrong this decoder keep iterating it, change your probability of error and see what happens to the probability of error after your decoder, right, you have to do it. Unless you do that, you will not really learn how these decoders are working how they make errors what is going on etcetera and unless you do it is very hard, is it alright. So, of course I have done this before.

(Refer Slide Time: 36:16)

So, let me show you what I got or maybe I should import it from there no. So, that is a better idea so, anyway you can say it here. So, can you see it, see it properly, can you see it clearly? So, there are different colors so, how do I do full screen, it is play slide show, it is that full screen, will that do that? What is this, see pause it, but there is only one picture no, it will take some other picture from the same directory is it? You have to be very careful with the windows you know, alright. So, this is what happens if you try and do what I said, you generate parity check matrixes from the Gallagher ensemble I think, I do not if I did Gallagher.

I do not think I did Gallagher ensemble, I did on socket ensemble here of course, I also took at to avoid overlap of rows in more than one position. I have done both for these things and then that is 3, 6 regular, what is plotted on the x axis is transition probability of b s c. So, as you go to the left it becomes, channel becomes better and better your performance should improve, y axis you have bitterer rate. I do not know if you can see the numbers very clearly, it is font is not very big, but it is logarithmic, it starts with 10 power 0 and then goes to 10 power minus 1 etcetera.

So, is the x axis 10 power zero is all the way on the right, then you have 10 power minus 1, 10 power minus 2, 10 power minus 3 so on and there are different colors. First of all there is this vertical line which is marked threshold 0.04, I will describe that later, I will show it tell you what happens there later. But there are other lines, there is a blue line

which is for n equals 30, I do not know if you can see that legends up on top very clearly. It is n equals 30 and then the red line is for n equals 100 and then you have the green line, which is n equals 1000 and at that point you can see it is, it is quite good, I mean it is not, it is pretty good code.

When you have 10 power minus 2 of transition probability, you are getting bitterer rates down to close to 10 power minus 6. So, that is it is not bad and when you go to 10,000 it is even better and there is this threshold, which is analysis, which you can approach if n becomes technically infinity, but large enough. So, even for 10,000 it is fairly close, I mean it is come very close to the threshold and we will see how to compute this threshold. It is quite easy to do that, we will come up with, easy to compute maybe not very easy prove, but very easy to compute. We will, we will, I will show you how computation is done. So, plot like this I would have urge all of you to recreate so, you will really learn LDPC codes if you do this, at least for the b s c and then you can slowly modify other things later on. Any questions?

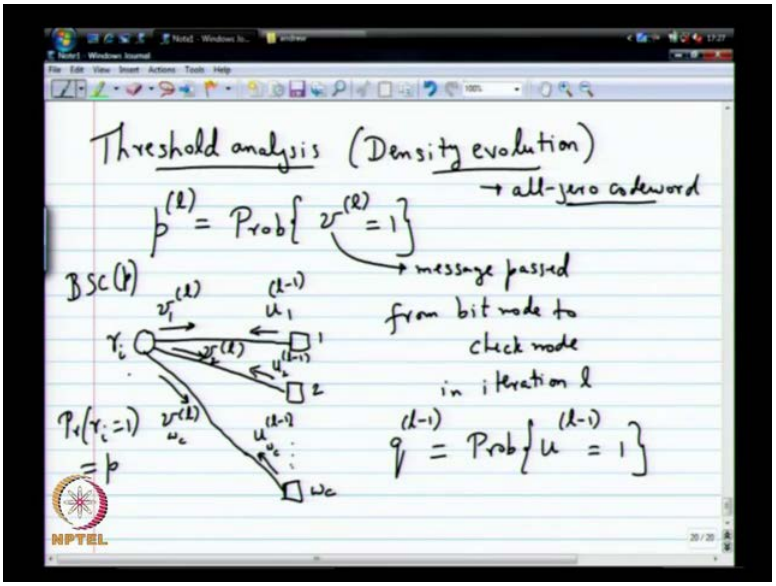
Student: ((Refer time: 39:38))

Well it is, the question was if the algorithm is for a specific channel. So, it is a hard decision decoder definitely so, I guess b s c is a natural assumption, but if you put a w g and then, you will have to run the slice of first and then do this. So, you will soon see soft decision decoder also, but this is the hard decision decoder and so it works. No questions, happy? So, this is, this is what it does, quite well. So, these remember, this is a rate half code, rate half code what is known as capacity for b s c is 0.11. So, what it means is if you have a rate half code for large enough n, any p less than 0.11 you can get technically easy row probability of error, very close to 0 probability of error.

So, 0.11 well 0.11 be it is this guy, right so, roughly 0.11. So, that is 0.11, this threshold is 0.04, it is saying basically below 0.04, I am going to do well. So, it is a little bit aware from capacity and it is difficult to actually build LDPC codes which will get very close to capacity for b s c. So, even after so much research nobody has shown you can get really really close, but you can get quite close with other kind of LDPC codes. These are regular LDPC codes, if you do irregular you can get a bit more to the right, right half. So, what we are going to see next maybe very quickly is to how to compute this 0.04 for a Gallagher A decoding algorithm.

So, how do you compute that, that number threshold like I said the threshold has proved to be a very very valuable, valuable calculation in the design of LDPC and simulations, I mean in optimization of LDPC codes in getting good performance etcetera. But there are some limitations in its definition, in its definition there are some assumptions and you should not be discouraged by because, it is a very valuable too and is really driven the development of LDPC codes. So, main idea here is you will define so, you remember this is my general description of the decoder, I will try to track the probability that a particular message that is passed is an error. So, what do I mean by error so, first thing I will assume is l 0 code word was transmitted.

(Refer Slide Time: 42:29)



So, that will be the first assumption in the analysis so, this is threshold analysis. So, it is also called density evolution. So, I think density evolution is probably a better description so, I will define. So, the way I have defined it here I will define p l as the probability that v l equals 1, remember what is v l? Maybe I should draw that, maybe I should capture that and draw it here. So, this guy is v l it is the message, message that is passed, this is a message passed from bit node to check node in iteration l. Of course, it is a random variable it denotes that message and I am interested in the probability that v l is equal to 1.

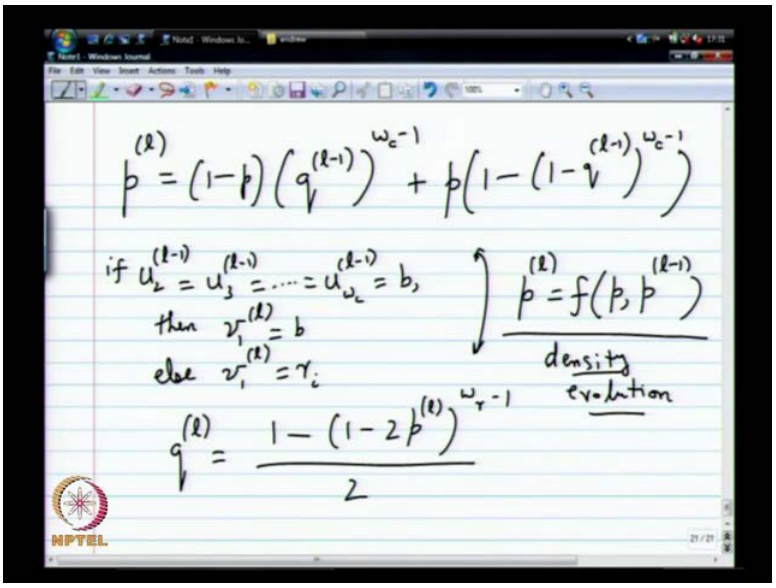
Remember my assumption is all 0 code word was transmitted, I have assumed all 0 code word is transmitted so, this probability some kind of a probability of error. For instance,

if I show that this probability is 0 then what happens? All the messages v that were passed are 0's so what will happen? Remember in the next step of the iteration, if all these things are 0, everything that is passed back is also 0 and I would have de coded to correct code word. So, it makes sense to look at this probability in the all 0 code word case, probability that v l is equal to 1. So, of course it depends on, it depends on the probability that r i is 1 and it also depends on the probability that the u's are 1.

So, what I will do is I will define a q l or let maybe q l minus 1, which is the probability that v l, u l minus 1 is 1. So, I wanted look at the probability that v l, v l is 1 of course, that will depend on probability that these guys are 1. The crucial assumption here is that u 1 l minus 1, u 2 l minus 1, u w c l minus 1 all of these guys have the same distribution as the u l minus 1. And they are independent, that a pretty important assumption, as s is all 0 code word assumption, both of those are important and all these can be justified. I will maybe not be able to fully justify it, only partially justify, in fact it can only be partially justified, but it is, it is still an useful tool. So, we will assume all that, is that clear.

So, I will think of a v l which is a random variable, which denotes the message passed from bit node to check node and iteration l and my assumption in the analysis will be v 1 l v 2 l v w c l all of them are IID according to v l. Same is the case on the other direction, u l will be the message that is passed from check node to bit node in iteration l and whenever I have u 1 u 2 u w c, they are all IID according to that same distribution. So, q l minus q is the probability p l is that probability, remember we also have the transition probability on to channel p, BSc p is that, that is the probability that r i is equal to 1, that is p. Now, I have the rule for finding v 1 l from u 2 l minus 1, u w c l minus 1 and r i.

(Refer Slide Time: 47:01)



If I look at the rule very closely and analyze it, I will get the following equation for p l in terms q l minus 1. So, I am not going describe a proof for this it is, it is quite easy but, it is, it has involved some thinking, but it is not too difficult, think about it and work it out for yourself, only then I think it will be very clear to you. So, you can you see based on a rule, what is our rule? If you want I will reproduce that rule for you. So, that is the rule, if all of these guys agree to a particular value then that will become that value. Else, we will set it to r i, you can you can look at this rules stare at it very closely and you will get this inequality, this equality.

So, go back and think about it, I am not going to describe this in much more detail, it is a simple probability computation you can think about this, is it okay? So, you have got p l in terms of p and q l minus 1 or you have to get q l minus 1 in terms of p l minus 1 and that is not very difficult because, that is simply a xoring operating. I am going to assume it is independent so, you can very quickly show the following inequality that q l will be equal to 1 minus 1 minus 2 p l raised to the power w r minus 1 divided by 2. This you can show just based on the xoring property. So, this is the basic density of illusion analysis.

So, you have random variables which denote the messages that are being passes and you track the probability distribution of that random variable from iteration to iteration. So, you evolve the density through iterations so, it is called density of illusion. These are the

two formulas for Gallagher A of course, if you change the Gallagher A rule, this formula it will change. So, clearly that will change, density evolution depends crucially on the, on the algorithm you are using. Of course, the all 0 code word assumption and the independence assumptions are vital, alright. You have to prove that and there are proofs maybe I will hint the proof later, but right now let us not distract us with those issues, this is the equation.

So, essentially what you have done by combining these two is you can write p l as some function of p n p l minus 1. So, this kind of an equation is called the density evolution equation and that is something we have derived for the, for the BSc. So, some function f l sorry is that the question?

Student: ((Refer time: 50:33))

Yeah, so I am going to talk about it. So of course, we ran out of time for today. We will pick up from here tomorrow and figure out what happens to this iteration, so the what happens to this iteration? So, that is, that is what will give you the threshold 0.04 is given by looking at this iteration very very carefully. Tomorrow is 4:30 there is an industrial lecture. So, 4:30 we are meeting tomorrow.