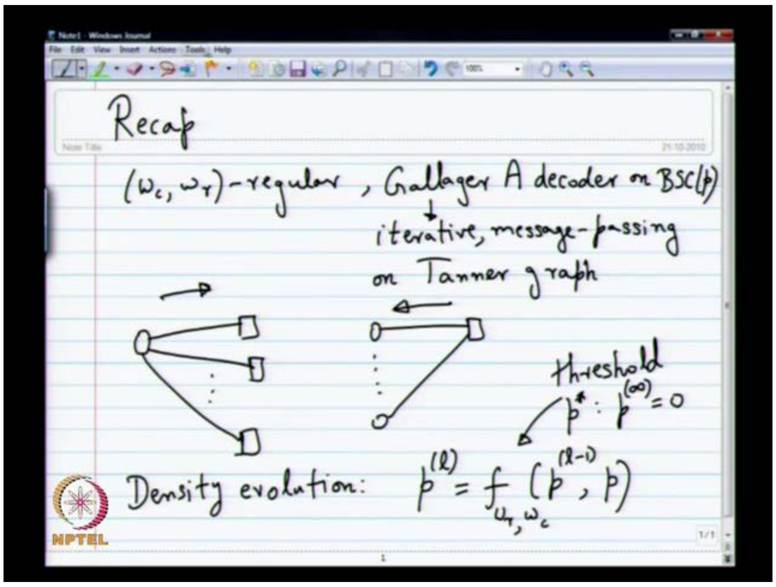
Coding Theory
Prof. Dr. Andrew Thangaraj
Department of Electrical Engineering
Indian Institute of Technology, Madras

Lecture - 26
Irregular LDPC Codes

(Refer Slide Time: 00:16)


Recap

(w_c, w_r)-regular, Gallager A decoder on BSC(p)

$\downarrow$ iterative, message-passing on Tanner graph

threshold $p^*: p^{(\infty)} = 0$

Density evolution: $p^{(l)} = f_{w_r, w_c}(p^{(l-1)}, p)$

So, let us begin once again the quick recap. So, what we did mostly was to look at w c comma w r regular LDPC codes and we looked at Gallagher a hard decision decoding algorithm. So, essentially this thing is a iterative message passing decoder, it operates on the tanner graph, so the way we describe that was there are two stages in the iteration first stage consists of messages being passed from the bit node to the check node.

The other stage consist of messages being passed in the opposite direction and what are these messages these massages are estimates of this, so this message flow on edges and particular messages it is flowing on a edge. It is basically an estimate of the bit node that the edge is connected always irrespective of the direction in which messages flowing, it are always estimates of the bit node. That it is connected and other way for doing is the first iteration that is some message that you send and then the subsistent subsequence iterations send another message.

We also saw this idea of density of evolution which is very crucial which gives you a nice handle on what is actually happening what is the idea behind density of evolution.

So, tracking the probability that a particular message is going to be a rather right that is the main idea. How do you track that way there is you have take a lot of care in doing that the first assumption that is crucial in density of evolution is that the all 0 code word was transmitted.

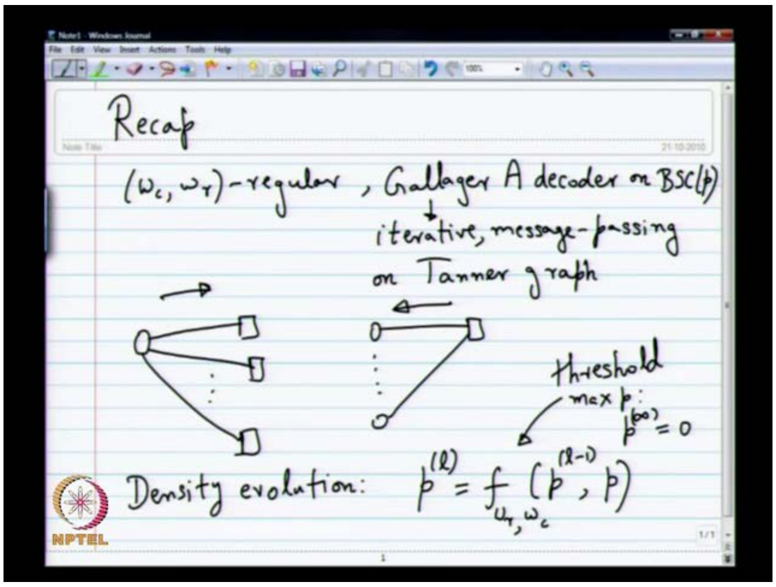
So, the all 0 code word was being transmitted is a crucial assumption and that can be justified by some means and once you make that assumption it is easy to track the probability that particular message is an error. Essentially, what you will have is you will have a formula like this. So, P l and the l th iteration the probability of error in the l th iteration will be some function which is parameterized by w r and w c of P l minus 1 and then the P which is the P s c P s c transition probability is that.

So, this is to do the crucial method of analyzing the crucial step in the analysis step in the LDPC codes, so there are ways of justifying this if you want this formula to hold for l iteration what should happen in some condition. There should be no cycles of length to l lower, so the Gath should be 12, only then this formula will hold and the kind of do not take that too seriously.

Then, let it not hold, but I will still do it and then we saw stimulation plots where that made sense. So, that also made sense and this formula was also useful because when n becomes really large probability that there is a cycle in that 12 goes to 0 in every neighborhood it goes to 0. So, it becomes reasonably good formula for large plot and we saw that also in simulation results as block length increased the prediction given by this formula was seen to hold more and more.

So, you must saw that the curve was shifting closer and closer to the crucial line as n became large. So, another crucial property for this iteration has is a threshold property what is the threshold property. You can show that there is a P star such that P l P infinity is 0, the maximum p, so let me write the little bit more clearly.

(Refer Slide Time: 04:30)



Recap

(w_c, w_r)-regular, Gallager A decoder on BSC(p)

iterative, message-passing on Tanner graph

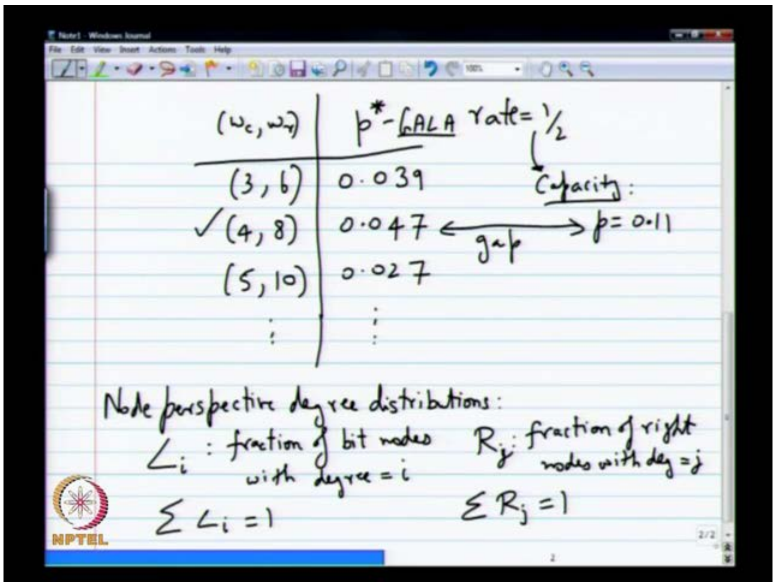
threshold
max p: $p^{(\infty)} = 0$

Density evolution: $p^{(\ell)} = f_{w_r, w_c}(p^{(\ell-1)}, p)$

So, threshold is the maximum P such that P infinity is equal to 0 in this iteration, so you keep on increasing P from 0 onwards of course, if P is 0 P infinity is also 0 plus if you keep increasing P infinity will continue to be 0 for some time. After that, it will shoot up to a non 0 value, so the maximum possible value for which this equal to 0 happens infinity equal to 0 happens is called the threshold.

Remember, once again technically you cannot say maximum because such a maximum will not exist in that set it will be supremum or something maximum is good enough for us will stick to that. So, any questions or comments on this, so then if you actually look at that, there are some thresholds that you can look at I think I did not do in last class, but I will tell you, so if now make like a table of w r w c and P star say for a particular rate.

(Refer Slide Time: 05:45)



(w_c, w_r)	p^* - GALA
(3, 6)	0.039
✓ (4, 8)	0.047
(5, 10)	0.027
⋮	⋮

rate = $\frac{1}{2}$

Capacity: $p = 0.11$

$0.047 \xleftrightarrow{\text{gap}} p = 0.11$

Node perspective degree distributions:

L_i : fraction of bit nodes with degree $= i$

R_j : fraction of right nodes with deg $= j$

$\sum L_i = 1$

$\sum R_j = 1$

So, it is a rate equals half what are the various w r w c possible this is P star for GALA a GALA decoder or gala decoder sounds like this. So, if one thing you can do is this, so I should say w c w, some reason I always like saying w r w c, I do not know why. So, 3, 6 is one choice which gives you a design rate of half and that threshold happens to be 0.039. If you want to be very exact another possibility is 4, 8 also give you a design rate of half and this would give you in fact the threshold of 0.047 and then 5, 10 gives you a threshold of 0.027. Then progressively do not leave voice sense, you can do this if you like if you take 6, 12, it will be even lower, so those are things you can look at.

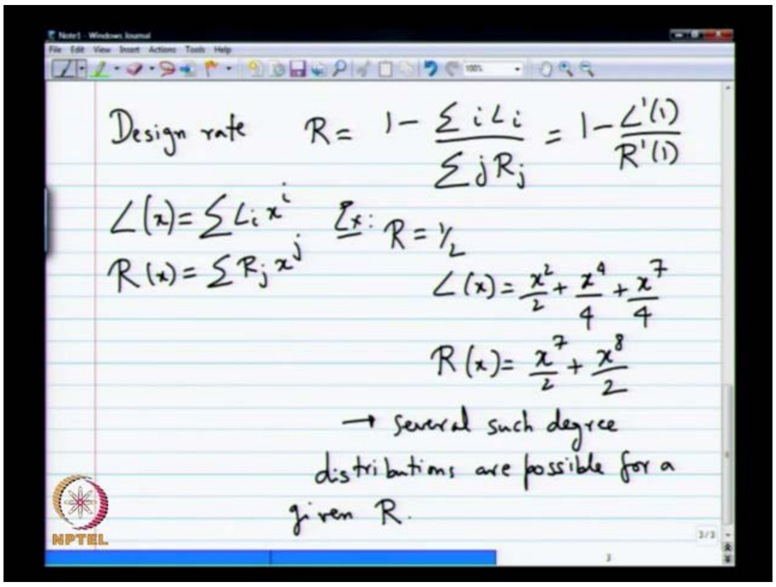
So, if you stick to regular codes based on threshold which is the best degree distribution 4, 8 is the best seems like the best degree distribution, but really there is not too much choice. Here, you know if you stick to regular codes it is seems like this too few it not to many choices, so you just one or two you can take then quickly you settle down settle on the right answer.

So, this basically proms the move towards irregular codes were you allow multiple degrees multiple bit node degrees multiple check nodes degrees or interims. The partly check matrix multiple column which multiple column weights multiple not just a single one and then see what happens to the threshold. So, if you have threshold etc all that you can study and then maybe they will be that a thresholds, so remember if you want to know capacity it turns out for rate half capacity is capacity achieving probability is basically capacity will come at P equals 0 0.11. So, what this means is if you are if you are channel transition probability is smaller than 0.11, you can have rate half for large enough block, let that is what is means.

So, for a comparison, if you compare this two guys is a bit far there is a gap P equals 0.11 is capacity for rate half the best we are getting with regular codes is 0.047. So, the question is the same rate is half what about the actual code rate become, it might become 1 by 10 or something. It would not become we can show some results that if you pick to our matrix at random with very high probability, you will have a rate close to half if you can begin linear depend us in spare matrix.

Now, we moved on to irregular codes and it defined something called the degree distribution polynomials there defined the degree distribution polynomials. So, with the node prospective, we defined that the node prospective degree distributions basically capital l sub by is the fraction of nodes with degree fraction of bit nodes or left nodes with degree equal to i. Same is the case with R i, what is R i fraction of right nodes with degree equal to may be R j just to say leave some other dummy variable. If you do this based on the designed rate, there are constrains the first constrains is that summation l i, it should be equal to 1. Similarly, summation R j should be equal to 1 this are all fractions clearly they should all add up to one fraction of nodes.

(Refer Slide Time: 10:24)



Design rate $R = 1 - \frac{\sum i L_i}{\sum j R_j} = 1 - \frac{L'(1)}{R'(1)}$

$L(x) = \sum L_i x^i$

$R(x) = \sum R_j x^j$

Ex: $R = 1/2$

$L(x) = \frac{x^2}{2} + \frac{x^4}{4} + \frac{x^7}{4}$

$R(x) = \frac{x^7}{2} + \frac{x^8}{2}$

→ several such degree distributions are possible for a given R.

Then, you can compute the design rate the design rate will turn out to be something like this R equals 1 minus summation i l i divided by summation j R j. So, you can also defined some polynomials if you like summation l i x power i R f x is summation R j x power j and then so those polynomials this will be l point one divided by R, 0.1. So, these are just convenient forms for expressions, so one example I gave for R half. I think you computed this for designed rate half we had a one example l of x being equal to x square by 2 plus x power 4 by 4 plus x power 7 by 4 and then R of x is equal to x power 7 by 2 plus x power 8 by 2.

So, one thing you can try for instance is to say I will allow degree 2 degree 4 and degree 7 for the left nodes, let me not fix the fractions, now I will simply say I will allow degree 2 degree 4 and degree seven for the left nodes. So, what does that mean it means l two is not 0 l four is not 0 l 7 is not 0 and l 2 plus l 4 plus l 7 is 1. All these case are between 0 and 1 and for the right hand node, I will allow only degree 7 and degree 8 everything else I will make 0.

Suppose, I add those two constraints how many possible degree distributions will you have for designed rate half. So, that is those kinds of questions you can ask it is a simple question there will some answers for that. So, first of all you have one equation for l's and one equation for R's saying summation l I is one summation R I is 1. Then there will be one more equation fixing the designed rate to be half what kind of equation will that, it will also be a linear equation.

So, you have three linear equations and how many variables do you have five variables and there are forced to be between 0 and 1, so you have some any quality constraints also, but in general you can expect infinite number of solutions there will be huge number of solutions. So, in fact you do not have to put a constraint on the sum of l i and all that so you can do some adjustments with that and if you if you want you can exactly solve this problem, but it is not so critical.

Today, as long as there are linear constraints nobody is cared about linear constraints, so it is very easy to program. So, you can do that, so final point they want to make is even if you restrict you degrees to be a few more than constant for a particular designed rate. You will have several degree distributions, so this was just one such degree distribution several such degree distributions are possible for all possible for a given designed rate given designed rate R.

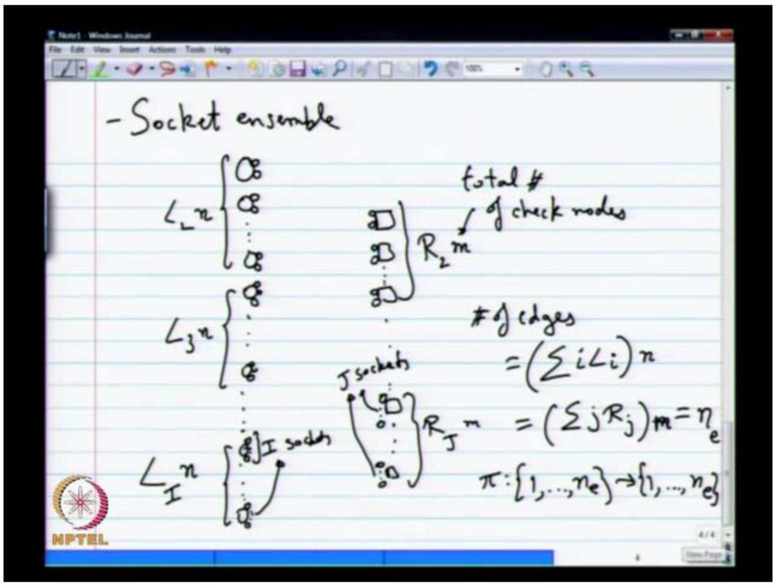
So, this increases your pole of candidates codes and candidate ensembles to speak, so not codes, so if a codes always remember when I say LDPC codes, it is just a ensemble when I fix a degree distribution the code never gets fixed. So, you have to pick the random member from the random matrix from the ensemble, so you have such a huge set and hopefully if you come up with a density evolution formula for irregular codes we have formula only for regular codes. If you do so for irregular codes, you can optimize over all these degree distributions for a particular designed rate to get the best possible threshold.

So, that is the basic idea behind moving towards irregular codes, the regular codes we saw that there is gap between the capacity and the threshold that it is a chewing. So, maybe you can do a little bit better there, so that is the basic idea behind moving towards irregular codes and you can see there is huge number of degree distribution for a given designed rate.

Even if you fix the number of allowed degrees there is a huge number, so if you vary that also you get a even bigger number of possibilities, so the question is how complex is this research will be. It will be very complex, I am not saying it will be very easy, but it is doable today computers are powerful there is very powerful algorithm like whatever different evolution is a very common. That is used all these genetic type algorithm for optimizations, so you can use all these throw some very heavy machinery at it eventually it will work. So, that is the idea, but the idea is other thing is you do not have to do this at some gigabytes second or something.

So, this is some initial design that you are doing once and you freeze it that is all, so you do it as you can take like six months to do it as long as it that at the end you are done you know you do not have to do it online or anything like that. So, let us see let us see let us see how to go about doing this of course, coming up with construction for irregular degree distribution also is a little bit tricky. So, for regular we had some constructions, but you can always define a socket construction.

(Refer Slide Time: 16:28)



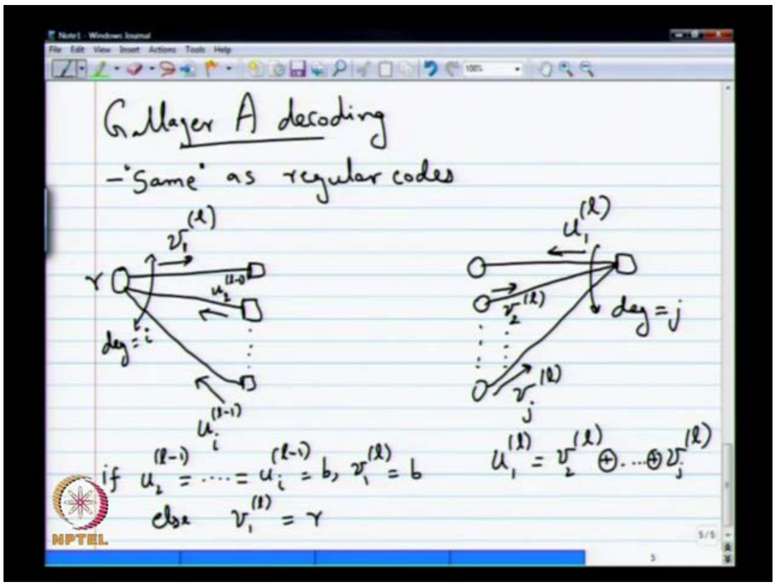
So, we will always think of a socket ensemble for irregular codes so how do you do the socket ensemble. So, you have l one n nodes of degree what of a degree 1, so I am not going to put degree 1, so I will just put start with degree 2 l 2 n of degree 2, then l 3 n of degree 3 and so on. Finally, may be some l, I do not know what do you want to call this guy maximum may be I will put I here, so I n bit nodes, so all these guys have degree 2 all these nodes have degree 3 etc. So, the first set l 2 n has degree 2 the next set l three n has degree 3 so on, so what can I do, I can put two sockets in each of these nodes.

Then, I would put three sockets in each of these nodes so on till I will put how many sockets here I sockets in each of these nodes. Now, the same thing I can do for check nodes also this going to be R 2 what R 2 M, M is the number of check nodes it is not the it is not n. So, m is the number of total number of check nodes, so likewise all the way down to R capital j m and I will put two sockets here and I will put j sockets here. Now, you have to count the total number of edges what is the total number of edges, so there are two ways of doing it. It is going to be summation I, LI times what times n is that correct or not maximum this side.

This will be the same as summation j R j times m, so this is the total number of edges may be I call it some n e what do I do, I pick pi to be a permutation of any objects and then place the edges according to this random permutation. So, you number this sockets from one to n e on the left side you number the sockets from the same one to same n e on the right side there will be exactly n e sockets on both sides right pi will be a permutation from 1 n e 2, 1 n e. So, for each permutation, you will get a code, of course, it is a multiplied problems still remains, so let us say we throw away all the multiplied edges for distance and then you get then you get codes from it.

So, technically for any irregular degree distribution also you can do a socket construction there is no problem, so construction is specification seems designed rates seems only next step is to look at decoding. So, once we will once you assure that the decoding that we had the GALLA decoding for instance also extends to the irregular case. So, that is the good for step and then we want to extend density evolution to the irregular case once all of those is done then you can go ahead and work on this complicated optimization or search for the best possible degree distribution for a particular designed rate.

(Refer Slide Time: 20:47)



Gallager A decoding

– 'Same' as regular codes

if $u_2^{(l-1)} = \cdots = u_i^{(l-1)} = b$, $v_1^{(l)} = b$

else $v_1^{(l)} = r$

$u_1^{(l)} = v_2^{(l)} \oplus \cdots \oplus v_j^{(l)}$

So, that is the next two steps that we will tie and do so what about Gallager decoding, so before this if you want me to mention what will you do about encoding, what about encoding of irregular code. So, once you have the parity check matrix encoding is at least principle trivial, so do you grow in elimination convert it into systematic form and then do the encoding. So, that is going to be a little bit more complex if you want to do efficient encoding there are ways of doing that also. All those things will extend without any problem the critical thing is the decoding, so let us see how the decoding extends decoding is also actually quite simple.

Remember, how did we describe Gallager, finally, on the stenograph each iteration has two steps in the first steps the message were passed from bit nodes to check nodes in the regular codes. Each bit nodes were connected to a to the same number of check nodes you had w R on each sides, but what will happen in the irregular case. There will be different depending on how many there are you simply pass that many messages, the same principle holds you send messages on the edges message.

Sent is an estimate of the bit nodes that is connected to except that in the irregular case every bit node will not do the exact same thing as in it would not have the same number of neighbors that is all the principles is the same. It will pass messages according to the same algorithm only, but bit node of degree 2 will send out only two messages well a bit node of degree 10 will send out 10 different messages that is all. So, rather than that, I can very easily write down the rules for the iteration, so I am going to say Gallager decoding same as regular with some codes. You have to just understand that this one minor change does not the degree must be different, so how does this, this look.

So, you have a bit node which is connected to let us say I check node, so let us say degree here is I, so what would have happened in the l minus 1 th iteration you would have got. I think if I remember, I called that u l minus 1 is that correct u 2 may be right, so you will go all the way down to u I l minus 1. So, may I write this again, so the message that was received in the previous iteration is like this and the message that passed from on the first edge in this iteration is b 1 let us say l.

How do you decide b 1 l if all the u's agree equal to b, then v 1 l equal to b, so let us say the channel received information is l's v 1 l equals R, so the rule remains exactly the same except that there might be some bridges which I have connected. Only two nodes, two check nodes and there might be some bridge which I connected to more than 2. So, depending on what we are connected to you simply change the degree there and you have this exact same rule which you can repeat there is no complication as far as the iterative message passing decoder is concerned. It works on even a irregular graph exactly the same way without any problem so what happens on the check node side.

So, the check node might have degree, so you can run this decoder and if you want to measure performance for the decoder you have to have some analysis is good. Suppose, we cannot analyze what would be a very brook force way of analyzing the decoder you have to do stimulation, but then if you want stimulate down to 10 power minus 3 frame error rate you have to stimulate 100, 1,000 blocks. So, one way of optimizing the degree distribution is doing what you pick a particular degree distribution, how do you measure the performance of the degree distribution you stimulate and recreate and create not recreate create the b R verses e b over n node picture.

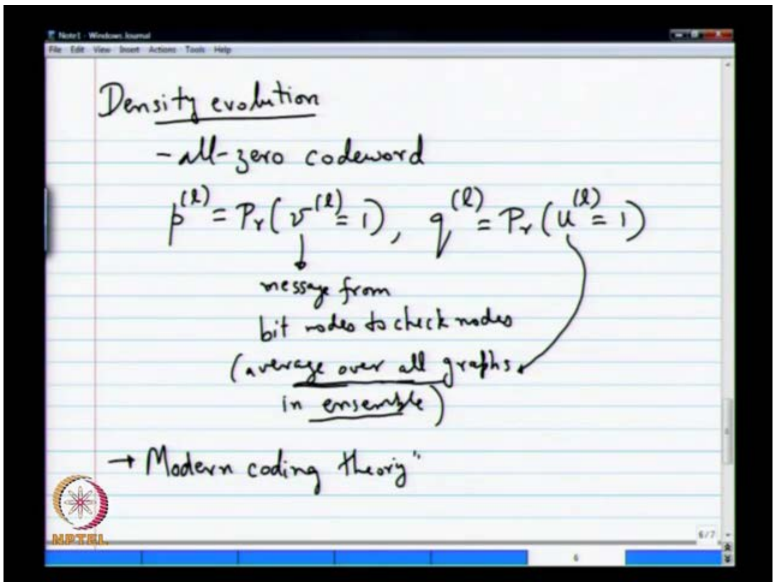
It will cross a 10 power minus 3, 107, you find that point and then you find one more degree distribution see how it behaves. That would be one way of measuring performance, but given that we have a huge number of degree distributions some procedures like this will not even convert in finite time. Even though you have even though you have like 6, 7 months in your hand you may not be able to do anything like that also, you really desperately need some way of extending the threshold analysis. Even if it is approximate, even if there are some hand waving's, then here and there even if it is not very exact if you have some number which can be easily computed given the degree distribution which measures the threshold.

So, to speak, then you can optimize the degree distribution for that number so in fact it turns out the threshold computation is also hard we would not have a closed form expression for threshold even in the regular case. We did not have a closed form expression, actually for this algorithm may be you can get some approximations, but it is very hard to get a closed form expression I think. So, general case you may not be able, so for the irregular codes also you may not get a closed form expression, but at least numerically you should be able to evaluate the threshold and that might be way better than doing 100, 1,000 blocks of stimulation.

Actually, implementing the decoder and you do not know how many iterations to do number the block length n is going to be very large and n is going to be a 100, 1,000 or something. Then you have to do so many iterations hundreds of iterations you have to do so it takes a long time to that takes a long time to finish till computers reach that stage. You still need some analysis of how this is going to work so let us see how to extend the density evolution the way I describe it.

I will do I will do a kind of a hand waving justification for something so there are there is a slightly more rigorous justification possible, but at the end of the day it is an approximate analysis. So, I will try and mention the main steps and give you some guidelines on how to understand it, but I will derive it with a very simple kind of hand waving kind of idea. So, that will that is how I will derive it, so the basic idea is we want a repeat the same analysis that we had before, so we want to do density evolution.

(Refer Slide Time: 29:22)



Density evolution

- all-zero codeword

$p^{(l)} = P_r(v^{(l)} = 1)$, $q^{(l)} = P_r(u^{(l)} = 1)$

$\downarrow$

message from
bit nodes to check nodes
(average over all graphs
in ensemble)

→ "Modern coding theory"

How did density evolution work, in the previous case we had two different quantities that we tracked across iterations there was this P l. Then the q l P l was the probability that the message from bit two check was an error and q l was the probability that the message was the probability that the message from check to bit nodes was an error. So, I will give you a simple way of getting an approximate expression for these things in the irregular case there will be some holes in the way I described it to rigorously properly justify it.

You need some one more course material, so maybe it is not possible in this codes I would not, I would not do too much in the later on, I will mention all the caviars and what else is need to make it little bit more rigorous. So, that is what we are going to do so but before that the all 0 code word assumption can be justified in the same manner as before we will actually, we did not really justify the all 0 code word given before, but it can be justified.

So, I am going I am going into that detail, but you assume the all 0 code word and that is no problem that will be justified for the same reason that it works for regular codes. So, it is no problem at the extends without any approximation so like before I want to have P l to be probability that v l is equal to 1 and likewise they also want to have q l to be probability that u l is equal to 1.

So, now comes crucial point and defining v l and u l, so in the regular case I really had no problem in defining v l and u l to be I mean not defining in assuming that v l and u l. That was a crucial assumption if you remember in the so that is the first the first kind of approximate analyze approximation that will enter the analysis. I will no longer look at one graph I will look at all the graphs on average over that so it turns out you can justify this IID nature when you average over all the graphs I am not going to going to great detail. In fact you can justify it only as n turns to infinity the asymptotic situation it is justified.

So, let us let us leave a try that and then again for u l also the same thing holds once again you have to average over all graphs in the ensemble only then every edge will be equally likely to come from an average kind of neighborhood. This looks the same, that is the idea, so I do not want to go into great detail here, but we will just remember that the messages that we will be tracking are not the messages on a particular graph are not looking fixing a graph and doing decoding.

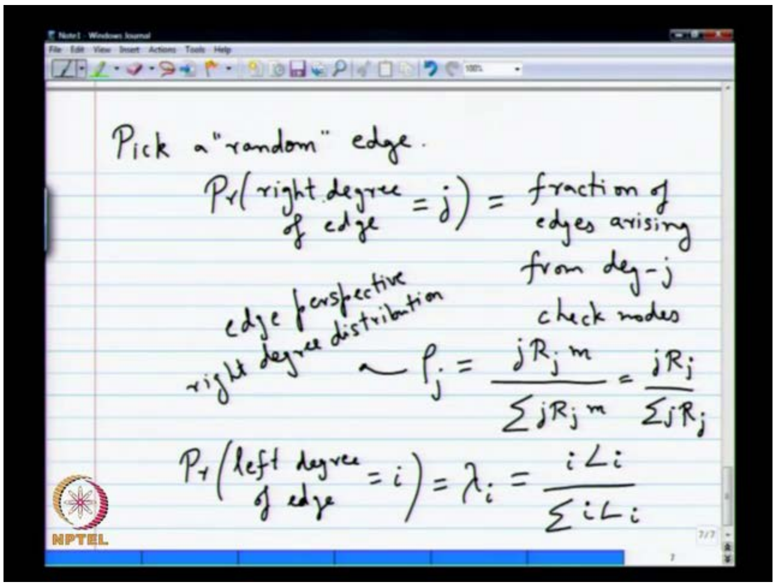
You are not tracking messages there what is the messages is being tracked you will be looking at all the graphs together. So, all 0 code word is being received over the b s c the same received word is being decoded by all graphs from the ensemble how many graphs are there are huge number of graphs all is very large. So, huge number of graphs all of them are being decoded decoding is done simultaneously, then what message will attack an average an edge that I pick randomly from all of those things. So, that message I will get that, I can say will be IID seems to make sense look, so many huge things may be IID it is not too difficult to justify.

So, the way there is regular ways of justifying it, but intuitively that is what happening that is the picture you should have in mind, I do not have one graph I am not tracking the message on one graph. I will first pick the particular edge, the particular edge that I can specify in the certain ensemble. So, number from edge out of this guy may be specify that then that one you have to now look at every single code.

So, it is neighborhood each in each code the neighborhood will be different you can kind average over all of those things that is the idea. So, I do not know anything did I do anything wrong. So, we are back, so it is a bit of like I said it is a bit of official idea you have to justify it very carefully. We will not see that justification class I love, you look at the book by Bunk K Richardson modern coding theory. We have a very good justification of this and there's lots of detail in that book. So, if you are interested you can take a look at it, but we will proceed with like I said my own pseudo interpretation of that.

I will give you a simple derivation, so remember we are interesting the density tracking the probability of u l and v l being in error through these two steps. There are two steps in the iteration, one step is the processing at the bites nodes and the other step is the processing at the check nodes, we will start with the processing at the check nodes, suppose I have a check node with degree j.

(Refer Slide Time: 38:32)



Pick a "random" edge.

$$\Pr\left(\begin{matrix}\text{right degree}\\ \text{of edge}\end{matrix} = j\right) = \begin{matrix}\text{fraction of}\\ \text{edges arising}\\ \text{from deg-}j\\ \text{check nodes}\end{matrix}$$

edge perspective right degree distribution $\sim \rho_j = \dfrac{jR_j m}{\sum jR_j m} = \dfrac{jR_j}{\sum jR_j}$

$$\Pr\left(\begin{matrix}\text{left degree}\\ \text{of edge}\end{matrix} = i\right) = \lambda_i = \frac{iL_i}{\sum iL_i}$$

So, remember once again, so let me so for some reason it has not oh I think it is not maybe it is not the standard one then I will so I want to pick a random edge what do I mean by random edge. So, when I say random edge the edge can be from any one of the codes so there is huge number of codes with the same degree distribution and I want to pick a random edge from this code. So, you pick a random edge and then ask the question what it is right degree and what it is, left degree. So, you have ask that question because only then you can do the evolution.

So, when I want to do the evolution of I mean when I want to find q l or u P l the probability the messages or an error the degree of the check nodes and the degree of the bit nodes plays an important role. Then I cannot analyze the probability of error on any particular edge I have to pick the edge over at random, so when I keep doing this a random edge when I pick a random edge you have a probability that this random edge is connected to a certain degree check nodes. The probability that this random edge is connected to a certain degree bit node, so let us try to compute that.

So, if you pick a random edge what is the probability that right degree equals right degree of edge remember what I mean by right degree of edge the degree of the check nodes that it is connected? So, what is the probability that equals j how you will answer this question. So, this will basically be let me remove the question mark this will be the fraction of edges fraction of edges arising from arising from degree j check nodes right that is the idea so in a graph. You have so many several check nodes of different degrees so there will be several edges which originates from different degree check nodes.

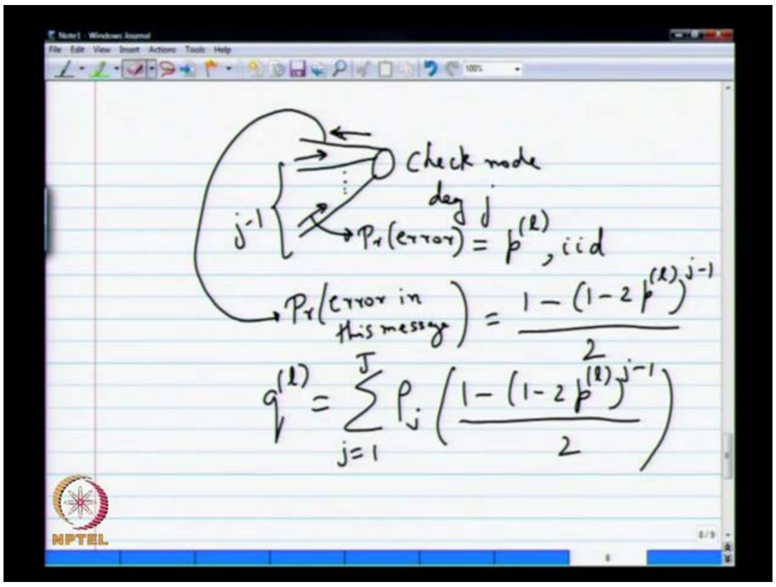
So, when I pick a random edge in a particular graph even in a particular graph a particular edge is likely to be connected to a check node of degree j with the sudden probability. That probability will be equal to the fraction of edges that come out of degree j check nodes and this fraction can be very easily computed what will be this fraction j times R j times m divided by submission j R j m.

So, m will cancel, so you simply get j R j by submission j R j, so since this quantity figures with the importance in our analysis we will call it row j. So, give it a new name this is the probability that the right degree of a randomly chosen edge is going to be equal to a particular j that is row j. So, this is also called the edge prospective degree distribution.

So, R is the row prospective right degree distribution row is the edge prospective degree distribution, so basically says what the probability that the randomly chosen edge even in a particular graph will be connected to a degree j check node. So, the same question you can ask for the left degree probability that left degree of a randomly chosen edge equals i I am using i and j these are dummy guys.

This will be also again fraction of edges arising from degree i bit nodes so we will call lambda i this is just notation, and just like before you can quickly, that this will be the formula right I L I divided by submission I L I So, these two fractions will play an important role in analysis, so that is fine let us accept those fractions as reality. Now, when I do that, let us see how I can go about doing this in this analysis without too much pain so what I am going to do is, I am once again going to look at the check node of degree j.

(Refer Slide Time: 43:26)



$$P_r(\text{error}) = p^{(l)}, \text{ iid}$$

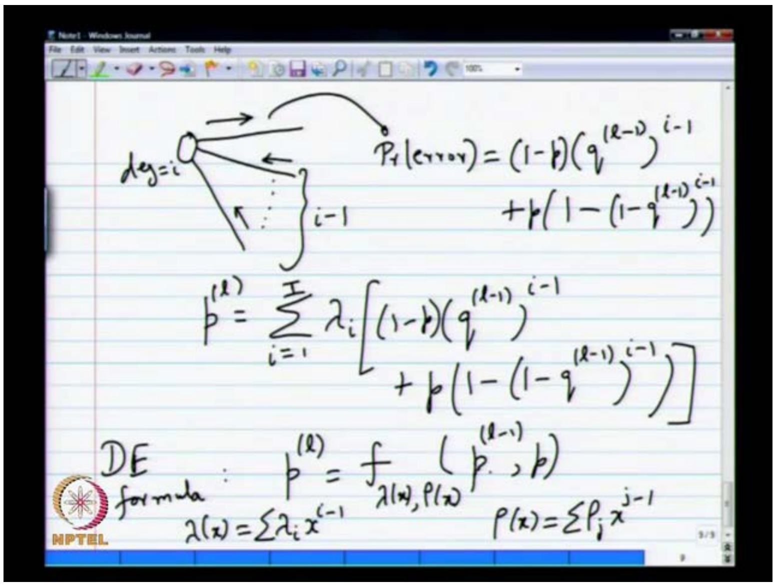
$$P_r\left(\begin{array}{c}\text{error in}\\ \text{this message}\end{array}\right) = \frac{1-(1-2p^{(l)})^{j-1}}{2}$$

$$q^{(l)} = \sum_{j=1}^{J} \rho_j \left(\frac{1-(1-2p^{(l)})^{j-1}}{2}\right)$$

Then, look at the messages that are coming in it will have j node j edges connected to it so this will be j minus 1 and this will be the message that I am interested in what is the probability of error in this message. So, I would received some messages from this j minus one guys and I will be sending out a message in this direction I want to look at what is the probability that the error in this message. So, remember what the probability that all these incoming messages are in error that is going to be IID according to v l probability of error here is some P l and it is IID right that is given to me.

Now, compute the probability of error in that message and for that I know the formula already so that formula is very simple this will be 1 minus 1 minus 2 P l power j minus one divided by 2, so that is the probability that I have an odd number of errors in this in this j minus one that I can do for a particular check node of a particular degree j, now this edge that is choose has a probability row j. So, in I mean if you instead of doing all that work I can give this hand waving justification say that it is justified by strong values, so that is the hand waving part.

(Refer Slide Time: 47:54)



$$P_r(\text{error}) = (1-p)\left(q^{(l-1)}\right)^{i-1} + p\left(1-\left(1-q^{(l-1)}\right)^{i-1}\right)$$

$$p^{(l)} = \sum_{i=1}^{I} \lambda_i \left[(1-p)\left(q^{(l-1)}\right)^{i-1} + p\left(1-\left(1-q^{(l-1)}\right)^{i-1}\right)\right]$$

DE formula : $p^{(l)} = f_{\lambda(x),\rho(x)}\left(p^{(l-1)}, p\right)$

$\lambda(x) = \sum \lambda_i x^{i-1}$ $\rho(x) = \sum \rho_j x^{j-1}$

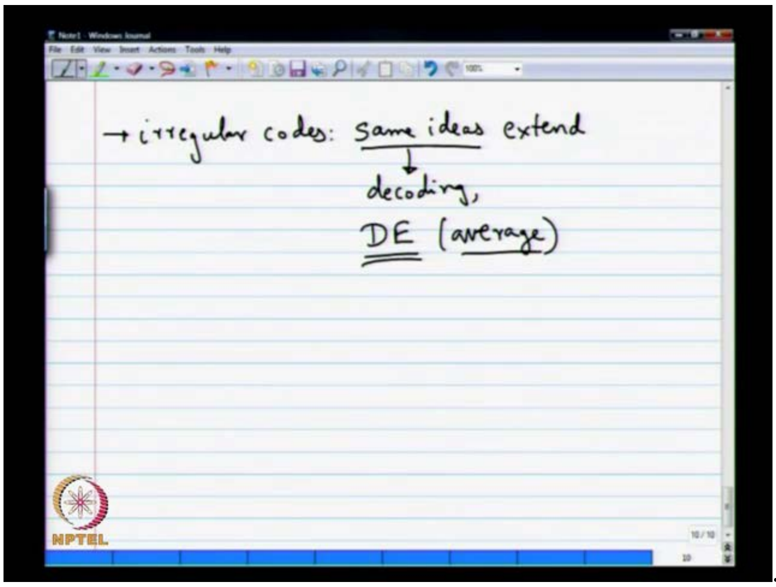
So, not rigorously justified so you have a bit node let us say it is of degree i, then it will have i minus one guy's from home there will be incoming messages and there will be an outgoing message whose probability error I want to compute. So, once again this guy probability of error will be is very known to me, so this is going to be 1 minus P times q l q l minus 1, I do not know I think this I think q l minus 1 raise to the power i minus 1 plus P times 1 minus one minus q l minus 1 raise to the power i minus 1.

This is the formula that I know and what will I do for P l I simply sum over I going from one to capital I lambda i which is the probability that might randomly chosen edge. So, to speak has connected to a degree i bit node and then I do this same way this multiplied by one minus P q l minus 1 raise to the power i minus 1 plus P 1 minus one minus q l minus one raise to the power i minus 1. So, that is fine alright, so I have a wave from outgoing from q l minus one to P l and then I also have a way of going from P l to q l.

Now, I can repeat this process and in general I will finally, a density evolution formula for P l which will be a function and the parameterization will be in simple terms you can write it as lambda of x and row of x what is lambda of x and row. There will be some polynomials involving the lambda, so I will write that down, so where lambda of x is simply submission lambda I it is it is common to write x power i minus 1 instead of x power i here for lambda. So, I will tell you y is soon enough, but it usually it turns like that row of x is written as submission row j once again x power j minus 1.

There is a very simple reason why they do j minus one instead of j for node prospective it is common to do j. Then for edge prospective, we do j minus one that is all so this is the density evolution formula, but remember this is not valid for a particular graphs. It is valid when you average over all graphs, so how good is this is an important question that we have to ask we will answer that soon enough, but this is the idea, so what is the summary of this lecture finally.

(Refer Slide Time: 51:14)



$\rightarrow$ irregular codes: $\underline{\text{same ideas}}$ extend

$\downarrow$

decoding,

$\underline{\underline{\text{DE}}}$ ($\underline{\text{average}}$)

.

So, even for irregular codes, so you have same ideas extending what are the same ideas decoding message passing decoding definitely extends more importantly density evolution extends, it is average density evolution. We did not provide a very rigorous justification for it, but I said it can be justified in a more rigorous way than what I gave you right note. So, we will stop here for now and in the next lecture move on and see something more about this density evolution.