Soft iterative decoding of block codes

- Soft
  - Probabilistic decoding
- Iterative
  - Message-passing on a graph
  - Distributed computation
- Block codes
  - Repetition code
  - Parity-check code
  - Hamming code
  - Low-density parity-check codes

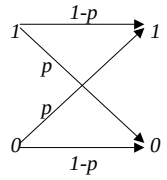
John L. Fan
Iospan Wireless, Inc.

john@johnfan.com

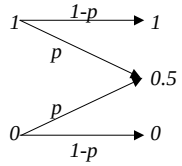
Simple parity-check code

$$x_1 \oplus x_2 \oplus \cdots \oplus x_n = 0$$

- Rate = (n-1)/n
- Binary symmetric channel
  - Detects 1 error
  - Corrects 0 errors



- Binary erasure channel
  - Detects n erasures
  - Corrects 1 erasure

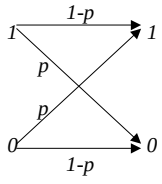


- Code
  - Any word with even parity
  - 2^{n-1} words in total

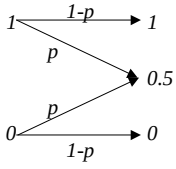
Simple repetition code

$$x_1 = x_2 = \cdots = x_n$$

- Rate = 1/n
- Binary symmetric channel
  - Detects n-1 errors
  - Corrects $\left\lceil \frac{n-1}{2} \right\rceil$ errors

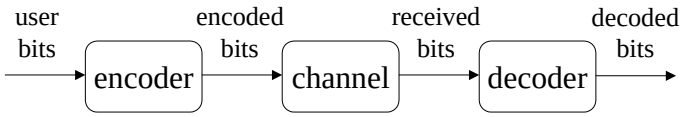


- Binary erasure channel
  - Detects n erasures
  - Corrects 1 erasure

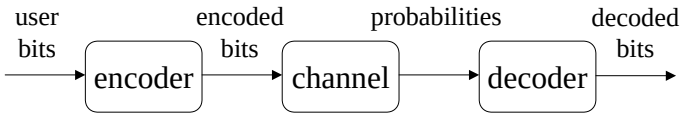


- Code
  - Two codewords: (0,0,…,0) and (1,1,…,1)

Channel



- Binary symmetric channel



- Binary erasure channel
- *Binary-input AWGN channel* (with appropriate soft channel decoder)

Probabilities

- Key point: probabilities are often available from the channel (or channel decoder)
- Probability p, from 0 to 1
- Log-likelihood ratio (LLR)

$$LLR(p) = \log \frac{p}{1-p}$$

- "Soft" decoding (using probabilities) vs. "hard" decoding (using bits)
- Questions
  - how to perform soft decoding?
  - what is the benefit of soft decoding?

Probability results

- Joint probability $P(A,B)$
- Independence $P(A,B) = P(A)P(B)$
- Total probability
  - if events $\{A_i\}$ form a partition of sample space

$$\sum_i P(A_i) = 1 \qquad \sum_i P(A_i, B) = P(B)$$

- Conditional probability $P(A \mid B) = \frac{P(A,B)}{P(B)}$
- Bayes' theorem $P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$
- Chain rule

$$P(A, C \mid B) = P(A \mid B, C)P(B \mid C)$$

- Markov chain (special case of a chain)

$$P(A, C \mid B) = P(A \mid C)P(B \mid C)$$

Types of probability

- **intrinsic** probability : *a priori*, prior
  - this is the original probability, *a=P(x)*
- **posterior** probability : *a posteriori*
  - this is the conditional probability *p=P(x/N)* based on some event N
- **extrinsic** probability
  - this represents what has been learned from the event N
  - given by the difference of the posterior and intrinsic probabilities
- note that these probabilities should be expressed "with respect to" the event N

$$\frac{ae}{(1-a)(1-e)} = \frac{p}{(1-p)}$$

$$\log \frac{a}{1-a} + \log \frac{e}{1-e} = \log \frac{p}{1-p}$$

$$LLR(a) + LLR(e) = LLR(p)$$

Repetition code - soft decoding

- Consider a three-bit repetition code

$$x_1 = x_2 = x_3$$

- Codewords are (0,0,0) and (1,1,1)
- The ECC receives probabilities:
  - intrinsic probabilities
  - assumed to be independent

$$a_i = P(x_i = 1)$$

$$(a_1, a_2, a_3)$$

- Making a hard-decision
  - (1, 1, 1) => decoded as (1,1,1)
  - (0.5, 0.5, 1) => decoded as ?
  - (0.3, 0.5, 0.8) => decoded as ?
  - (0.3, 0.4, 0.8) => decoded as ?
  - (0.1, 0.1, 1) => decoded as ?

Using Bayes' theorem

- What is the posterior probability?

Let N represent the event that the check is satisfied. By Bayes' theorem,

$$P(x_1 = 1 \mid N) = \frac{P(N \mid x_1 = 1)P(x_1 = 1)}{P(N)}$$

Meanwhile,

$$P(N \mid x_1) = \sum_{x_2, x_3} P(N, x_2, x_3 \mid x_1)$$

$$= \sum_{x_2, x_3} P(N \mid x_1, x_2, x_3) P(x_2) P(x_3)$$

Then the posterior probability for the repetition code is:

$$p_1 = P(x_1 = 1 \mid N) = \frac{a_1 a_2 a_3}{P(N)}$$

where $P(N) = a_1 a_2 a_3 + (1 - a_1)(1 - a_2)(1 - a_3)$

Probabilities for repetition code

$$p_1 = \frac{a_1 a_2 a_3}{a_1 a_2 a_3 + (1 - a_1)(1 - a_2)(1 - a_3)}$$

- Computing posterior (p_i) from intrinsic (a_i)
  - Intrinsic => posterior
  - (0.5, 0.5, 1) => (1,1,1)
  - (0.3, 0.5, 0.8) => 0.6316*(1,1,1)
  - (0.3, 0.4, 0.8) => 0.5333*(1,1,1)
  - (0.1, 0.1, 1) => (1,1,1)
  - (Note for the last two examples that a hard-decoder using the input (0,0,1) would make the incorrect decision.)
- In addition, the extrinsic (e_i) can be found as follows:

$$e_1 = \frac{a_2 a_3}{a_2 a_3 + (1 - a_2)(1 - a_3)}$$

Parity-check code - soft decoding

- Consider a three-bit repetition code

$$x_1 \oplus x_2 \oplus x_3 = 0$$

- Codewords are (0,0,0), (1,0,1), (0,1,1) and (1,1,0)
- Applying Bayes' theorem and cranking through the math gives:

$$e_1 = a_2(1 - a_3) + (1 - a_2)a_3$$

$$p_1 = \frac{a_1 e_1}{a_1 e_1 + (1 - a_1)(1 - e_1)}$$

- Example:

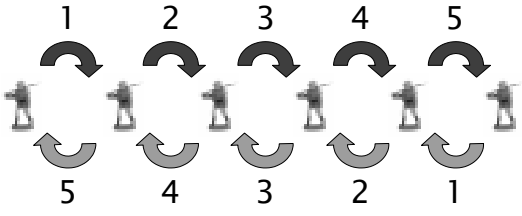
  intrinsic $(a_1, a_2, a_3) = (0.2, 0.5, 0.9)$

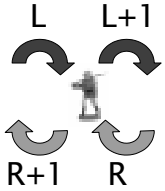
  extrinsic $(e_1, e_2, e_3) = (0.5, 0.74, 0.5)$

  posterior $(p_1, p_2, p_3) = (0.2, 0.74, 0.9)$

Counting soldiers

- Message-passing algorithm
- An illustration of iterative decoding
- Goal: to count the total number of soldiers, and pass this information to everyone
- Restriction: Each soldier can only communicate with neighbors

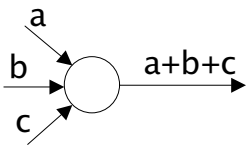




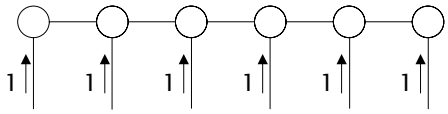
Total: L + R + 1

Graph for soldier-counting

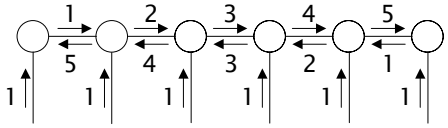
Message-passing rule



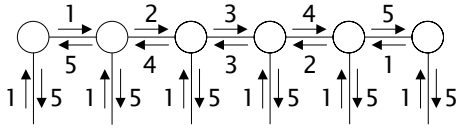
Prior



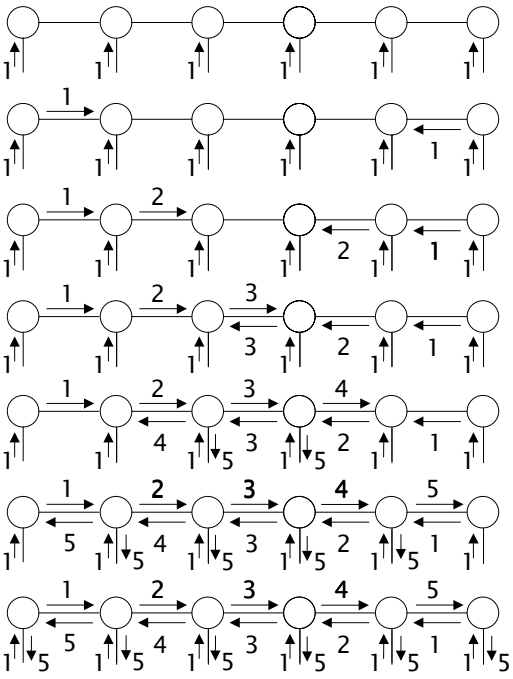
Forward-backward



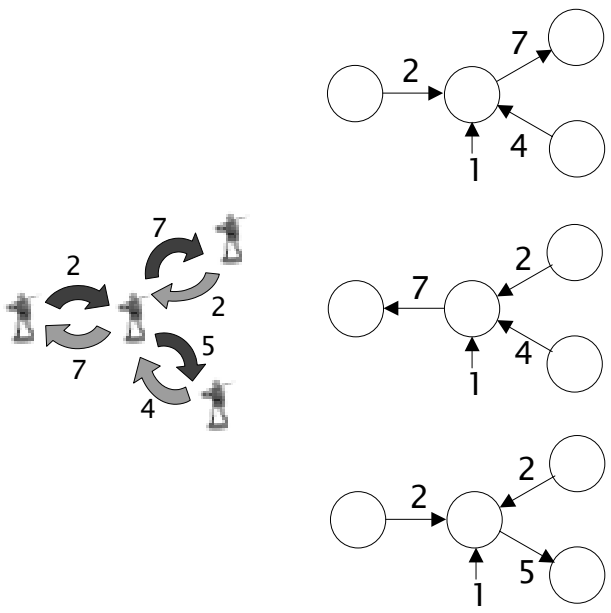
Extrinsic



Messages in soldier counting

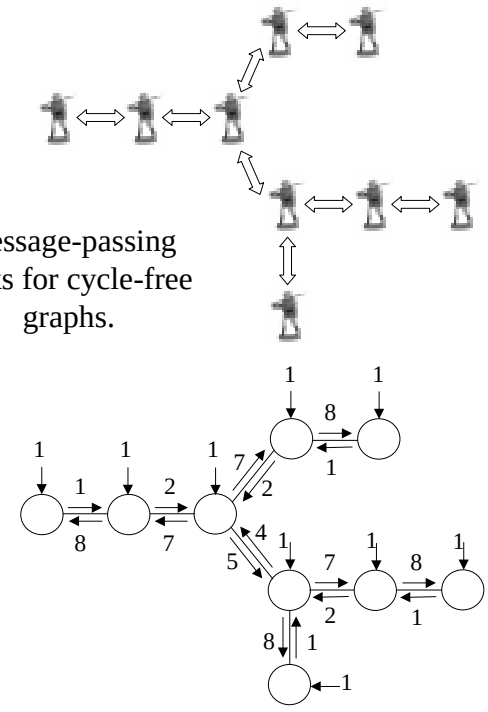


A soldier with 3 neighbors

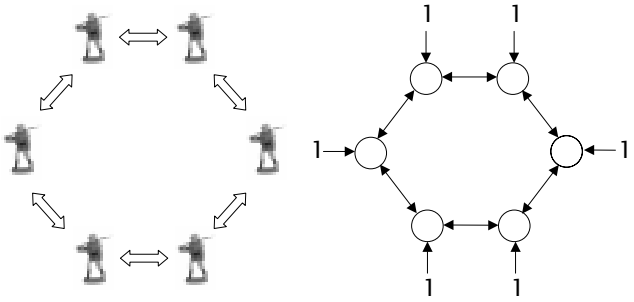


Tree formation

Message-passing works for cycle-free graphs.



Circle formation

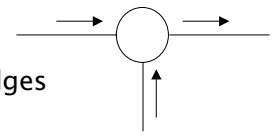


Message-passing
can fail when the graph contains cycles!

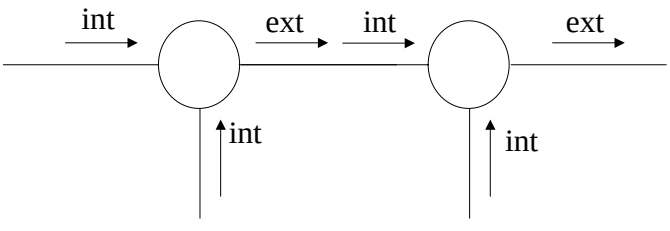
Distributed computation

- Soldier-counting is one type of *message-passing algorithm*
  - computations at the nodes
  - messages passed in both directions on the edges
  - the output along one edge of a node is a function of the inputs along all the *other* edges of the node
- Distributed computation
  - Parallel computation
  - Divide and conquer
  - Use simpler decoders to obtain the performance of a more complicated one
- Our target application for message-passing: the soft decoding of block codes

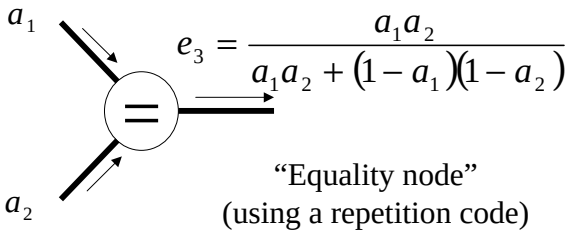
Graphs



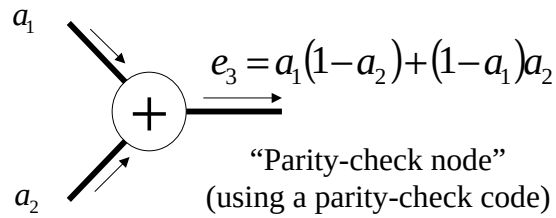
- Composed of nodes and edges
- Functions (e.g., codes or constraints) associated with nodes
- Variables are associated with the edges
- Messages passed in both directions on edges
- Computation performed at a node N:
  - Input: the **intrinsic** probability w.r.t. node N
  - Output: the **extrinsic** probability w.r.t. node N



Equality and parity-check nodes



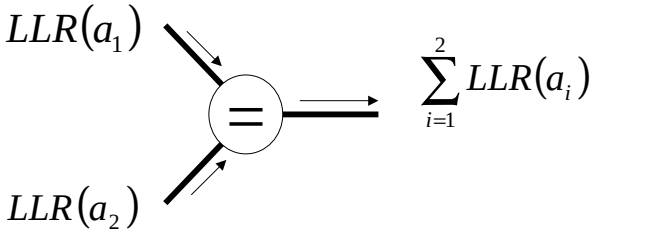
"Equality node"
(using a repetition code)

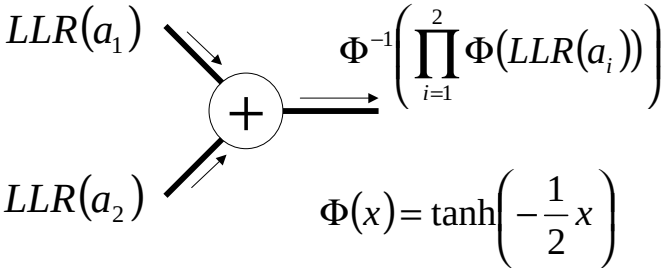


"Parity-check node"
(using a parity-check code)

Log-likelihood ratio

- For binary variables, it may be useful for implementation to use a log-likelihood ratio:





Hyperbolic tangent

- Log-likelihood ratio

$$LLR(p)=\log\frac{p}{1-p}$$

- Hyperbolic tangent

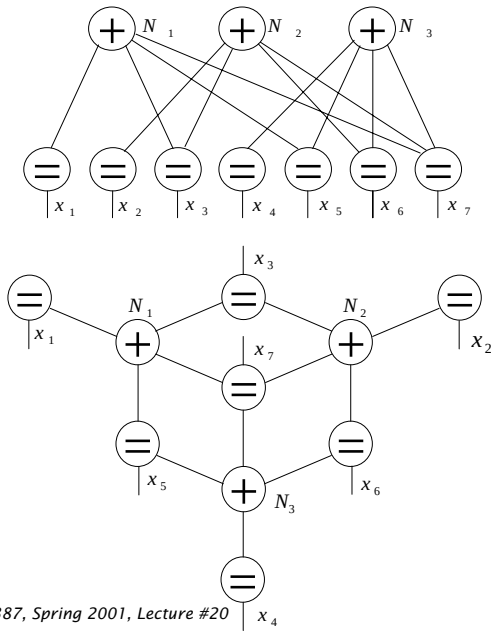
$$\tanh(x)=\frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$$

- "Soft bit"

$$2p-1=\tanh\left(\frac{1}{2}LLR(p)\right)$$

Hamming code

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



Hamming code: performance

- Rate = 4/7
- Binary symmetric channel
  - Detects 2 errors
  - Corrects 1 errors
- Binary erasure channel
  - Detects 7 erasures
  - Corrects 3 erasures
- Binary-input AWGN channel: