Introduction to Low-Density Parity Check Codes

Brian Kurkoski
kurkoski@ice.uec.ac.jp

Outline: Low Density Parity Check Codes

Review block codes

History

Low Density Parity Check Codes

Gallager's LDPC code construction

Bipartite Graphs — the Tanner Graph

Overview of decoding algorithms

- Bit Flipping decoding
- Soldier Counting problem
- Message passing decoding

The Communications Problem

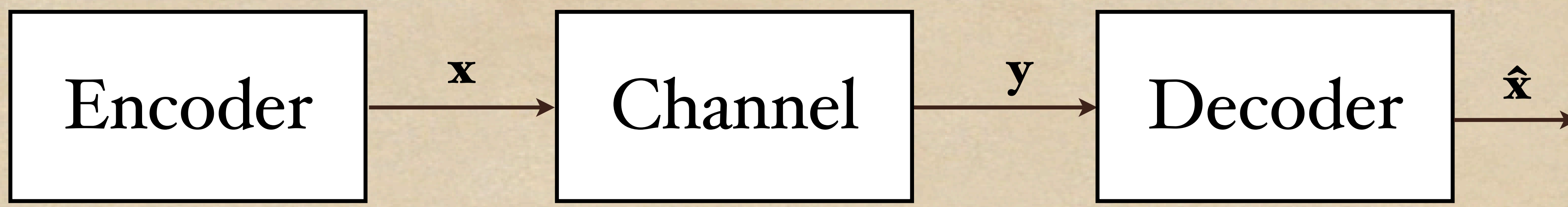


Figure 1.

The problem is to communicate reliably over a noisy channel.

Let **x** be the transmitted data, and **y** be the received sequence:

$$\mathbf{x} = (1\ 0\ 0\ 0) \rightarrow \mathbf{y} = (1\ 0\ 0\ 1).$$

The approach to solve the problem is to add redundancy:

$$\mathbf{x} = (\underbrace{1\ 0\ 0\ 0}_{K}\ \underbrace{1\ 1\ 1}_{N-K}) \rightarrow \mathbf{y} = (1\ 0\ 0\ 1\ 1\ 1\ 1) \rightarrow \hat{\mathbf{x}} = (1\ 0\ 0\ 0\ 1\ 1\ 1).$$

Using this redundancy, the decoder can estimate the original data.

The Communications Problem

The problem is solved by using error-correcting codes, ECC. Consider a major class of ECC: linear block codes over the binary field F_2.

Block code has length N,

and has 2^K codewords which form a subspace of $(F_2)^N$.

The code's rate is $R = K/N$.

A block code C is defined by a $(N\text{-}K)$-by-N parity check matrix H. The code is the set of length-N vectors $\mathbf{x} = (x_1\ x_2\ \ldots\ x_N)$ such that:

$$\mathbf{x} \cdot H^t = \underline{0}$$

Let $\mathbf{u} = (u_1\ u_2\ \ldots\ u_K)$ be a length K vector of information. Let G be a K-by-N generator matrix for C:

$$\mathbf{u}G = \mathbf{x}.$$

Arithmetic is performed over the binary field F_2.

+	0	1
0	0	1
1	1	0

All codes have an important property, the minimum distance.

!	0	1
0	0	0
1	0	1

Parity Check Matrices

A binary, linear error-corrrecting code can be defined by a $(N-K)$-by-N parity-check matrix, H. Each row in this matrix is h_i:

$$H = \begin{bmatrix} — & h_1 & — \\ — & h_2 & — \\ & \vdots & \\ — & h_{N-K} & — \end{bmatrix}$$

For example, the (7,4) Hamming code has:

$$H = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

This code has block length N=7, and information length K=4. The codebook, C, is the set of length N words, $\mathbf{x}$, which satisfy:

$$\mathbf{x} \cdot H^t = \underline{0}$$

Codes Defined by Parity Check Matrices

The (7,4) Hamming codebook has 2^4=16 codewords:

C = {0000000, 1000111, 0100110,...}

It is easy to check if a length N vector, $\mathbf{y} = (y_1\, y_2\, \ldots\, y_N)$ is a codeword:

$$\mathbf{y}H^t = (y_1\ y_2 \ldots y_N) \cdot \begin{bmatrix} - & h_1 & - \\ - & h_2 & - \\ & \vdots & \\ - & h_{N-K} & - \end{bmatrix}^t = \underline{0} \Rightarrow \mathbf{y} \text{ is a codeword}$$

Example. For the (7,4) Hamming code:

h_1: $y_1 + y_3 + y_4 + y_5 = 0$

h_2: $y_1 + y_2 + y_4 + y_6 = 0$

h_3: $y_1 + y_2 + y_3 + y_4 + y_7 = 0$,

$$H = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Verify that $\mathbf{y}$ = (1 1 1 0 0 1 1) is a (7,4) Hamming codeword:

h_1: 1 + 1 + 0 + 0 = 0

h_2: 1 + 1 + 0 + 1 = 1

h_3: 1 + 1 + 1 + 0 + 1 = 0.

The Communications Problem

Given a received word $\mathbf{y} = (y_1\ y_2\ \ldots\ y_N)$, the decoder's goal is to find the maximum likelihood decision:

$$\widehat{\mathbf{x}} = \arg\max_{\mathbf{x}\in C} P(\mathbf{y}|\mathbf{x})$$

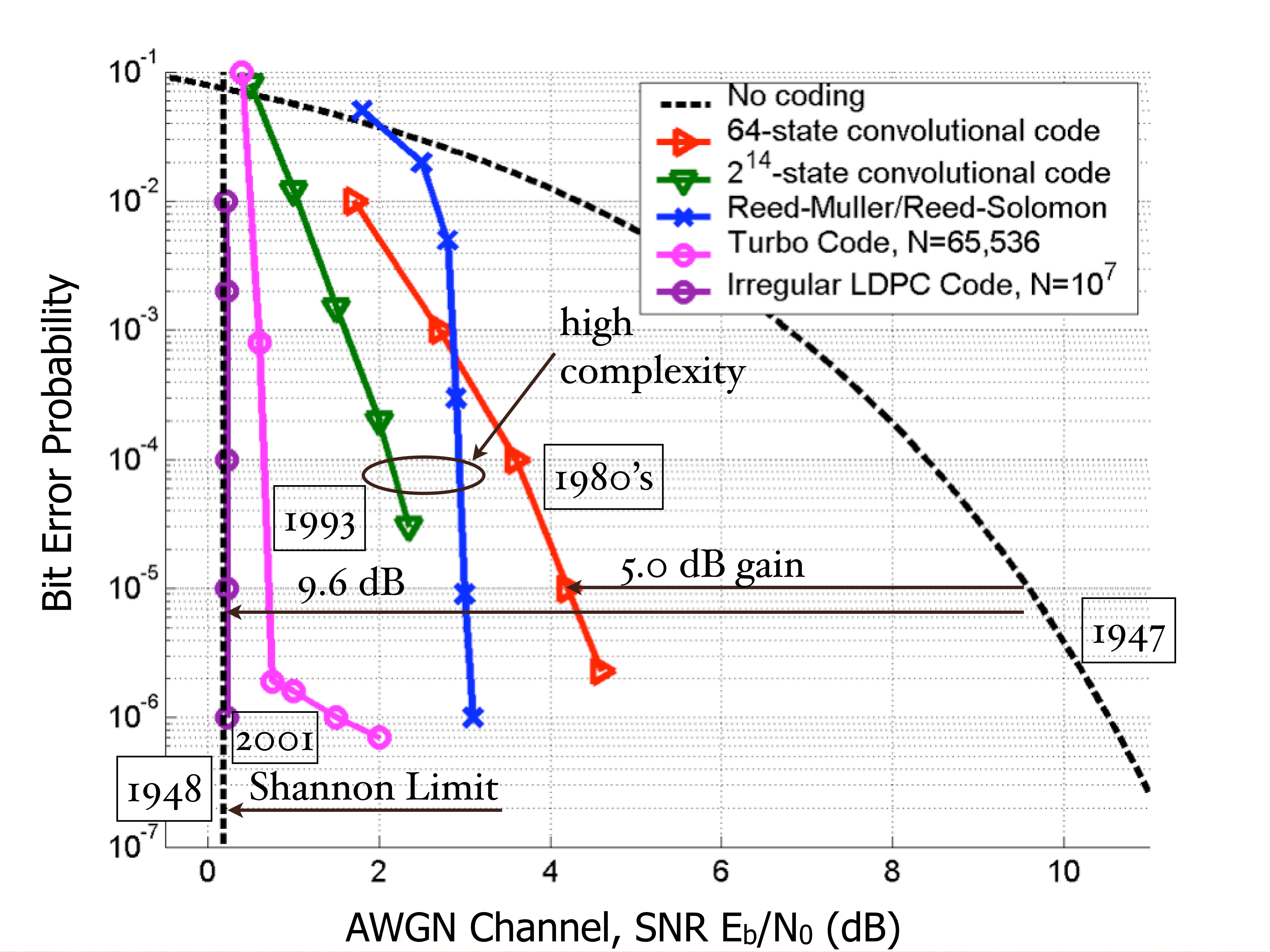
Decoder complexity is a serious restriction in using error correcting codes. It is impractical to evaluate the above equation directly, it is exponentially difficult.

Various types of codes:

- Reed-Solomon Codes
- BCH Codes
- Convolutional Codes

are used in practice not only because they are good codes, but because the decoders have reasonable complexity.

Reducing the Gap to Capacity. Rate R=1/2 Codes



History of Low-Density Parity Check Codes

1948 Shannon published his famous paper on the capacity of channels with noise

1963 Robert Gallager wrote his Ph.D. dissertation "Low Density Parity Check Codes". He introduced LDPC codes, analyzed them, and gave some decoding algorithms.

Because computers at that time were not very powerful, he could not verify that his codes could approach capacity

1982 Michael Tanner considered Gallager's LDPC codes, and his own structured codes. He introduced the notion of using bipartite graph, sometimes called a Tanner graph.

1993 Turbo codes were introduced. They exceeded the performance of all known codes, and had low decoding complexity

1995 Interest was renewed in Gallager's LDPC codes, lead by David MacKay and many others.

It was shown that LDPC codes can essentially achieve Shannon Capacity on AWGN and Binary Erasure Channels.

Low-Density Parity Check Codes

A **low-density parity check (LDPC) code** is a linear block code whose parity check matrix has a small number of one's.

$$H = \begin{bmatrix}
1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1\\
1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0\\
0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0&1&0&0&0\\
0&0&1&0&0&0&1&0&0&0&0&0&0&1&0&0&0&1&0&0\\
0&0&0&1&0&0&0&0&0&0&1&0&0&0&1&0&0&0&1&0\\
0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1\\
1&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0\\
0&1&0&0&0&0&1&0&0&0&1&0&0&0&0&1&0&0&0&0\\
0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&1&0\\
0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&1&0&0&0\\
0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1
\end{bmatrix}$$

The number of 1's in an LDPC code's matrix grows linearly with the size *N*.

The number of 1's in a regular-density matrix grows as N·(M-N).

LDPC Code Constructions

Note that *all* linear codes have a parity check matrix, and thus can be decoded using message-passing decoding.

However, not all codes are well-suited for this decoding algorithm.

Semi-random Construction

Regular LDPC Codes (1962, Gallager)

Irregular LDPC (Richardson and Urbanke, Luby et al.)

MacKay Codes

Structured Constructions

Finite-Geometry based constructions (Kou, Lin, Fossorier)

Quasicyclic LDPC Codes (Lin)

combinatorial LDPC codes (Vasic, et al.)

LDPC array codes (Fan)

Regular LDPC Codes

The parity check matrix for a **(*j*,*k*) regular LDPC Code** has j one's in each column, and k one's in each row.

Gallager's construction technique:

1. Code parameters N, K, j, k are given.
2. Construct the following matrix with $\frac{N-K}{j}$ rows and N columns:

$$H_1 = \begin{bmatrix} 1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1 \end{bmatrix}$$

3. Let $\pi(H_1)$ be a pseudo-random column permutation of H_1.
4. Construct regular LDPC check matrix by stacking j submatrices:

$$H = \begin{bmatrix} H_1 \\ \hline \pi(H_1) \\ \hline \pi(H_1) \end{bmatrix}$$

Regular LDPC Code Example

This code has N=20, K = 5, j=3, k=4.

$$H = \left[\begin{array}{cccccccccccccccccccc}
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
\hline
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
\hline
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1
\end{array}\right]$$

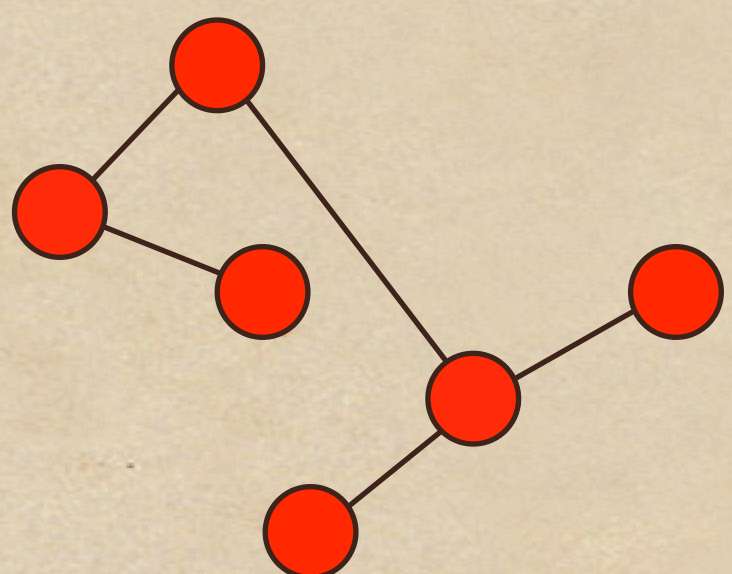
The number of ones is $N \cdot j$ = $(N-K) \cdot k$. From this, it is easy to show that the rate of the code is:

$$R = 1 - \frac{j}{k}$$

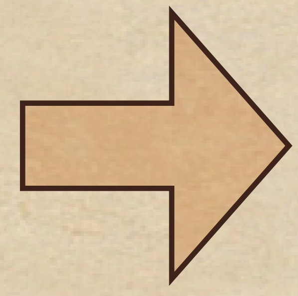
In this example, R = 1 - 3/4 = 1/4.

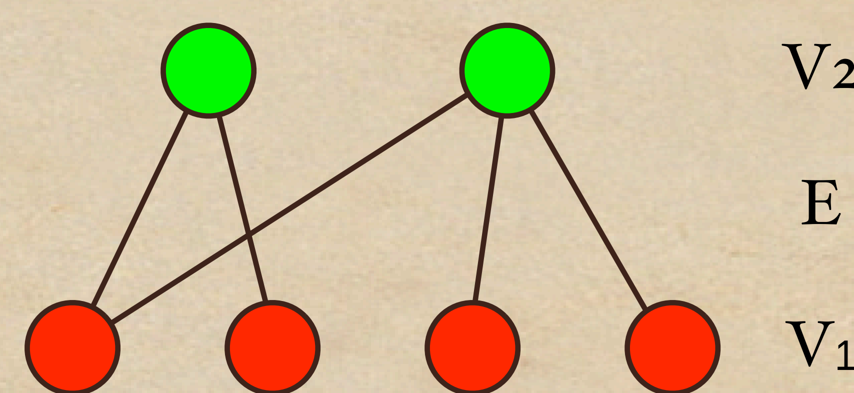
Bipartite Graphs

- A simple undirected graph G: = (V,E) is called a **bipartite graph** if there exists a partition of the vertex set so that both V_1 and V_2 are independent sets. We often write G: = ($V_1 + V_2$, E) to denote a bipartite graph with partitions V_1 and V_2.



Undirected Graph (V,E)





Bipartite Graph (V_1+V_2,E)

V, vertex set

E, edge set

V_1, V_2 node set

E, edge set

Graph Representations of Codes—Tanner Graph

A **Tanner Graph** is a bipartite graph which represents the parity check matrix of an error correcting code.

H is the $(N-K)$-by-N parity check matrix. The Tanner graph has:

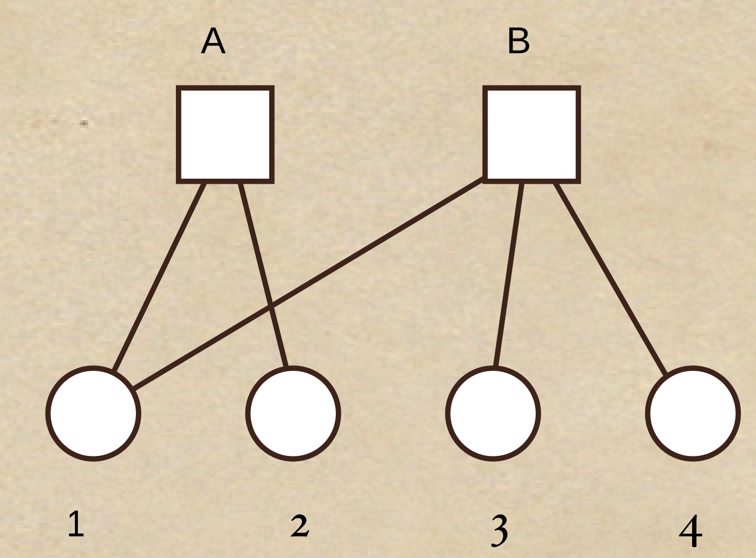
N **bit nodes** (or variable nodes), represented by circles.

$N-K$ **check nodes**, represented by squares.

There is an edge between bit node i and check node j if there is a one in row i and column j of H.

Example:

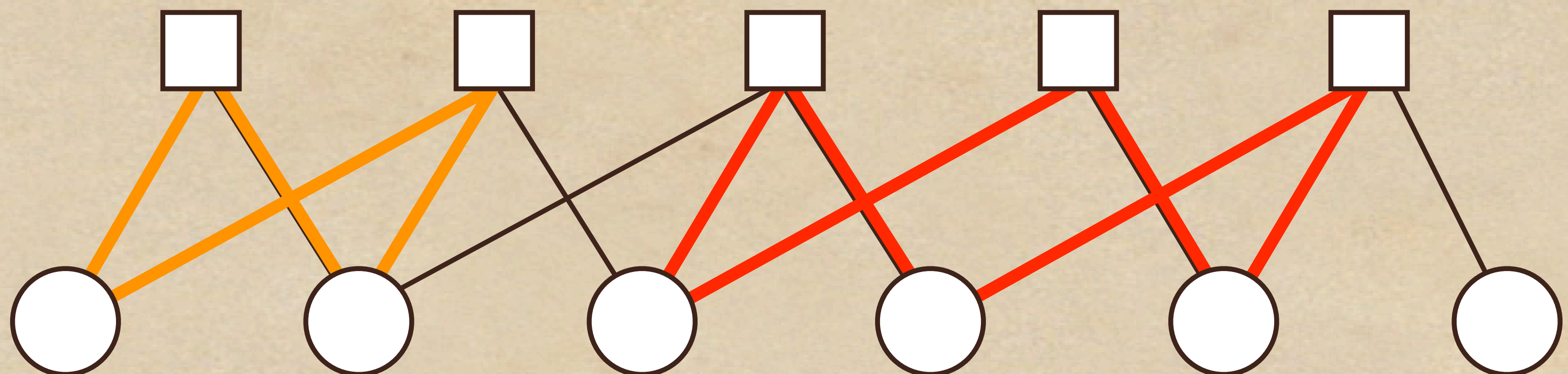
$$H = \begin{array}{cc} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} & \\ \left[\begin{array}{cccc} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \end{array}\right] & \begin{array}{c} \text{A} \\ \text{B} \end{array} \end{array}$$



Tanner Graph

Cycles and Girth of Tanner Graphs

Cycle
L=4
L=6



A ***cycle*** of length L in a Tanner graph is a path of L edges which closes back on itself

- The ***girth*** of a Tanner graph is the minimum cycle length of the graph.

$$H = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Decoding Algorithms

Gallager's Bit Flipping algorithm for LDPC codes

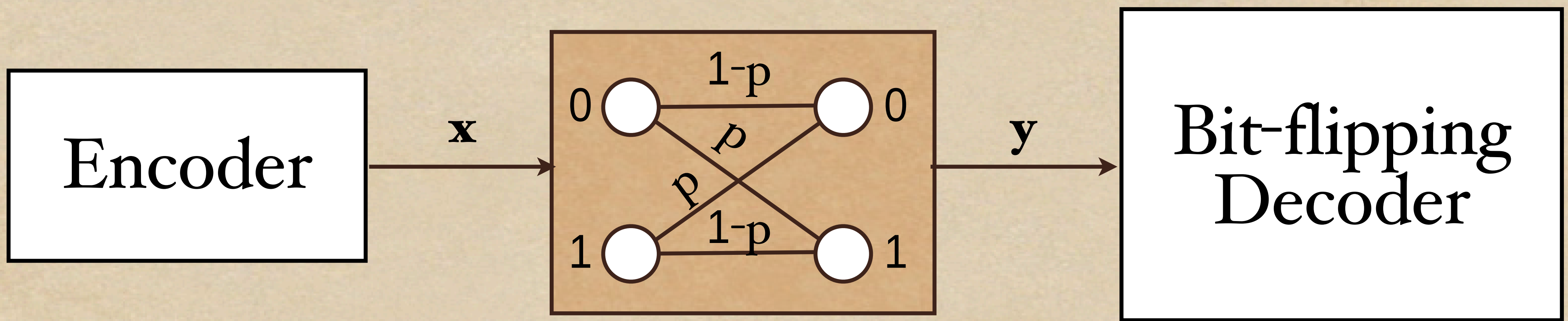
Message-passing algorithms:

- "Soldier Counting" algorithm
- Probablisitic Decoding of LDPC Codes.

Bit Flipping Decoding: Channel

Gallager's bit-flipping algorithm is for decoding on the binary symmetric channel (BSC).

The BSC has transition probability p.



Channel: BSC(p)

Transmitted Sequence

$\mathbf{x}$ = 0 0 0 0 0 0

Received Sequence

$\mathbf{y}$ = 0 0 1 0 1 0

$\mathbf{z}$ is the noise sequence: $\mathbf{y} = \mathbf{x} + \mathbf{z}$.

LDPC codes are linear codes: $\mathbf{x}_1 + \mathbf{x}_2$ is a codeword.

⇒ considering the all-zeros codeword is sufficient.

Bit Flipping Decoding: Example Code

Consider the following parity check matrix H:

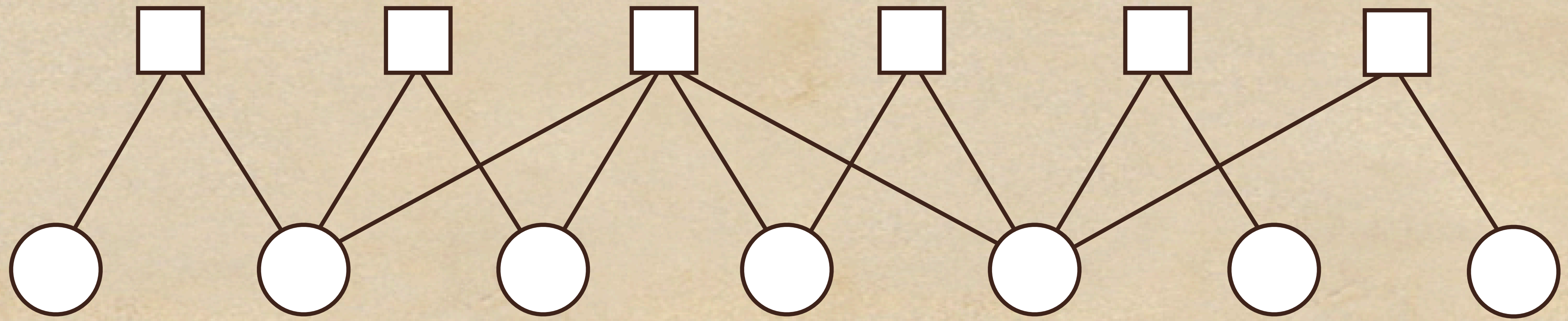
$$H = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

This code has N=7 bits, and K=6 parity checks.

It has rate R=6/7, and only two codewords {0000000,1111111}

(H is one possible parity check matrix for the repeat code)

The Tanner graph corresponding to the parity check matrix:

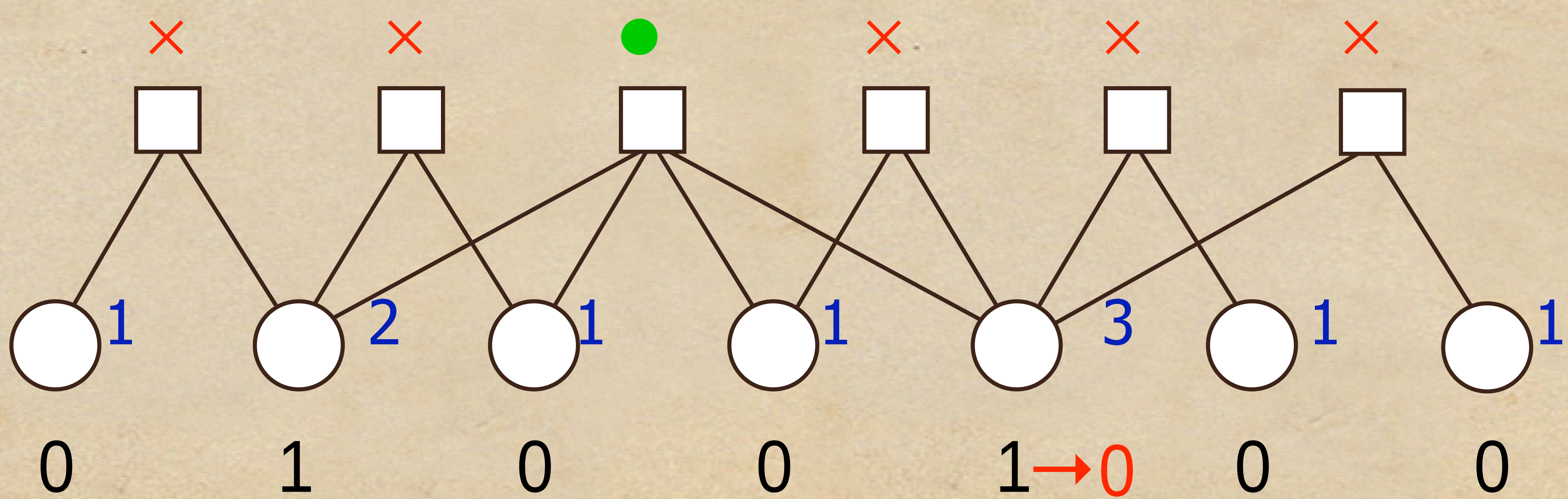


Bit Flipping Decoding: Decoding Algorithm

Gallager's Bit-Flipping Algorithm

1. Compute each parity check, for received **y**
2. For each bit, count the number of failing parity checks
3. For the bit(s) with the largest number of failed parity checks, flip the associated input y.
4. Repeat steps 1-3 until all the parity checks are satisfied, or a stopping condition is reached.

$\mathbf{x} = 0\,0\,0\,0\,0\,0\,0 \rightarrow \mathbf{y} = 0\,1\,0\,0\,1\,0\,0$

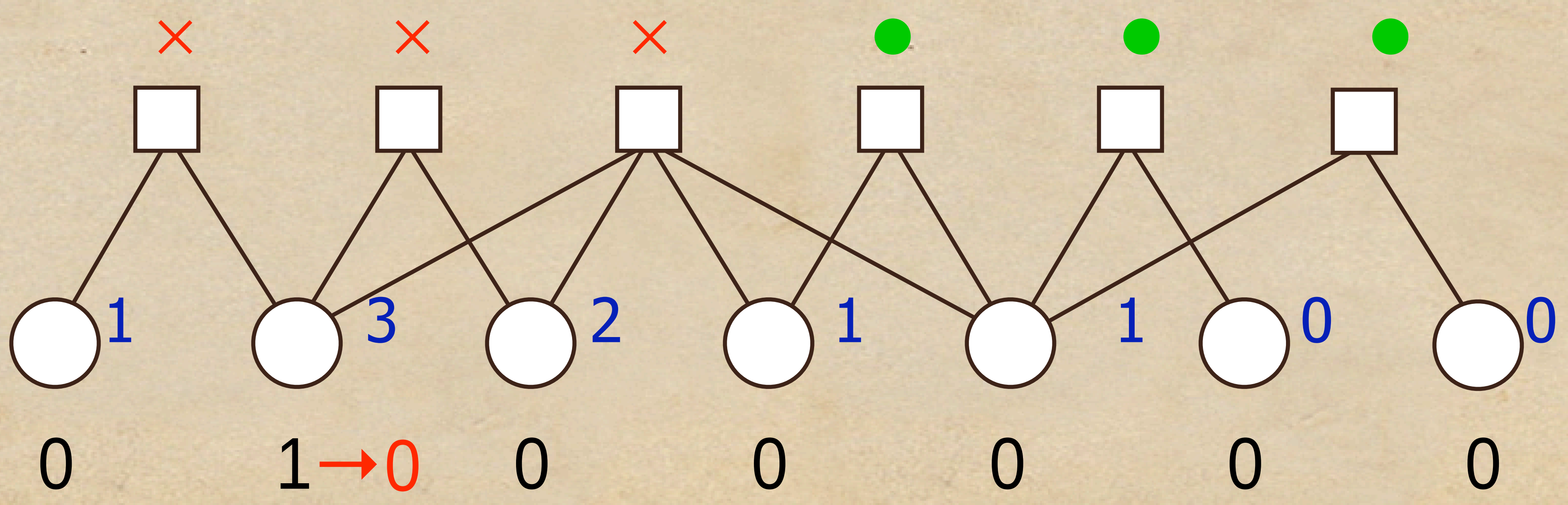


Bit Flipping Decoding: Decoding Algorithm

Gallager's Bit-Flipping Algorithm

1. Compute each parity check, for received **y**
2. For each bit, count the number of failing parity checks
3. For the bit(s) with the largest number of failed parity checks, flip the associated input *y*.
4. Repeat steps 1-3 until all the parity checks are satisfied, or a stopping condition is reached.

$\mathbf{x} = 0\,0\,0\,0\,0\,0\,0 \rightarrow \mathbf{y} = 0\,1\,0\,0\,1\,0\,0$

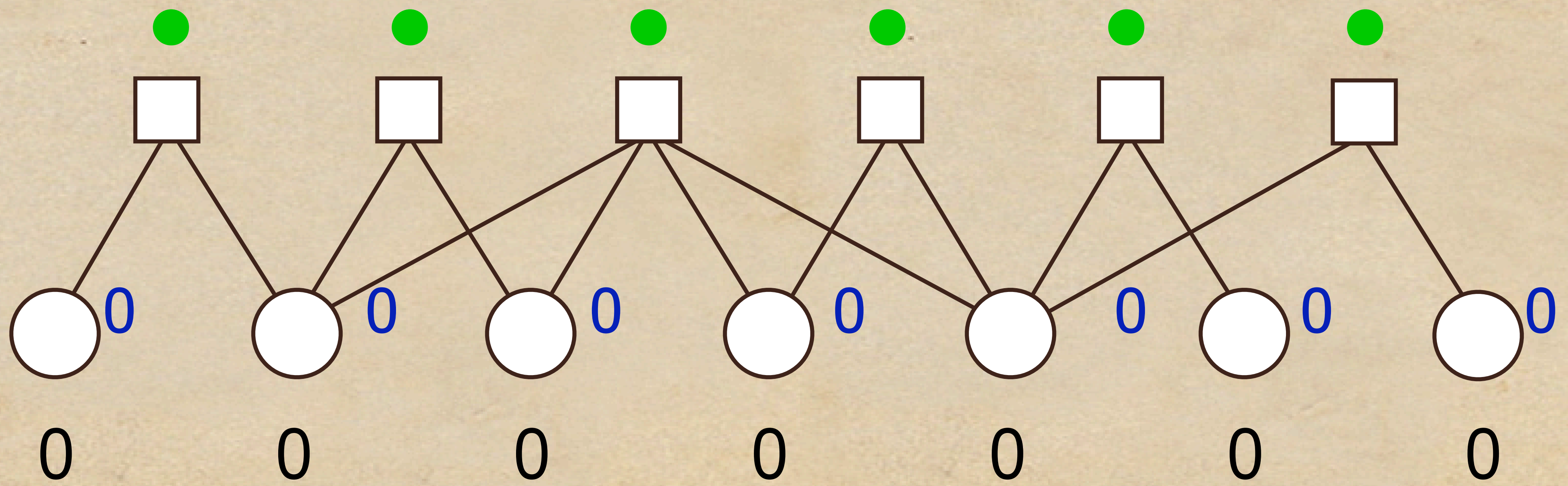


Bit Flipping Decoding: Decoding Algorithm

Gallager's Bit-Flipping Algorithm

1. Compute each parity check, for received $\mathbf{y}$
2. For each bit, count the number of failing parity checks
3. For the bit(s) with the largest number of failed parity checks, flip the associated input y.
4. Repeat steps 1-3 until all the parity checks are satisfied, or a stopping condition is reached.

$\mathbf{x} = 0\,0\,0\,0\,0\,0\,0 \rightarrow \mathbf{y} = 0\,0\,1\,0\,1\,0\,0$



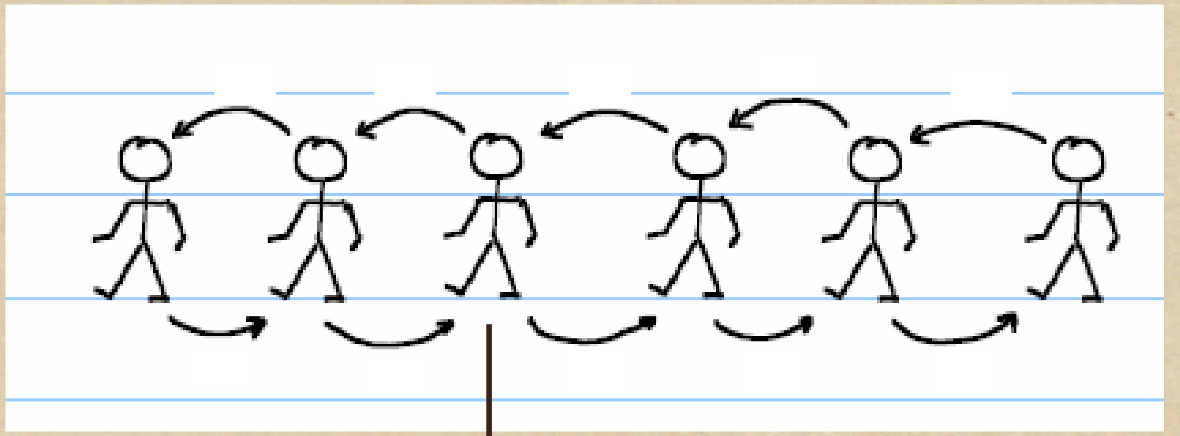
VALID CODEWORD!

Message Passing Problem, Soldier Counting

The Soldier Counting Problem: Each soldier in a row wants to know the total number of soldiers.

Each soldier can only communicate with his neighbors.

How to communicate the total number to each soldier?



2+3+1 = 6

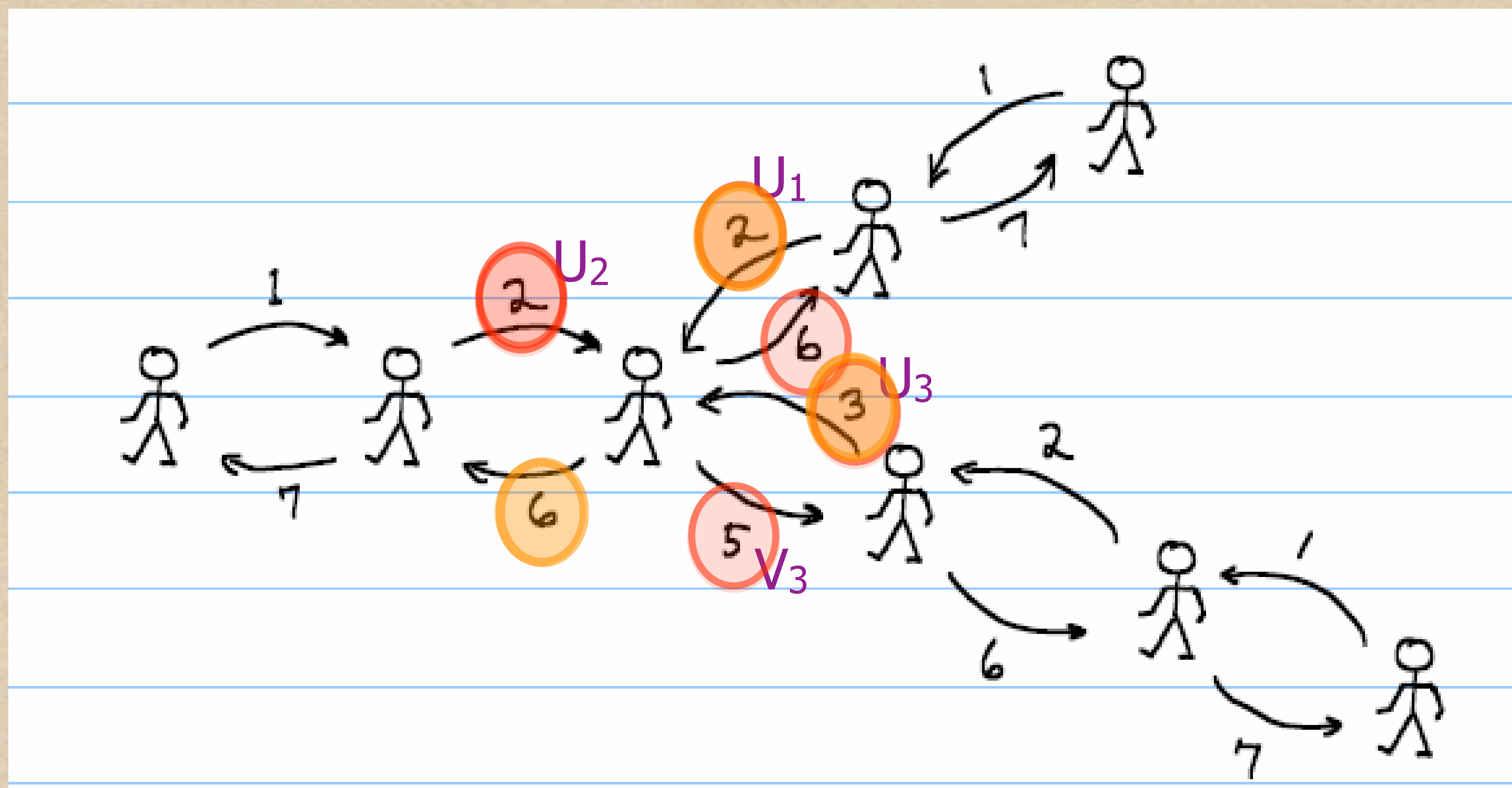
Solution: Message Passing.

1. When a soldier receives a number from his left, he adds one (for himself) and passes it to his left.
2. Similarly, for messages passing from the right.
3. A soldier with only one neighbor passes the number "one" to his neighbor.

Soldier figures due to William Ryan

Soldiers in a Y

For soldiers in more complicated formations, a solution is still possible.

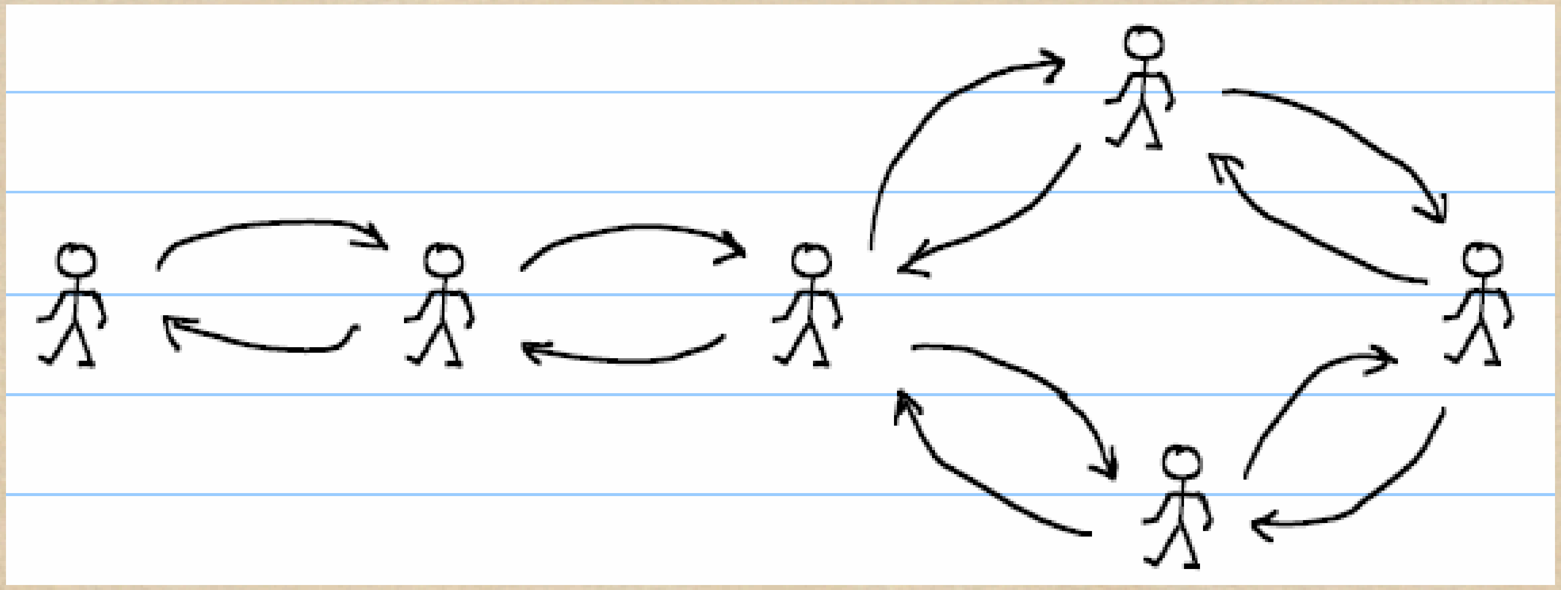


The soldier with three neighbors receives two messages U_1, U_2. The message that he sends on is $V_3 = U_1 + U_2 + 1$

Important: U_3 is not used in computing V_3. *Sum Product Update Rule.*

Soldiers in A Loop

For soldiers in a loop, there is no simple message-passing solution.



Message-Passing Decoding

Important decoding algorithm has various names:

Message-passing decoding

Sum-product decoding

Probabilistic decoding

An instance of "belief propagation"

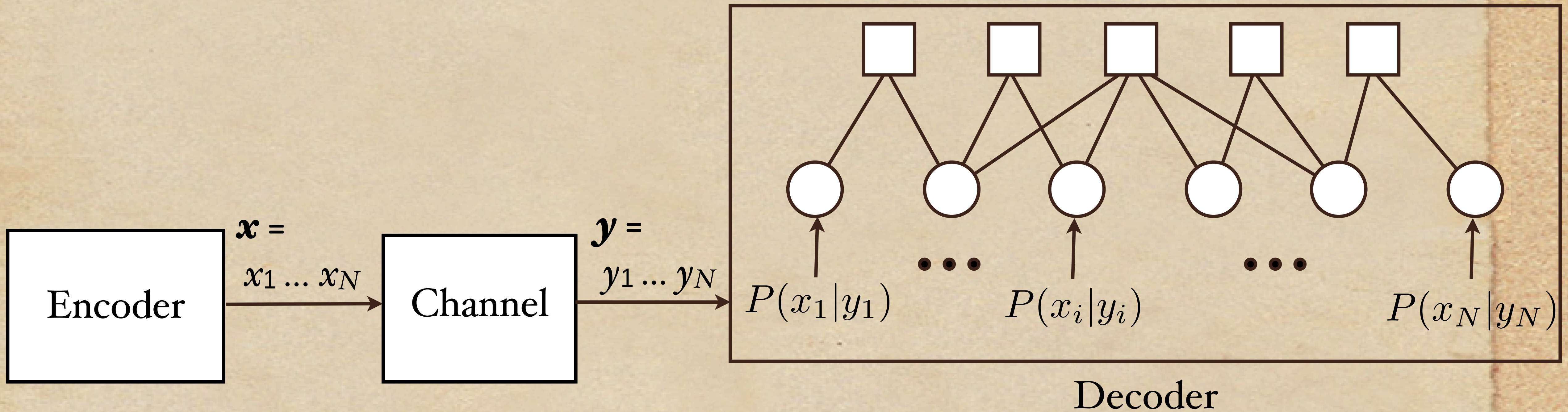
Instead of bit-flipping, the algorithm passes "probability messages" or "soft information"

Not just BSC, but message-passing decoding works for a variety of channels, for example AWGN, binary erasure channel.

Probabilistic Decoding

Probabilistic decoding is an instance of message passing

It is an effective way to decode LDPC Codes

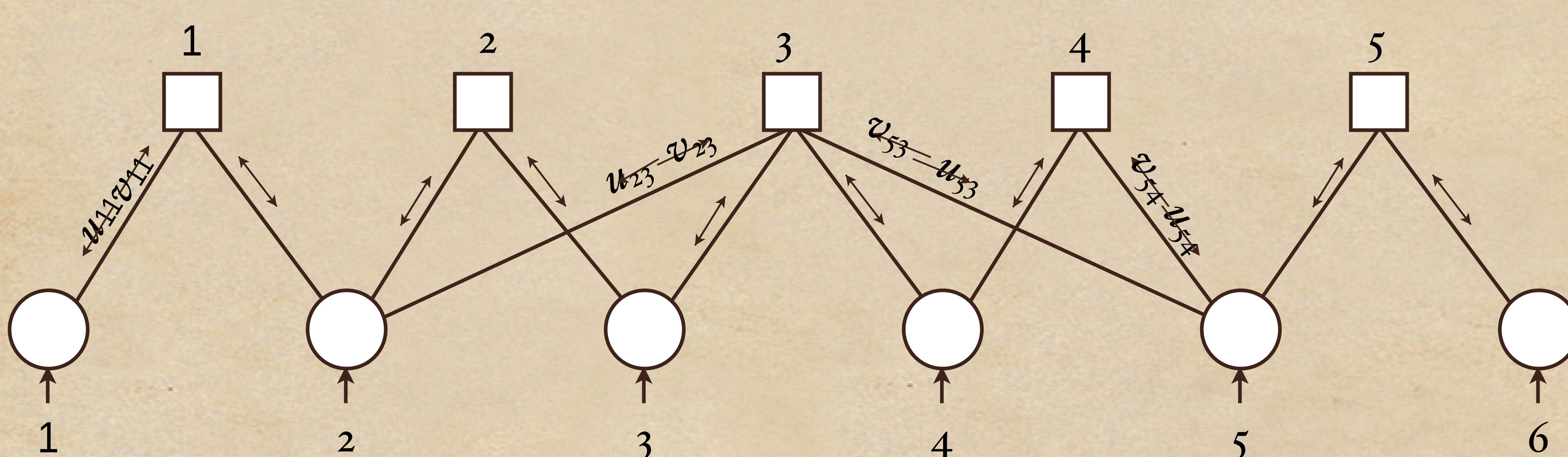


Message Passing

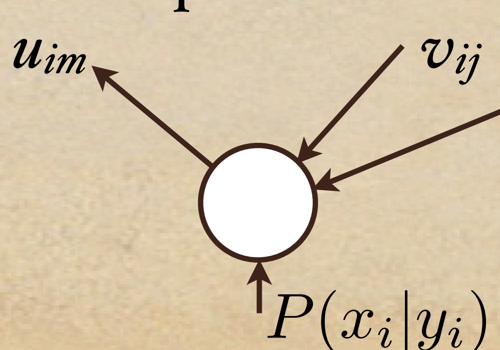
Messages are probabilities: P(x_i).

u_{ij} = P(x_i) is the message passed from the bit node *i* to check node *j*

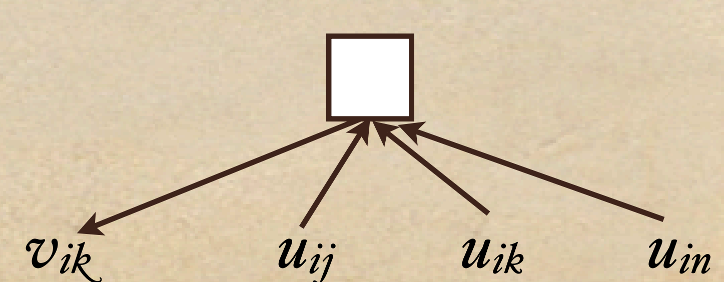
v_{ij} = P(x_i) is the passed from the check node *j* to bit node *i*.



The bit node computes its output *u*, from inputs *v* and P(x_i | y_i):



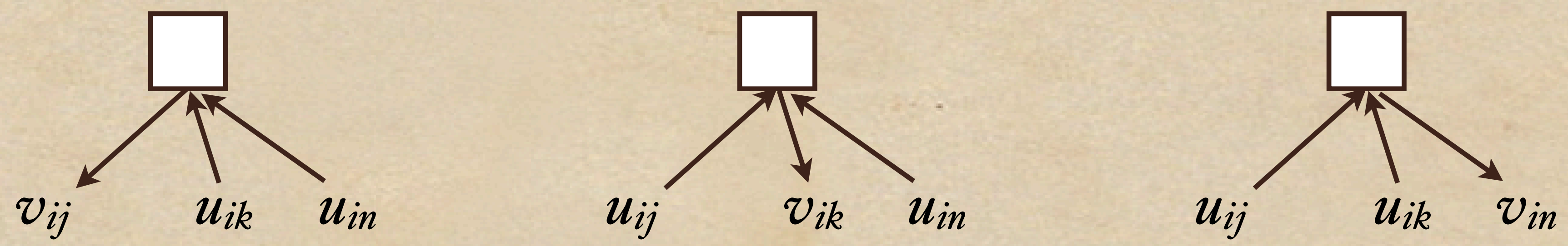
The check node computes its output *v*, from inputs *u*:



The Sum-Product Update Rule

The Sum-Product Update Rule [Kschischang, et al.]:

> The message sent from a node N on an edge e is the product of the local function at N with all messages received at N on edges other than e, summarized for the variable associated with e.



⇒ One message must be computed for

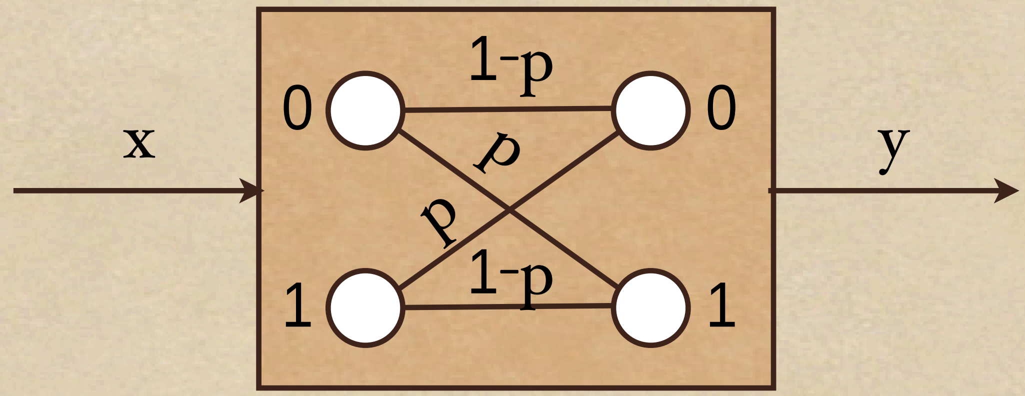
each edge, per check node

BSC Channel *A Posteriori* Probability: P(x_i | y_i)

Binary Symmetric Channel — Errors are independent

The *a priori* information about x_i are $P(x = 0) = P(x = 1) = 0.5$.

$P(y = 0) = P(y = 1) = 0.5$ because of the symmetry of the channel.



Channel: BSC(p)

An error occurs with probability p means:

$$P(y = 1 \mid x = 0) = p$$

$$P(y = 0 \mid x = 0) = 1\text{-}p$$

Using Bayes's Rule:

$$P(x=0|y=1) = P(y=1|x=0)\frac{P(x=0)}{P(y=1)}$$

$$P(x=0|y=1) = p\frac{0.5}{0.5} = p$$

$$P(x=0|y=0) = 1-p$$

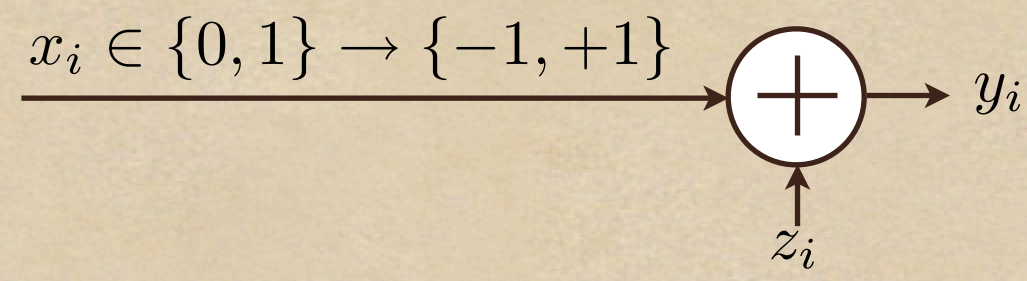
AWGN Channel *A Posteriori* Probability: P(x_i | y_i)

Additive White Gaussian Noise (AWGN) Channel

$$z_i \sim \mathcal{N}(0, \sigma^2)$$

- Assume the z_i are independent.

$$f_Z(z) = \frac{1}{\sqrt{2\pi}\sigma} e^{z^2/2\sigma^2}$$



$$P(x = 0|y) = P(y|x = 0)\frac{P(x=0)}{P(y)}$$

$$\boxed{\begin{aligned} P(x = 0|y) &= ke^{(y+1)^2/2\sigma^2} \\ P(x = 1|y) &= ke^{(y-1)^2/2\sigma^2} \end{aligned}}$$

$$k = \frac{1}{2\sqrt{2\pi}\sigma P(y)}$$

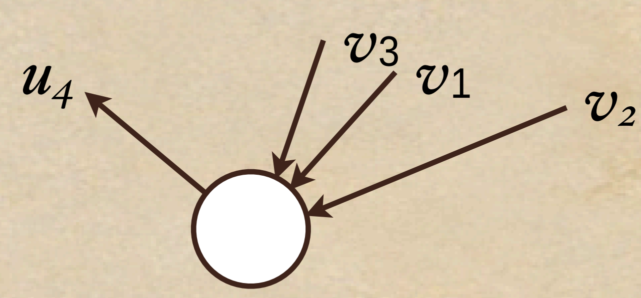
$$Find \quad k \quad by \quad P(x = 0|y) + P(x = 1|y) = 1$$

Bit Node Function

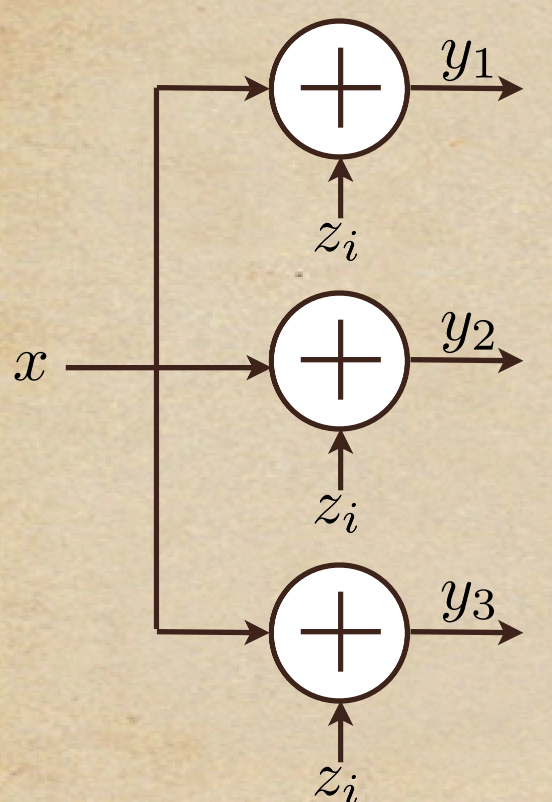
At the Bit Node, we have several different estimates about a bit x:

v_1 = P(x=0|y_1), v_2 = P(x=0|y_2), v_3 = P(x=0|y_3).

What is the combined estimate, u_4=P(x=0|$y_1 y_2 y_3$)?



Consider this system:



$$v_1 = P(x = 0|y_1)$$

$$v_2 = P(x = 0|y_2)$$

$$v_3 = P(x = 0|y_3)$$

Using the identity:

$$\frac{P(x|y_1 y_2 y_3)}{P(x)} = \frac{P(x|y_1)}{P(x)} \frac{P(x|y_2)}{P(x)} \frac{P(x|y_3)}{P(x)}$$

We can show:

$$P(x = 0|y_1, y_2, y_3) = \frac{\prod_i P(x = 0|y_i)}{\prod_i P(x = 0|y_i) + \prod_i P(x = 0|y_i)}$$

Or:

$$u_4 = \frac{v_1 v_2 v_3}{v_1 v_2 v_3 + (1 - v_1)(1 - v_2)(1 - v_3)}$$

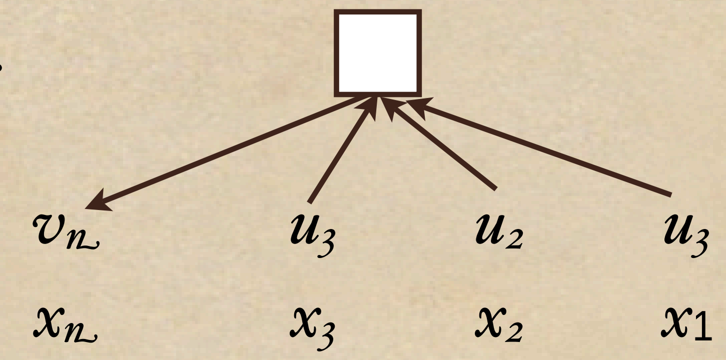
Check Node Function

At the Check Node, we know $x_1 + x_2 + \ldots + x_n = 0$.

What is $v_n = P(x_n=0)$? What is $P(x_n=1)$?

Let $\rho_n = P(x_n=0) - P(x_n=1)$.

Let $x(k) = \sum_{i=1}^{k} x_i$.



- Note that since $x_1 + x_2 + \ldots + x_n = 0$:

Even: $x_n = 0$ is the same as $x(n\text{-}1) = 0$

Odd: $x_n = 1$ is the same as $x(n\text{-}1) = 1$

$\Rightarrow P(x_n = 0) = P(x(n\text{-}1) = 0)$

- By Bayes' Rule:

$P(x_n = 0) = P(x(n\text{-}2) = 0, x_{n-1} = 0)$
$\quad + P(x(n\text{-}2) = 1, x_{n-1} = 1)$

- By independence:

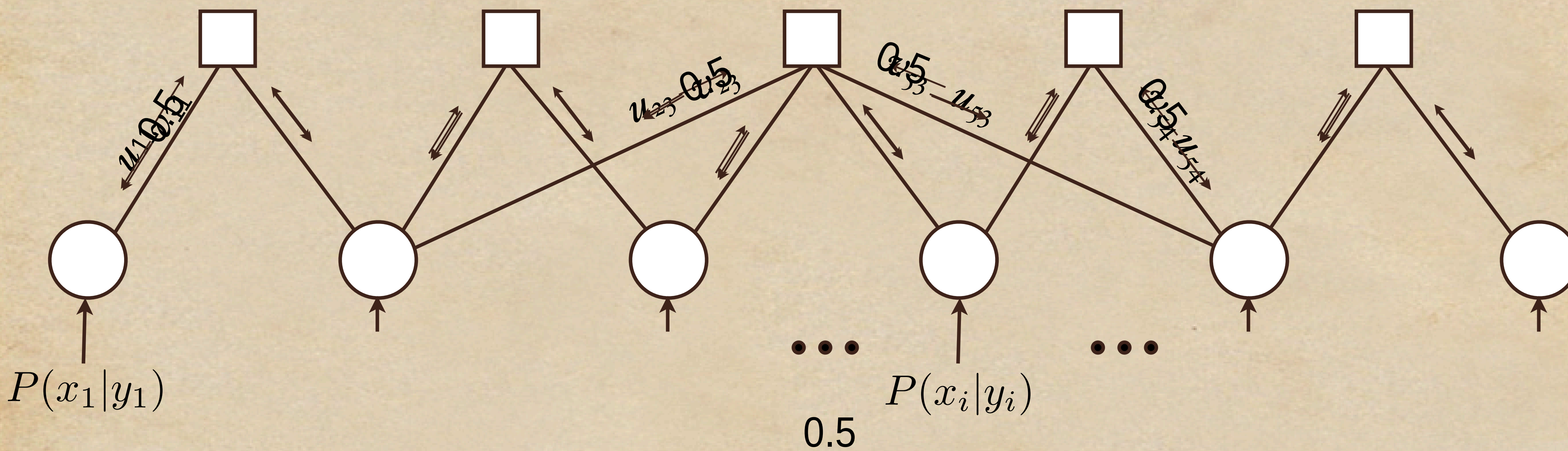
$P(x_n = 0) = P(x(n\text{-}2) = 0)\, P(x_{n-1} = 0)$
$\quad + P(x(n\text{-}2) = 1)\, P(x_{n-1} = 1)$

Can show:

$\rho_n = \rho_{n-1} \ldots \rho_1$

$$\boxed{v_n = \frac{1}{2}\Big((2u_1 - 1)\cdots(2u_{n-1} - 1) + 1\Big)}$$

Message Passing Decoding Algorithm



1. Initialize: v_{ij} messages to be $P(x_i) = 0.5$.
2. Compute the bit-to-check messages u_{ij} from v_{ij} (on the first iteration, we use $P(x_i \mid y_i)$).
3. Compute the check-to-bit messasges, v_{ij} from u_{ij}.
4. At each node, compute the temporary estimate $\hat{x}_i$. If $\hat{\mathbf{x}} H^t = 0$, then stop decoding, $\hat{x}$ is a valid codeword.
5. Otherwise repeat until Steps 2-4 until a maximum number of iterations has been reached.