EEE8099 – Information Theory & Coding
EEE8104 – Digital Communications

S. Le Goff

School of Engineering @ Newcastle University

Part 7
Practical Error-Correcting Codes
1993 - Today
LDPC Codes

The rediscovery of LDPC codes

1995: David J C McKay rediscovers LDPC codes.

He shows that these codes, when decoded in an iterative fashion, can perform very close to the capacity limit.

(1967 – 2016)

The iterative decoding algorithm used for LDPC codes is very natural and elegant. It can be performed using hard decisions (bit-flipping algorithm originally introduced by R G Gallager, the inventor of LDPC codes) or soft decisions (sum-product algorithm introduced by D J C McKay).

The rediscovery of LDPC codes

As expected, the sum-product algorithm provides the best decoding performance since it operates using soft decisions. It is also more complicated to implement than the bit-flipping algorithm.

The principles behind the sum-product algorithm are identical to these used for turbo decoding, with the following fundamental differences:

- In turbo decoding, two recursive and systematic convolutional (RSC) decoders keep on exchanging extrinsic info, which leads to a progressive improvement of their error performances.

The rediscovery of LDPC codes

There are only two RSC decoders but the complexity of each of them is high (remember the complexity of the MAP algorithm…).

- In sum-product decoding, many (short) parity-check decoders keep on exchanging extrinsic info, which leads to a progressive improvement of their error performances.

There are many parity-check decoders, but the complexity of each of them is rather low (it can’t be complex to decode a parity-check code, right?)

The rediscovery of LDPC codes

Consider a rate-1/3, length-6 ($n = 6$), ($w_c = 2, w_r = 3$)-regular LDPC code with the following parity-check matrix.

$$H = \begin{bmatrix} 110100 \\ 011010 \\ 100011 \\ 001101 \end{bmatrix}.$$

This implies that the coded bits must satisfy the four following parity-check equations:

$c_0 + c_1 + c_3 = 0$ $\quad$ $c_1 + c_2 + c_4 = 0$

$c_0 + c_4 + c_5 = 0$ $\quad$ $c_2 + c_3 + c_5 = 0$

The rediscovery of LDPC codes

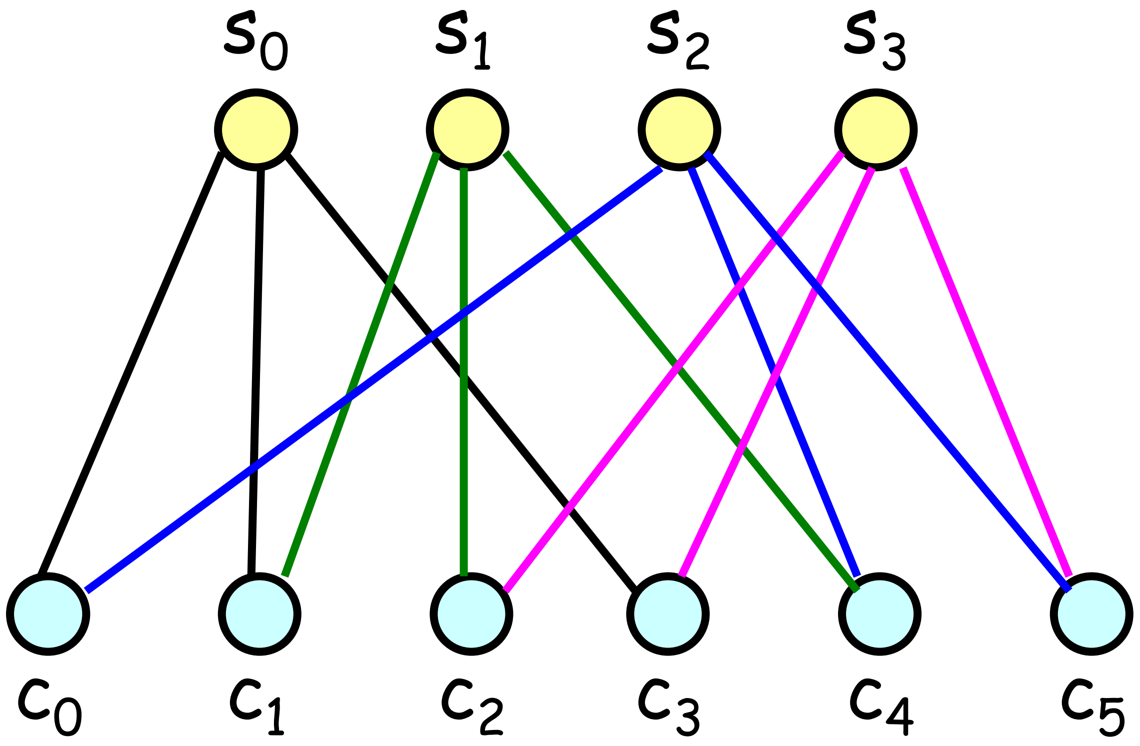
In other words, this LDPC code is simply a collection of four parity-check codes, exactly as a turbo code is a collection of two RSC codes.

We can thus decode this LDPC code by decoding each parity-check code individually and making sure that all parity-check decoders exchange extrinsic info with each others.

The rediscovery of LDPC codes

The Tanner graph of this code is shown below.

Check nodes representing the (n – k) parity checks (also called constraint nodes)



Bit nodes representing the n coded bits (also called variable nodes)

The rediscovery of LDPC codes

The Tanner graph can be used to decode LDPC codes.

Assume transmission over BPSK, AWGN channel. We receive a word $R = (r_0, r_1, r_2, r_3, r_4, r_5)$ composed of six channel samples.

Each channel sample r_l, $l \in \{0, 1, 2, 3, 4, 5\}$, has a Gaussian distribution.

The word R is an estimate of the transmitted 6-bit codeword $C = (c_0, c_1, c_2, c_3, c_4, c_5)$, with $c_l \in \{-1, +1\}$, $l \in \{0, 1, 2, 3, 4, 5\}$.

The rediscovery of LDPC codes

First, we take a hard decision on R using a decision block, thus yielding the binary word $\hat{R}$. We can then compute the vector syndrome $S = \hat{R} \cdot H^T$:

• If $S = \mathbf{0}$, we conclude that $\hat{R}$ is a valid codeword and no further processing is then required as $\hat{R}$ is the decoded codeword (the notation $\mathbf{0}$ denotes the all-zero vector).

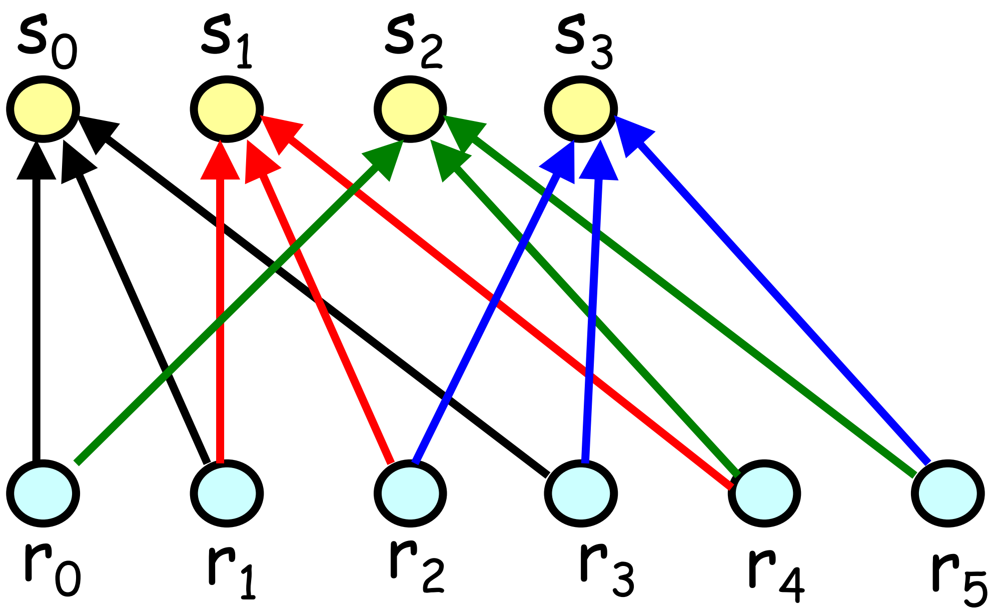
• If $S \neq \mathbf{0}$, we conclude that $\hat{R}$ is not a valid codeword and we thus need to decode $R = (r_0, r_1, r_2, r_3, r_4, r_5)$.

ML decoding is not an option when dealing with long codewords. We are going to employ an iterative decoding algorithm instead (as we do with turbo codes).

The rediscovery of LDPC codes

The Tanner graph can be used to understand the iterative decoding algorithm (also called belief propagation, message passing, or sum-product algorithm).

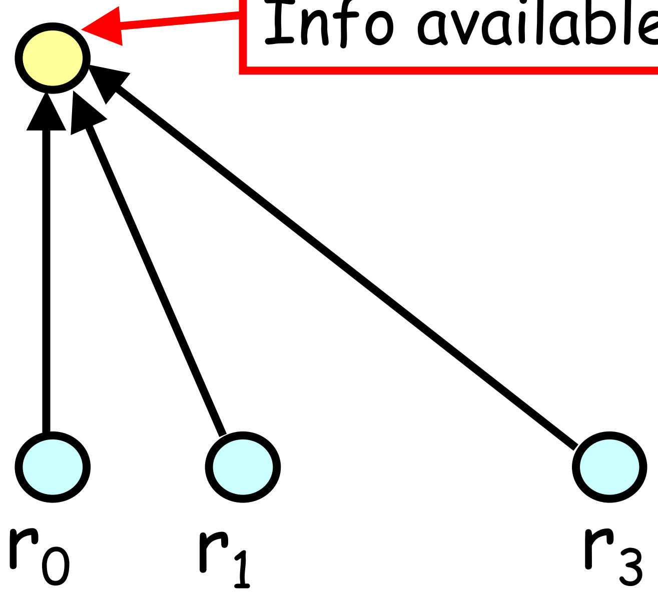
During the first decoding iteration, the channel samples $r_0, r_1, r_2, r_3, r_4, r_5$ are sent to the check nodes.



The rediscovery of LDPC codes

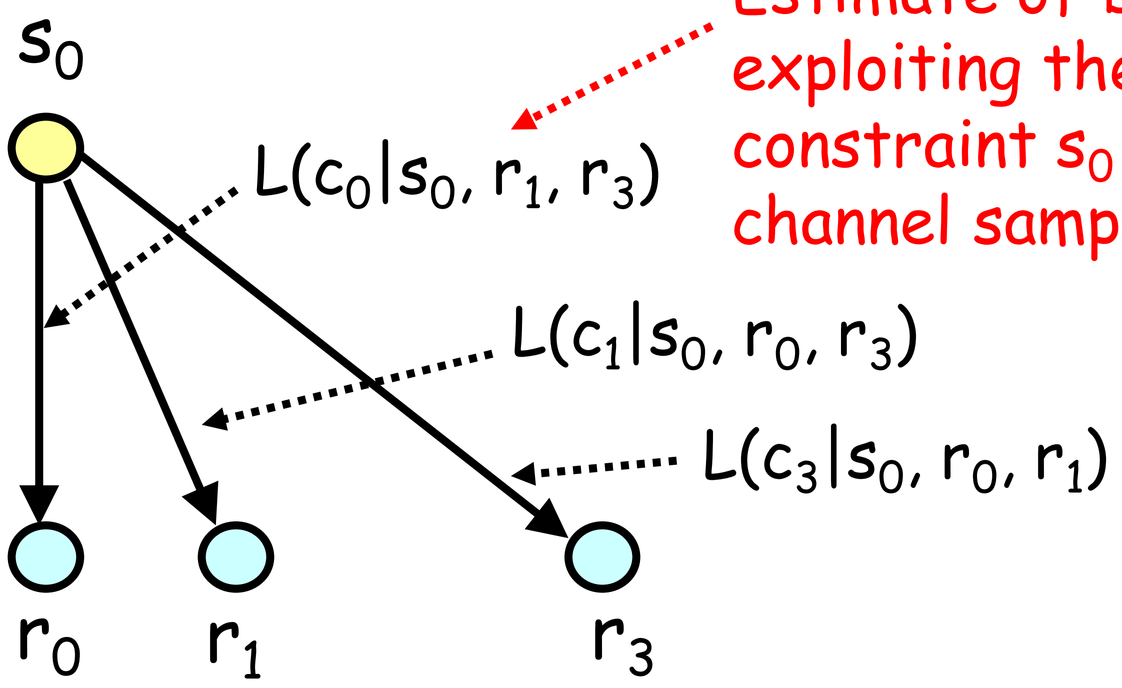
Each check node computes new estimates of each input bit based on the node constraint and the estimates $r_l,\ l \in \{0, 1, 2, 3, 4, 5\}$, of the other coded bits.

Constraint s_0 : $c_0 + c_1 + c_3 = 0$
Info available: channel samples r_0, r_1, r_3



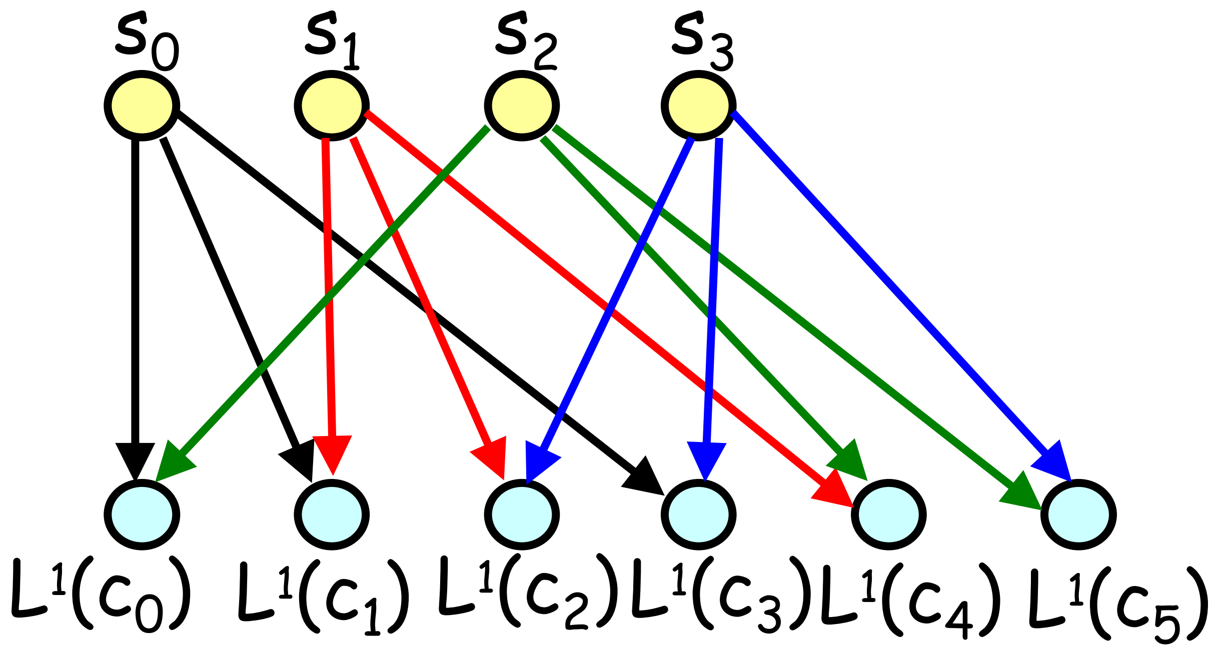
The rediscovery of LDPC codes

The new estimates of each coded bit are sent back to the relevant bit nodes.



Estimate of bit c_0 computed by exploiting the knowledge of the constraint s_0 as well as that of channel samples r_1 and r_3

The rediscovery of LDPC codes

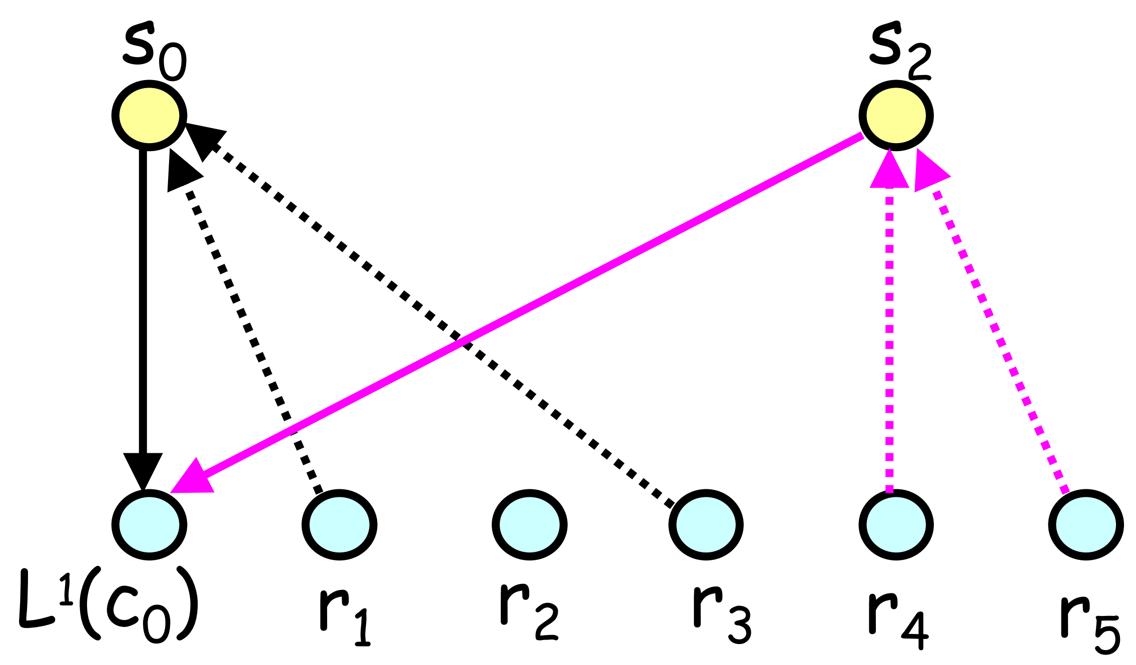


The bit node values are updated using info coming back from the check nodes.

The rediscovery of LDPC codes

For instance, the updated estimate $L^1(c_0)$ of the coded bit c_0 is given by

$$L^1(c_0) = r_0 + L(c_0|s_0, r_1, r_3) + L(c_0|s_2, r_4, r_5).$$



The rediscovery of LDPC codes

Since $L^1(c_0)$ is the combination of three independent estimates, it tends to be more reliable than the original channel sample r_0 alone.

We now have a vector

$$R^1 = \left(L^1(c_0), L^1(c_1), L^1(c_2), L^1(c_3), L^1(c_4), L^1(c_5)\right)$$

that constitutes a new (and hopefully improved) estimate of the transmitted codeword $C = (c_0, c_1, c_2, c_3, c_4, c_5)$.

A hard decision is taken on this vector, yielding the binary word $\hat{R}^1$, and the syndrome vector $S^1 = \hat{R}^1 \cdot H^T$ is computed once again.

The rediscovery of LDPC codes

• If $S^1 = \mathbf{0}$, we conclude that $\hat{R}^1$ is a valid codeword and the decoding process can be stopped as $\hat{R}^1$ is the decoded codeword.

• If $S^1 \neq \mathbf{0}$, $\hat{R}^1$ is not a valid codeword and the previous decoding procedure must be repeated again. In other words, we need to go for a second decoding iteration.

To show how it is done, consider once again the example of the first check node s_0.

The rediscovery of LDPC codes

The estimate $L^1(c_0) - L(c_0|s_0, r_1, r_3)$ is sent back to the check node s_0.

Why subtracting the term $L(c_0|s_0, r_1, r_3)$?

We must NOT send directly $L^1(c_0)$ back to s_0 because $L^1(c_0)$ contains some info, represented by the term $L(c_0|s_0, r_1, r_3)$, which has already been produced by s_0.

Removing the contribution of the info coming from s_0 from the estimate $L^1(c_0)$ simply consists of subtracting the term $L(c_0|s_0, r_1, r_3)$ from $L^1(c_0)$.

The rediscovery of LDPC codes

In iterative decoding, one must NOT send to a node some info that was previously generated by this node.

The estimate $L^1(c_0) - L(c_0|s_0, r_1, r_3)$ that is sent back to the check node s_0 is actually the sum of the channel sample r_0 and a quantity $L^{1,ext}(c_0) = L(c_0|s_2, r_4, r_5)$ called "extrinsic info".

The extrinsic info $L^{1,ext}(c_0)$ represents the additional info regarding the value of c_0 that has been gained during the first decoding iteration.

The rediscovery of LDPC codes

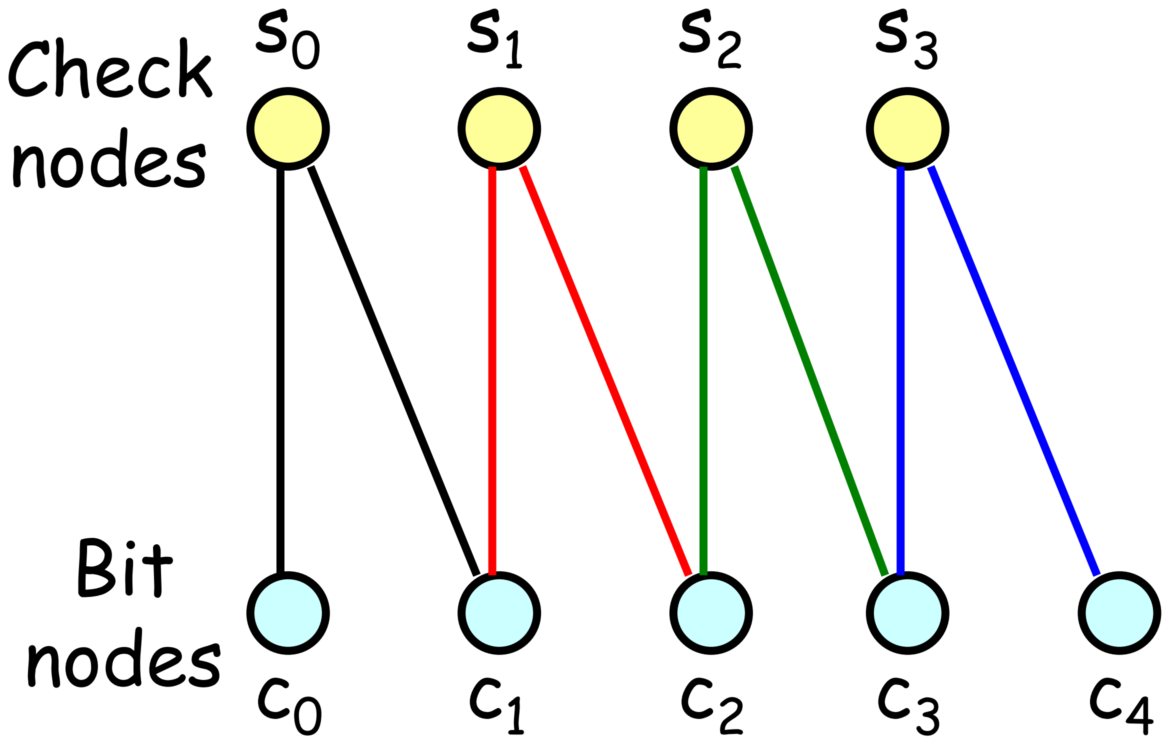
In iterative decoding, the extrinsic info is what allows the performance of the decoding algorithm to improve from one iteration to the next, until eventually near-ML performance is achieved.

We are now going to consider a simple example to illustrate the operation of the iterative decoding algorithm.

Consider a rate-1/5, length-5 ($n = 5$) LDPC code with the following parity-check matrix and Tanner graph.

LDPC codes - An example

$$H = \begin{bmatrix} 11000 \\ 01100 \\ 00110 \\ 00011 \end{bmatrix}$$



LDPC codes – An example

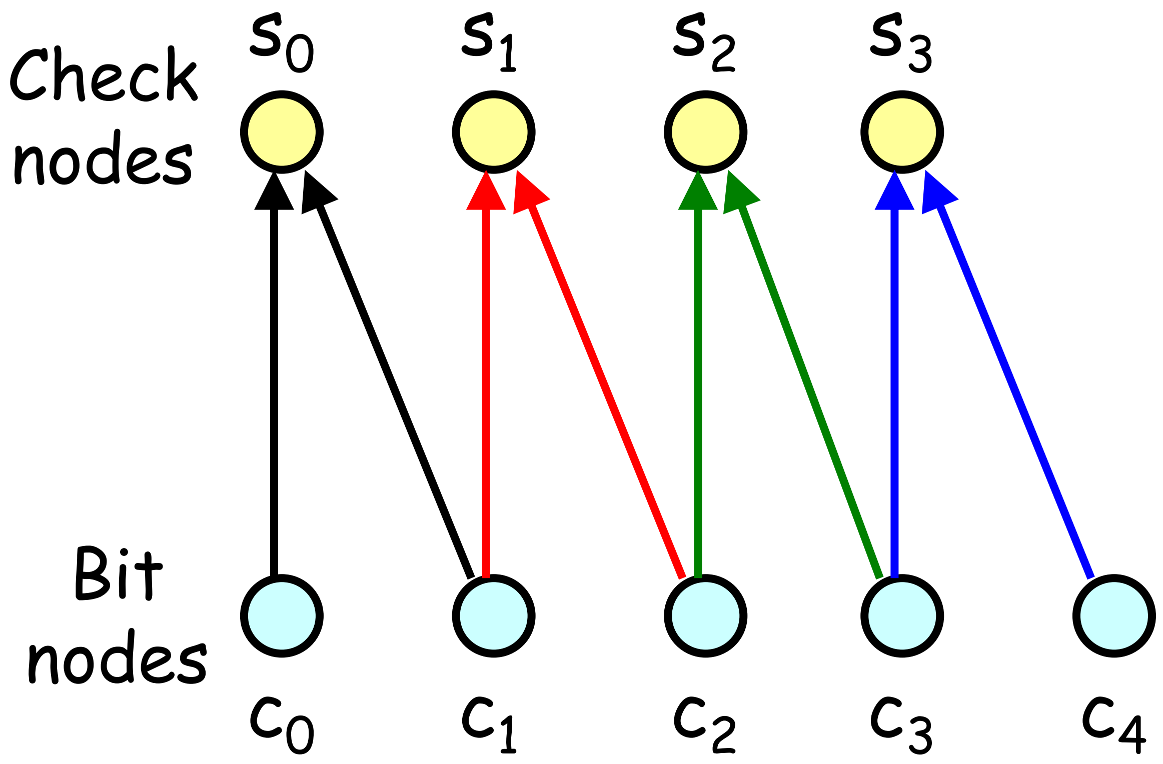
Assume that the word $R = (r_0, r_1, r_2, r_3, r_4) = (+1.0, +1.0, -2.0, +2.0, -1.0)$, is available at the BPSK, AWGN channel output.

After hard decision, the word R corresponds to the binary word $\hat{R} = (+1, +1, -1, +1, -1) \equiv (11010)$. We see that $\hat{R}$ is not a valid codeword because

$$\hat{R} \cdot H^T = (11010) \cdot \begin{bmatrix} 1000 \\ 1100 \\ 0110 \\ 0011 \\ 0001 \end{bmatrix} = (0111) \neq (0000).$$

LDPC codes - An example

First decoding iteration: the channel samples are sent to the relevant check nodes.



LDPC codes – An example

How to update the bit nodes in our example ?

A check node s_l, $l \in \{0, 1, 2, 3\}$, is associated with the constraint $c_l + c_{l+1} = 0$, which is another way of saying that the node constraint is $c_l = c_{l+1}$.

The quantity $L(c_l|s_l, r_{l+1})$ is calculated using the log-likelihood ratio (LLR) technique as follows:

$$L(c_l|s_l, r_{l+1}) = \frac{\sigma^2}{2} \cdot \ln\left(\frac{\Pr(c_l = +1|s_l, r_{l+1})}{Pr(c_l = -1|s_l, r_{l+1})}\right) = \frac{\sigma^2}{2} \cdot \ln\left(\frac{\Pr(c_{l+1} = +1|r_{l+1})}{Pr(c_{l+1} = -1|r_{l+1})}\right)$$

.

LDPC codes – An example

The expression can be further developed as follows:

$$L(c_l|s_l, r_{l+1}) = \frac{\sigma^2}{2} \cdot \ln\left(\frac{\Pr(c_{l+1}=+1; r_{l+1})}{\Pr(c_{l+1}=-1; r_{l+1})}\right)$$

$\Rightarrow L(c_l|s_l, r_{l+1}) = \frac{\sigma^2}{2} \cdot \ln\left(\frac{\Pr(r_{l+1}|c_{l+1} = +1)}{\Pr(r_{l+1}|c_{l+1} = -1)}\right)$, assuming that $\Pr(c_{l+1} = +1) = \Pr(c_{l+1} = -1)$.

$$\Rightarrow L(c_l|s_l, r_{l+1}) = \frac{\sigma^2}{2} \cdot \ln\left(\frac{\exp\left(-\frac{(r_{l+1}-1)^2}{2\sigma^2}\right)}{\exp\left(-\frac{(r_{l+1}+1)^2}{2\sigma^2}\right)}\right)$$

$$\Rightarrow L(c_l|s_l, r_{l+1}) = \frac{\sigma^2}{2} \cdot \left[-\frac{(r_{l+1}-1)^2}{2\sigma^2} + \frac{(r_{l+1}+1)^2}{2\sigma^2}\right] = r_{l+1}.$$

LDPC codes – An example

In the same way, we can show that $L(c_{l+1}|s_l, r_l\ ) = \ln\left(\frac{\Pr(c_{l+1} = +1|s_l, r_l)}{Pr(c_{l+1} = -1|s_l, r_l)}\right) = \ln\left(\frac{\Pr(c_l = +1|r_l)}{Pr(c_l = -1|r_l)}\right) = r_l.$

In summary, each check node s_l, $l \in \{0, 1, 2, 3\}$, generates two LLRs to be forwarded to the appropriate bit nodes as follows:
To bit node c_l: $L(c_l|s_l, r_{l+1}\ ) = r_{l+1}.$
To bit node c_{l+1}: $L(c_{l+1}|s_l, r_l\ ) = r_l.$

With three or more bits involved in the check nodes, equations would be much more complicated.

LDPC codes – An example

The check nodes generate the following estimates.

For node s_0 :
$L(c_0|s_0, r_1) = r_1 = +1.0$ and $L(c_1|s_0, r_0) = r_0 = +1.0.$
For node s_1 :
$L(c_1|s_1, r_2) = r_2 = -2.0$ and $L(c_2|s_1, r_1) = r_1 = +1.0.$
For node s_2 :
$L(c_2|s_2, r_3) = r_3 = +2.0$ and $L(c_3|s_2, r_2) = r_2 = -2.0.$
For node s_3 :
$L(c_3|s_3, r_4) = r_4 = -1.0$ and $L(c_4|s_3, r_3) = r_3 = +2.0.$

LDPC codes – An example

The estimates of each coded bit are then updated at the bit nodes:

$L^1(c_0) = r_0 + L(c_0|s_0, r_1) = r_0 + r_1 = +2.0.$
$L^1(c_1) = r_1 + L(c_1|s_1, r_2) + L(c_1|s_0, r_0) = r_1 + r_2 + r_0 = 0.0.$
$L^1(c_2) = r_2 + L(c_2|s_2, r_3) + L(c_2|s_1, r_1) = r_2 + r_3 + r_1 = +1.0.$
$L^1(c_3) = r_3 + L(c_3|s_3, r_4) + L(c_3|s_2, r_2) = r_3 + r_4 + r_2 = -1.0.$
$L^1(c_4) = r_4 + L(c_4|s_3, r_3) = r_4 + r_3 = +1.0.$

LDPC codes – An example

We have $R^1 = \left(L^1(c_0), L^1(c_1), L^1(c_2), L^1(c_3), L^1(c_4)\right) = (+2.0, +0.0, +1.0, -1.0, +1.0)$.

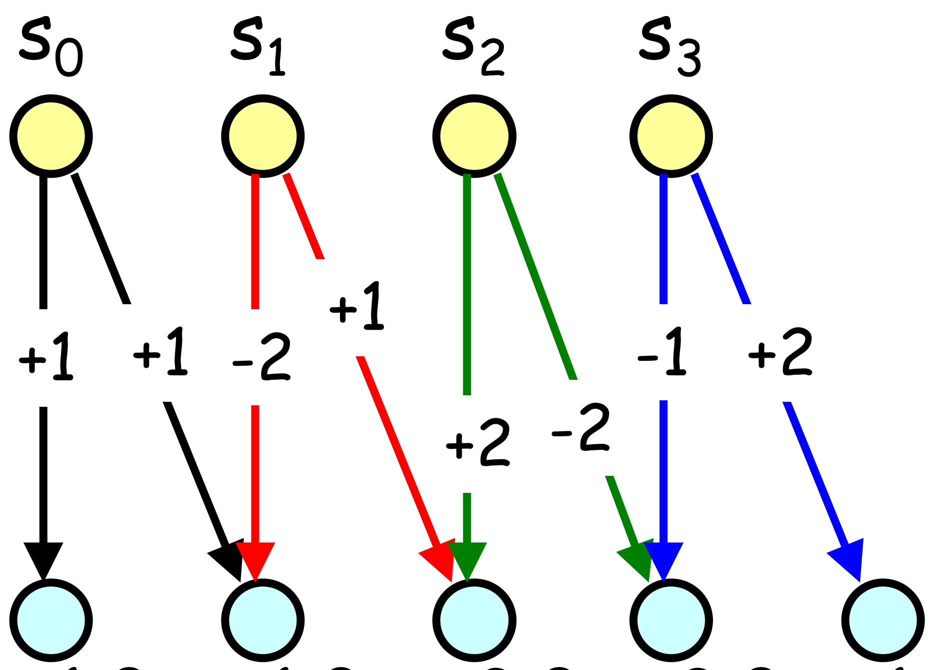
The binary word $\hat{R}^1 = (+1, -1, +1, -1, +1) \equiv (10101)$ obtained after hard decision is <u>not</u> a valid codeword since

$$\hat{R}^1 \cdot H^T = (10101) \cdot \begin{bmatrix} 1000 \\ 1100 \\ 0110 \\ 0011 \\ 0001 \end{bmatrix} = (1111) \neq (0000).$$

We thus need to perform a second decoding iteration.

LDPC codes - An example

Iteration 1



Initial values:	+1.0	+1.0	-2.0	+2.0	-1.0
Updated values:	+2.0	0.0	+1.0	-1.0	+1.0
Hard decisions:	1	0*	1	0	1

* for instance

LDPC codes – An example

The first step in the second iteration consists of forwarding to the appropriate check nodes the new estimates of the coded bits.

These new estimates are obtained by subtracting the previous contribution of the check nodes from the quantities $L^1(c_l)$, $l \in \{0, 1, 2, 3, 4\}$, as shown below. These estimates are in fact the sum of the original channel samples and the extrinsic info.

To check node s_0:

$r_{0,0} = L^1(c_0) - L(c_0|s_0, r_1) = +1.0.$

$r_{1,0} = L^1(c_1) - L(c_1|s_0, r_0) = -1.0.$

LDPC codes – An example

To check node s_1:

$r_{1,1} = L^1(c_1) - L(c_1|s_1, r_2) = +2.0.$

$r_{2,1} = L^1(c_2) - L(c_2|s_1, r_1) = 0.0.$

To check node s_2:

$r_{2,2} = L^1(c_2) - L(c_2|s_2, r_3) = -1.0.$

$r_{3,2} = L^1(c_3) - L(c_3|s_2, r_2) = +1.0.$

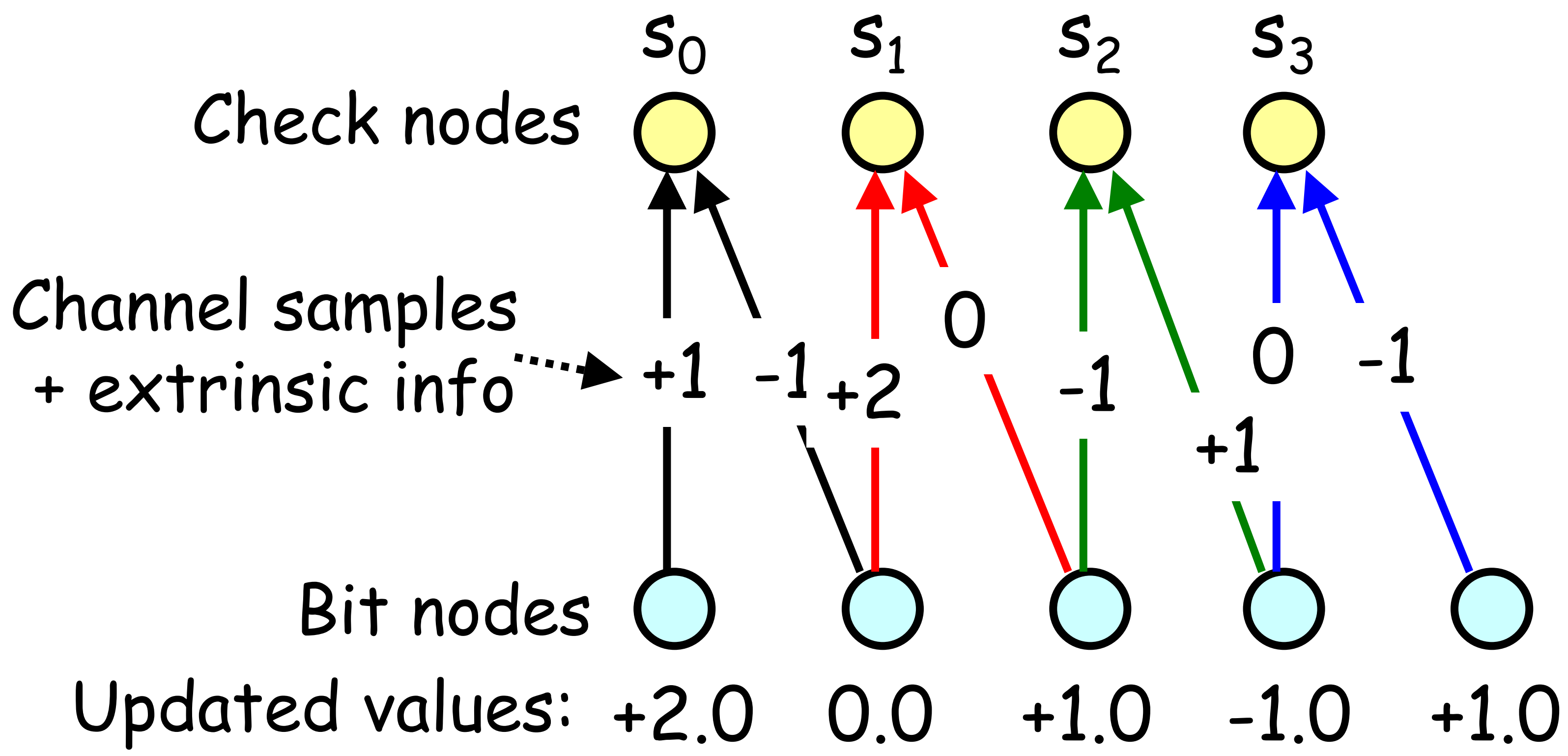
To check node s_3:

$r_{3,3} = L^1(c_3) - L(c_3|s_3, r_4) = 0.0.$

$r_{4,3} = L^1(c_4) - L(c_4|s_3, r_3) = -1.0.$

LDPC codes - An example

Iteration 2



LDPC codes – An example

The check nodes generate the following estimates to be sent to the appropriate bit nodes.

For node s_0 :
$L(c_0|s_0, r_{1,0}) = r_{1,0} = -1.0$ and $L(c_1|s_0, r_{0,0}) = r_{0,0} = +1.0.$
For node s_1 :
$L(c_1|s_1, r_{2,1}) = r_{2,1} = 0.0$ and $L(c_2|s_1, r_{1,1}) = r_{1,1} = +2.0.$
For node s_2 :
$L(c_2|s_2, r_{3,2}) = r_{3,2} = +1.0$ and $L(c_3|s_2, r_{2,2}) = r_{2,2} = -1.0.$
For node s_3 :
$L(c_3|s_3, r_{4,3}) = r_{4,3} = -1.0$ and $L(c_4|s_3, r_{3,3}) = r_{3,3} = 0.0.$

LDPC codes – An example

The estimates of each coded bit are then updated at the bit nodes:

$L^2(c_0) = r_0 + L\left(c_0 \middle| s_0, r_{1,0}\right) = 0.0.$

$L^2(c_1) = r_1 + L\left(c_1 \middle| s_1, r_{2,1}\right) + L\left(c_1 \middle| s_0, r_{0,0}\right) = +2.0.$

$L^2(c_2) = r_2 + L\left(c_2 \middle| s_2, r_{3,2}\right) + L\left(c_2 \middle| s_1, r_{1,1}\right) = +1.0.$

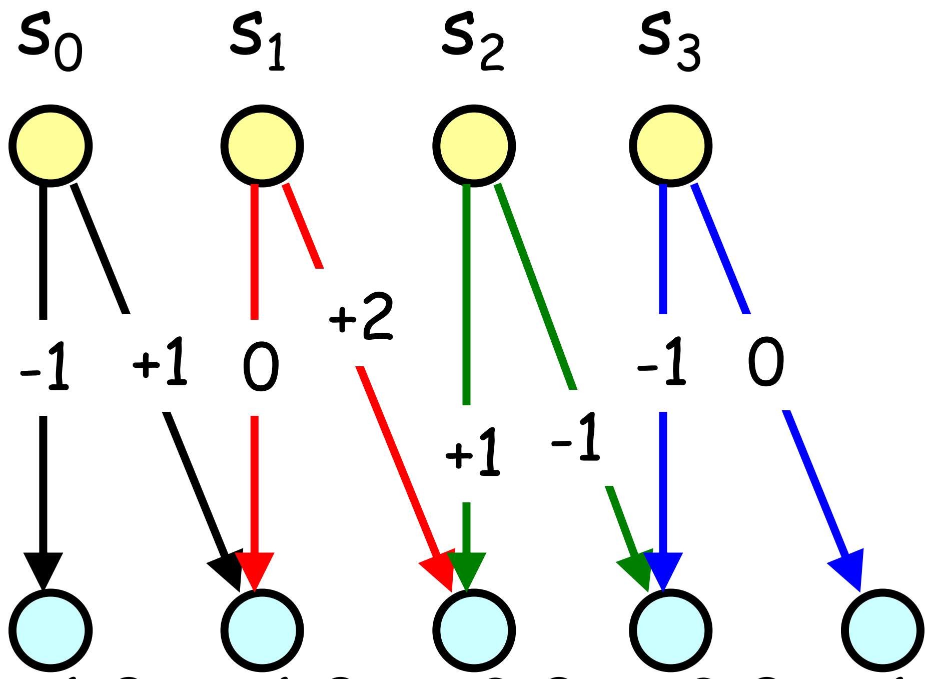
$L^2(c_3) = r_3 + L\left(c_3 \middle| s_3, r_{4,3}\right) + L\left(c_3 \middle| s_2, r_{2,2}\right) = 0.0.$

$L^2(c_4) = r_4 + L\left(c_4 \middle| s_3, r_{3,3}\right) = -1.0.$

We have $R^2 = \left(L^2(c_0), L^2(c_1), L^2(c_2), L^2(c_3), L^2(c_4)\right) = (0.0, +2.0, +1.0, 0.0, -1.0).$

LDPC codes - An example

Iteration 2



Initial values:	+1.0	+1.0	-2.0	+2.0	-1.0
Updated values:	0.0	+2.0	+1.0	0.0	-1.0
Updated decisions:	0	1	1	0	0

LDPC codes – An example

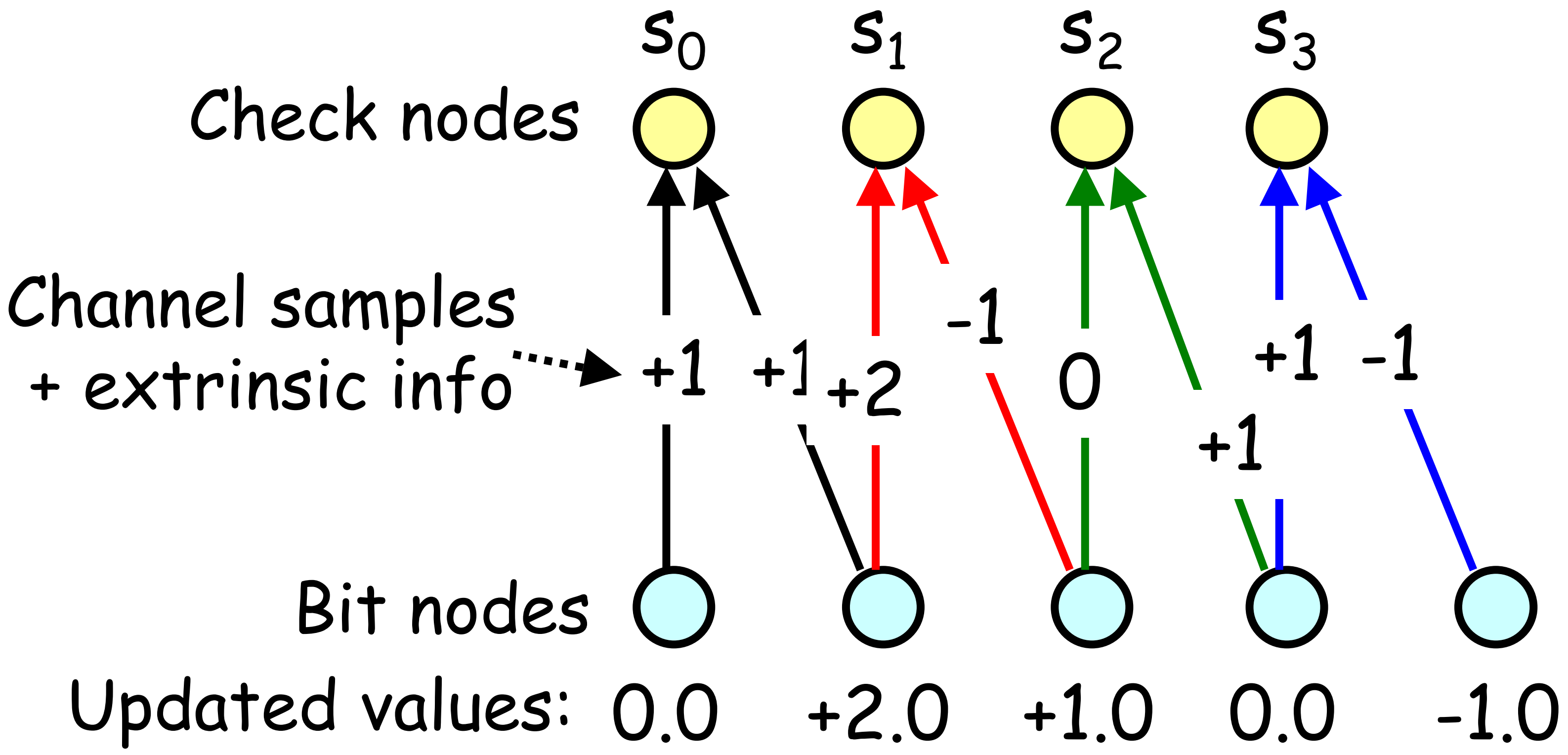
The binary word $\hat{R}^2 = (-1,+1,+1,-1,-1) \equiv (01100)$ obtained after hard decision is <u>not</u> a valid codeword since

$$\hat{R}^2 \cdot H^T = (01100) \cdot \begin{bmatrix} 1000 \\ 1100 \\ 0110 \\ 0011 \\ 0001 \end{bmatrix} = (1010) \neq (0000).$$

We need to perform a third decoding iteration.

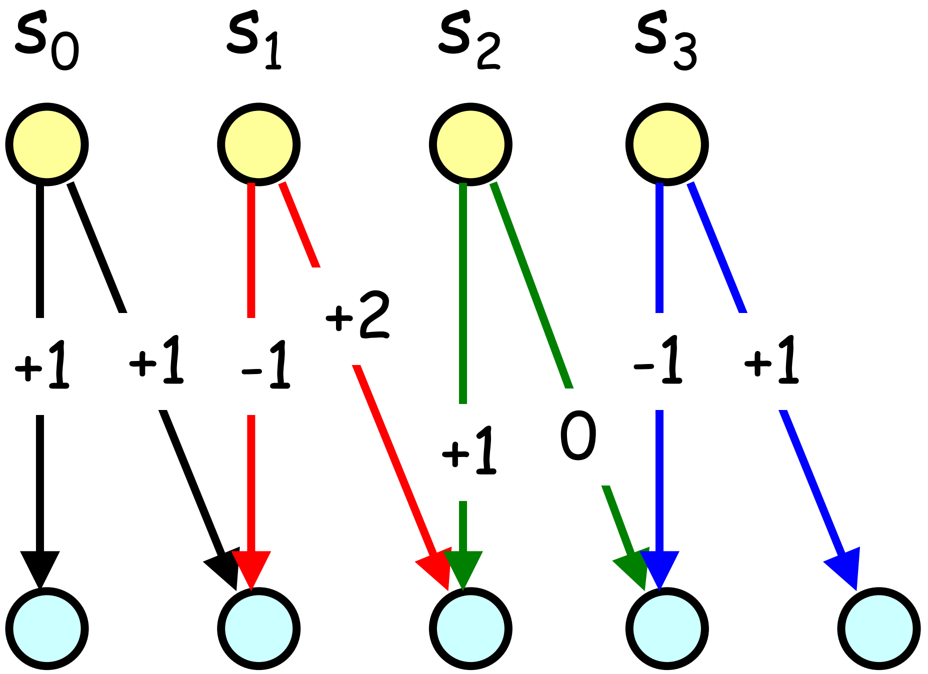
LDPC codes - An example

Iteration 3



LDPC codes - An example

Iteration 3



Initial values:	+1.0	+1.0	-2.0	+2.0	-1.0
Updated values:	+2.0	+1.0	+1.0	+1.0	0.0
Updated decisions:	1	1	1	1	0

LDPC codes – An example

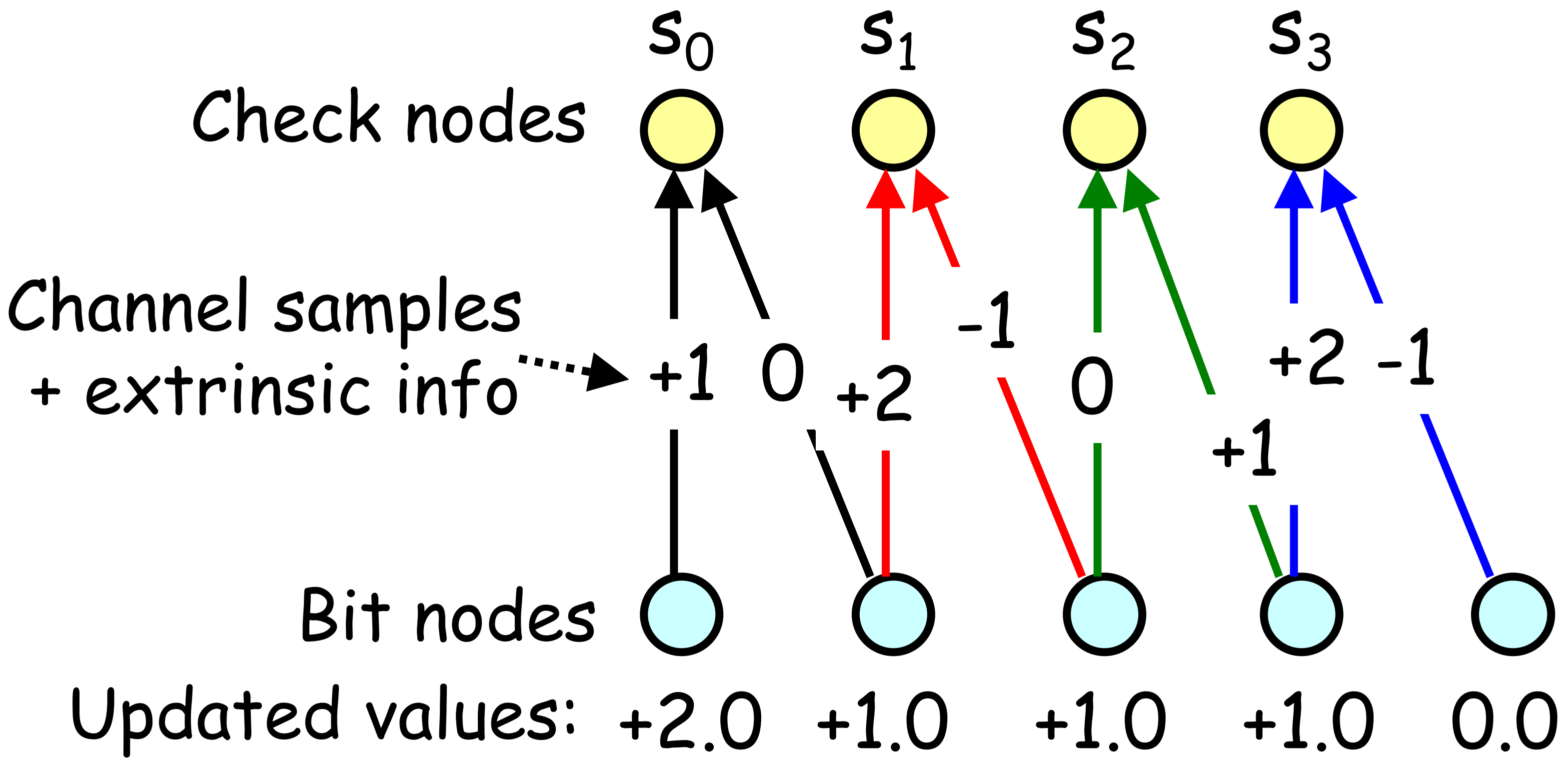
The binary word $\hat{R}^3 = (+1, +1, +1, +1, -1) \equiv (11110)$ obtained after hard decision is not a valid codeword since

$$\hat{R}^2 \cdot H^T = (11110) \cdot \begin{bmatrix} 1000 \\ 1100 \\ 0110 \\ 0011 \\ 0001 \end{bmatrix} = (0001) \neq (0000).$$

We need to perform a fourth decoding iteration.

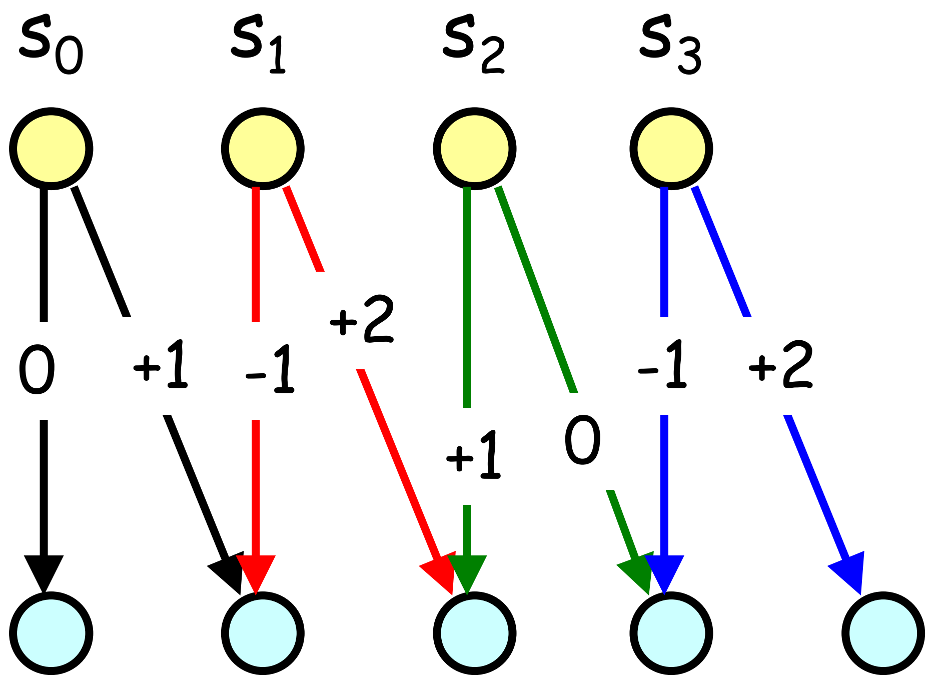
LDPC codes - An example

Iteration 4



LDPC codes - An example

Iteration 4



Initial values:	+1.0	+1.0	-2.0	+2.0	-1.0
Updated values:	+1.0	+1.0	+1.0	+1.0	+1.0
Updated decisions:	1	1	1	1	1

LDPC codes – An example

The binary word $\hat{R}^4 = (+1, +1, +1, +1, +1) \equiv (11111)$ obtained after hard decision is a valid codeword since

$$\hat{R}^4 \cdot H^T = (11111) \cdot \begin{bmatrix} 1000 \\ 1100 \\ 0110 \\ 0011 \\ 0001 \end{bmatrix} = (0000).$$

The decoding algorithm has finally converged after four iterations. The decoder thus decides that the transmitted codeword was $C = (11111)$.

A big leap into the future

The last word for Robert MacEliece:

Claude Shannon – Born on the planet Earth (Sol III) in the year 1916 A.D. Generally regarded as the father of the Information Age, he formulated the notion of channel capacity in 1948 A.D. Within several decades, mathematicians and engineers had devised practical ways to communicate reliably at data rates within 1% of the Shannon limit…

Encyclopedia Galactica, 166th edition