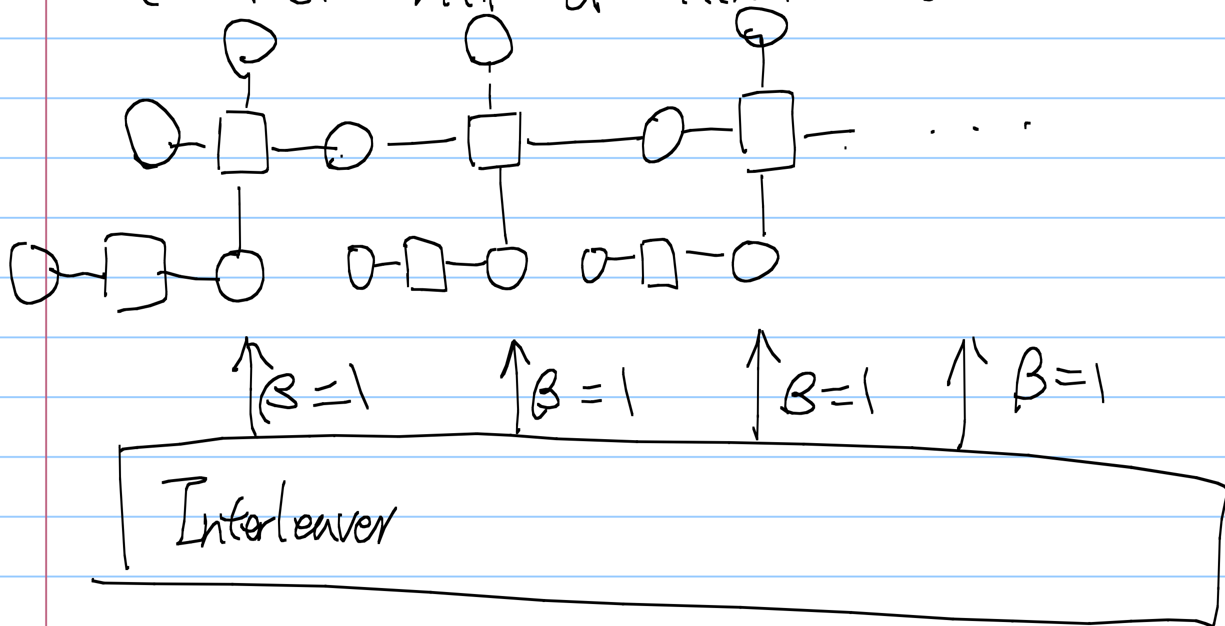
Lecture 15

Note Title 2/29/2012

Turbo decoding:

① Each iteration contains two parts

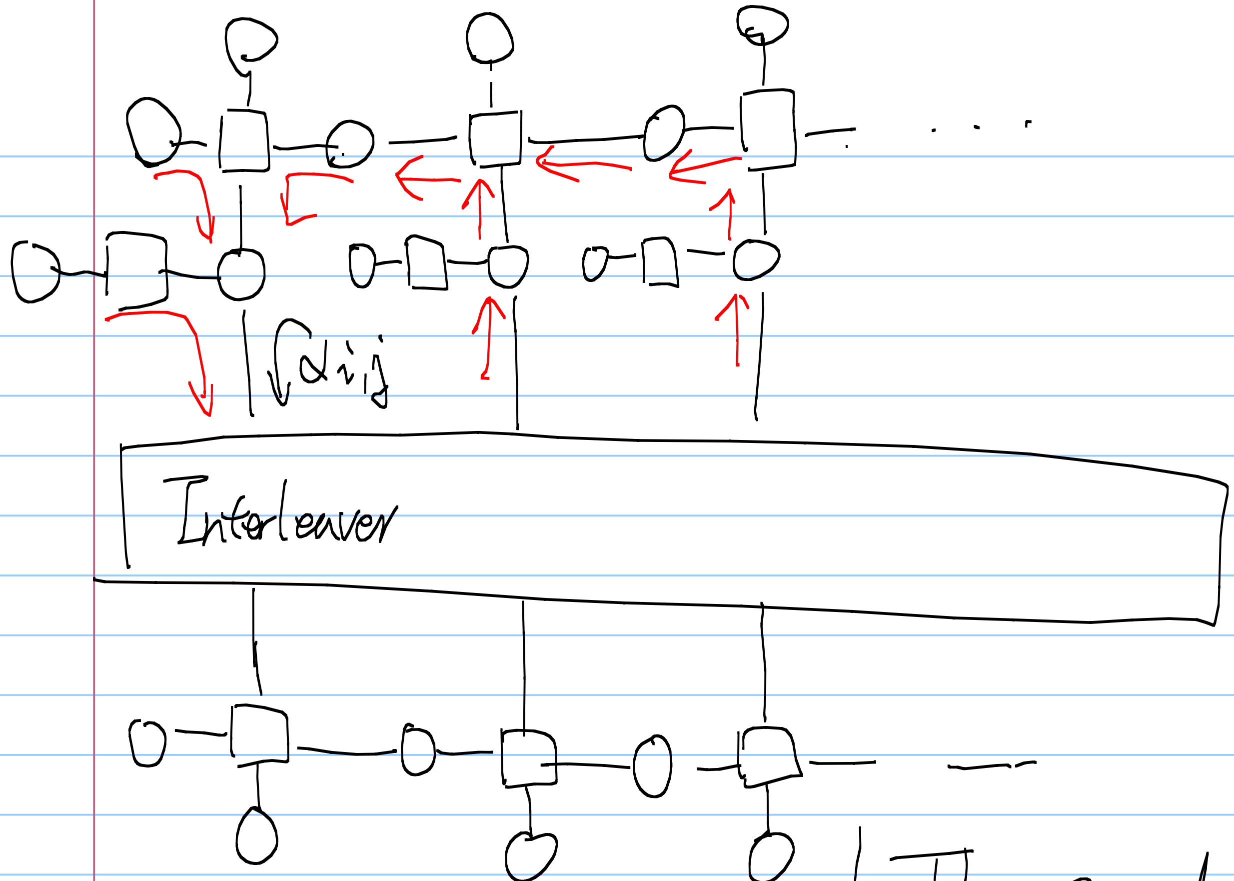
② The first half of iteration 1.



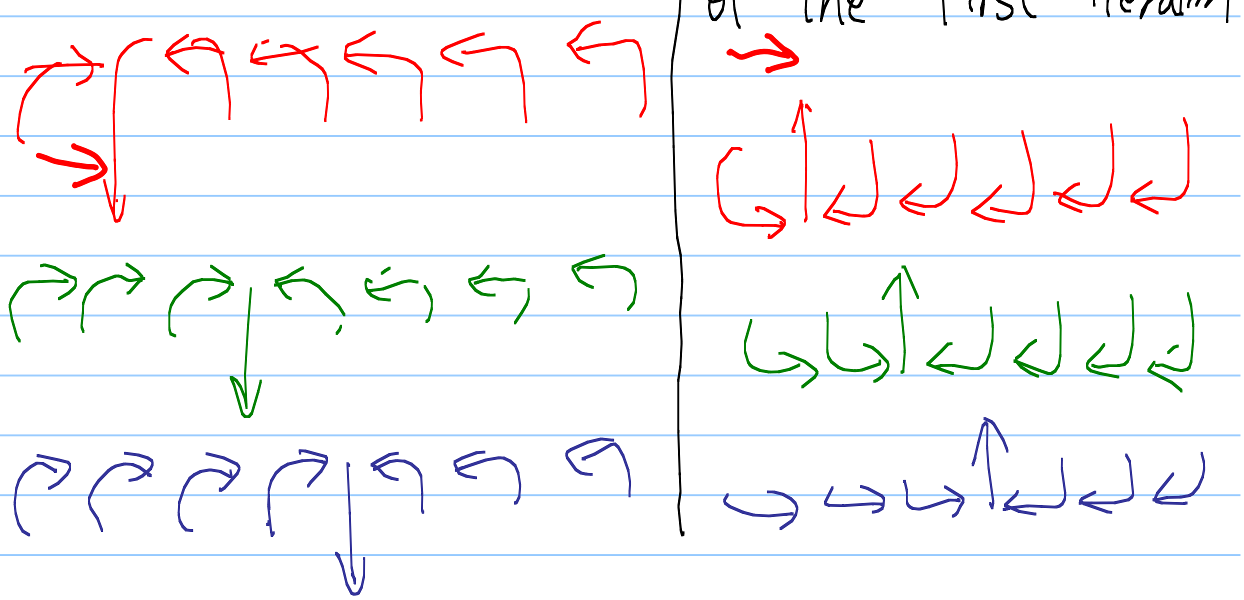
Since no information comes from the interleaver all the $\beta_{j,i}(\cdot)$ are initialized to one.

③ We can run the forward & backward iteration of the BCJR decoder.

④ The question is then how to combine the results of the first half (RSC_1) with that of RSC_2

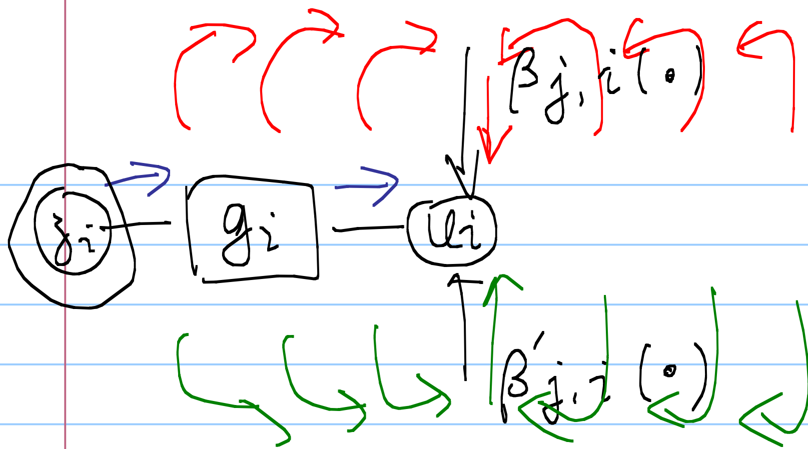


The second half of the first iteration



* When to stop? Terminate after a fixed # of iterations.

* How to make the final decision?



$$\hat{u}_i = \underset{u=0,1}{\operatorname{argmax}}\; g_i(z_i, u)\cdot\beta_{j,i}(u)\cdot\beta'_{j,i}(u)$$

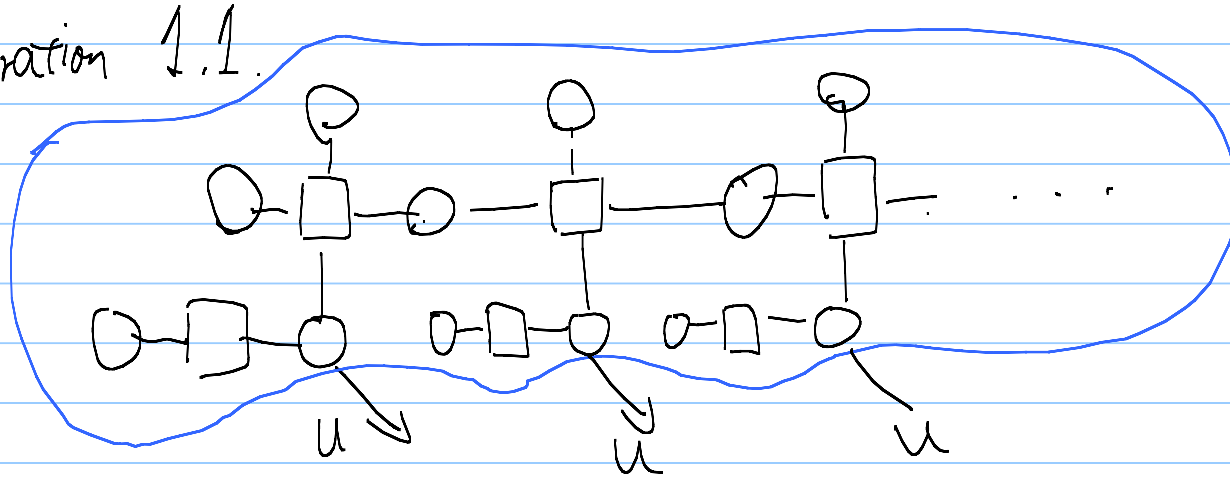
the final decoded codeword is

$(\hat{u}_1, \hat{u}_2, \hat{u}_3, \cdots, \hat{u}_k)$, a suboptimal bit oriented decoder.

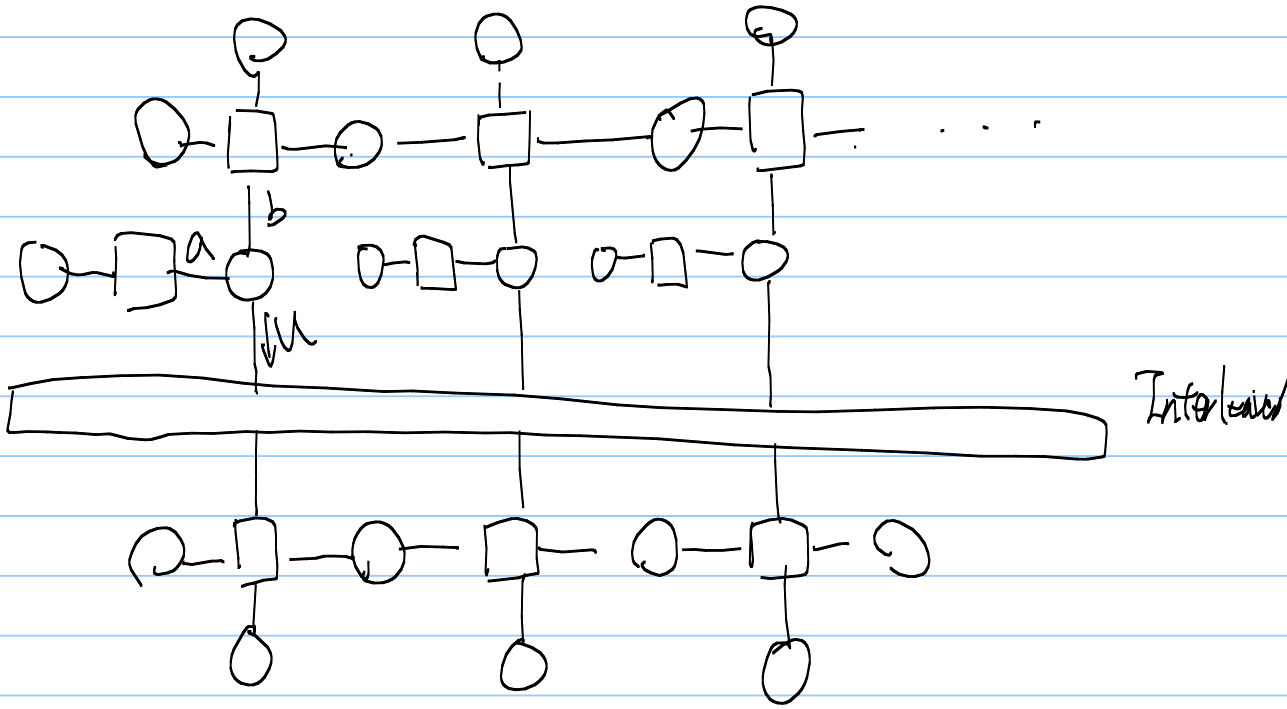
Revisit: The original turbo decoder & the concept of extrinsic info.

① Based on the off-the-shelf BCJR decoder

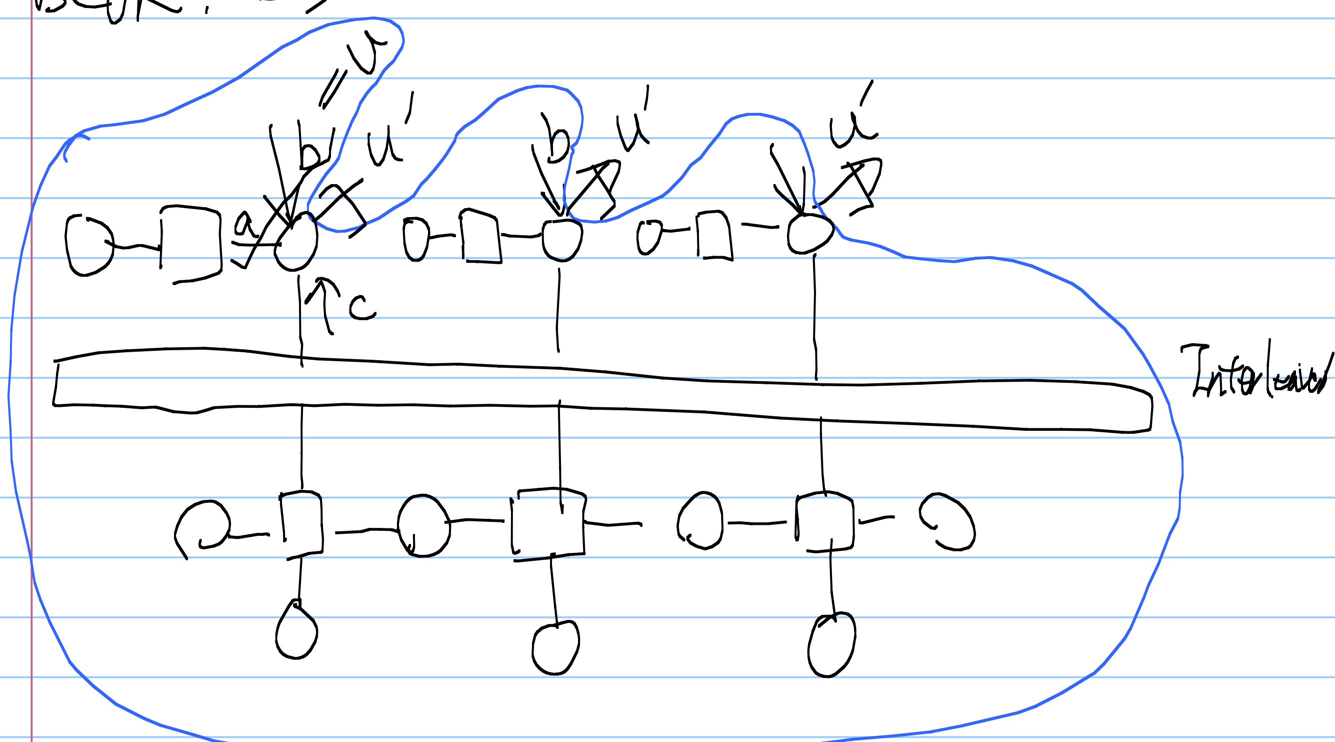
Iteration 1.1



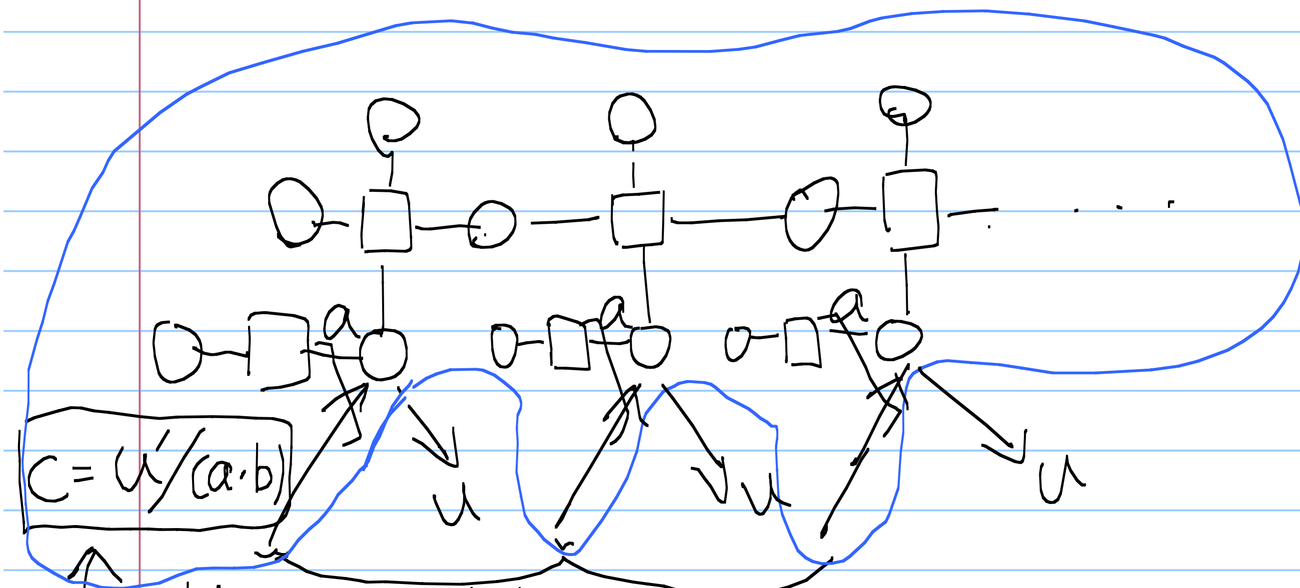
Iteration 1.2



because we use the off-the-shelf BCJR. ⇒



A problem arises for the 2.1 iteration

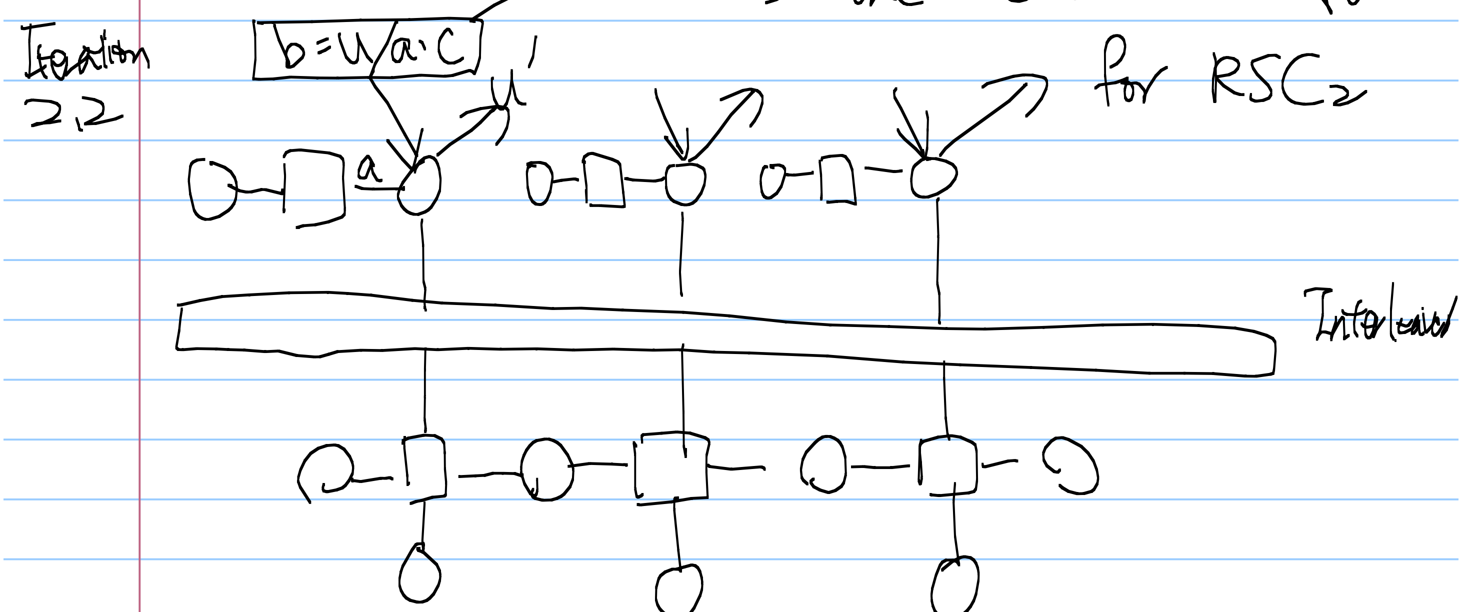


How to initialize these messages

Recall $u'(\cdot) = a(\cdot)\, b(\cdot)\, c(\cdot)$

We call the $C(\cdot)$ message as the <u>extrinsic info</u> for RSC 1, since it removes the $(a \cdot b)$ info from the previous iterations. C is then combined with $a(\cdot)$ to initialize the Off-The-Shelf BCJR

→ The extrinsic info for RSC_2



u': "soft decision" from RSC2

$c = u'/a\cdot b$: the extrinsic info from RSC_2

$c \cdot a$: The combined prior that is used to initialize the BCJR for RSC1.

u: "soft decision" from RSC1

$b = u/ac$: the extrinsic info from RSC_1

$b \cdot a$: The combined prior + extrinsic that is used to initialize the BCJR for RSC1.

The turbo code perf handouts.

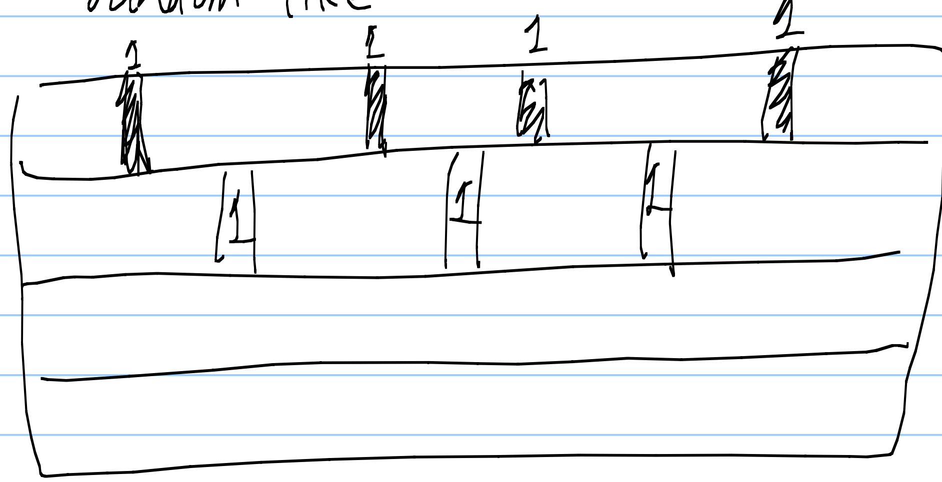
* In its original form, turbo decoding relies heavily on carefully distinguishing the "extrinsic", the "prior", & the combined prior information
* The factor graph representation provides a more intuitive way of describing the turbo decoder.
* Many variants: Symmetric vs Asymmetric turbo codes, RSC1 $\overset{?}{=}$ RSC2, Interleaver design, ...

* Low-Density Parity-Check Codes

The 2nd class of error control codes that is capacity-approaching.

Invented by Gallager 63, rediscovered by MacKay in 90's

* LDPC codes are simple binary linear codes with the parity-check matrix being "sparse" & "random-like"

$H =$ 

Each row has only a bounded # of of 1s, which are randomly shuffled.

* Low density. $\because$ $\dfrac{\#\text{ of 1s}}{\#\text{ of possible entries}}$

$$= \frac{O(n \cdot d)}{O(n^2)}$$

LDPC encoder:

① Given the H matrix, use Gaussian elimination to make $H' = [I \ P]$

& $G = \begin{bmatrix} P \\ I \end{bmatrix}$

polynomial-time complexity.

② Can we do linear / almost linear encoder? $O(n)$ or cn^2 for small c.

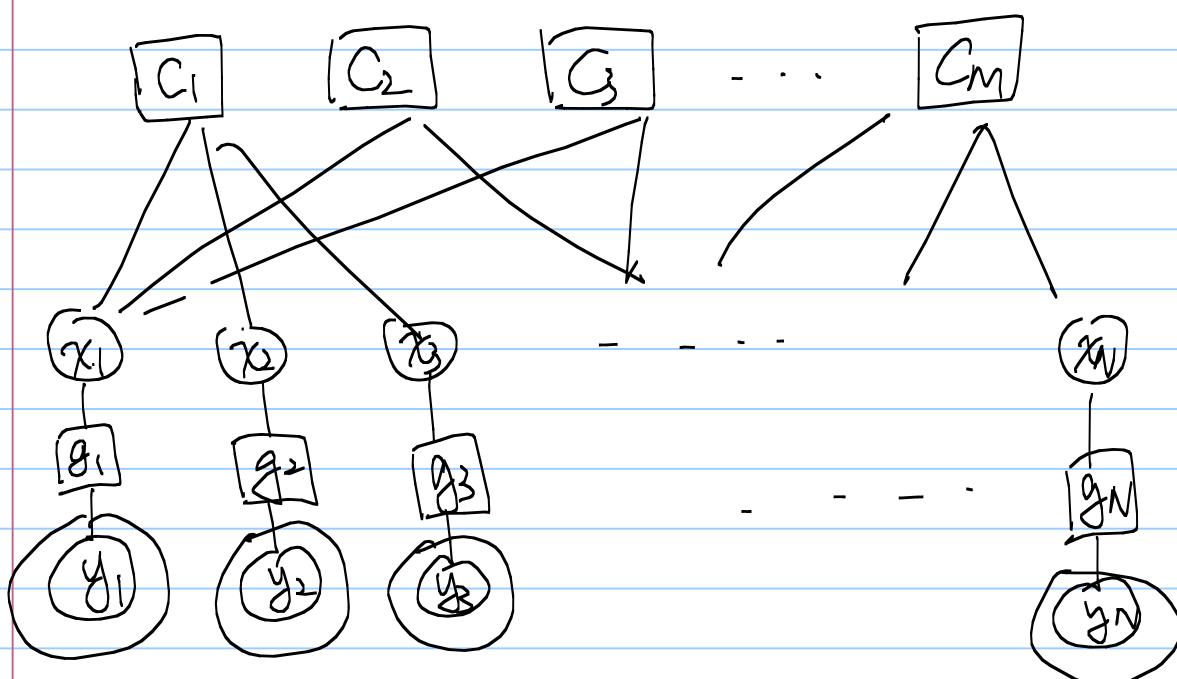
We will come back later

* LDPC decoder

* The factor graph representation of the likelihood function, assuming that we have independent channel

$$P_{\vec{Y}|\vec{X}}(\vec{y}|\vec{x}) = \prod_{i=1}^{N} P_{Y_i|X_i}(y_i|x_i)$$

$$\underset{\vec{x}:H\vec{x}=0}{\arg\max}\ P_{\vec{Y}|\vec{X}}(\vec{y}|\vec{x}) = \underset{\vec{x}}{\arg\max} \prod_{j=1}^{M} f_{c_j}(x_i \in \partial j) \cdot \prod_{i=1}^{N} P(y_i|x_i)$$



$$f_{c_j}(\cdot,\cdot,\dots,\cdot) = \begin{cases} 1 & \text{if the \# of 1s is even} \\ 0 & \text{otherwise} \end{cases}$$