When we have more than 3 variable nodes connecting to the check node a, it is easy to show using induction that

$$\tanh\left(\frac{L_{ai}}{2}\right) \leftarrow \prod_{j\in N(a)\setminus i} \tanh\left(\frac{L_{ja}}{2}\right). \tag{1.73}$$

Hard threholding can be applied to the final belief to estimate the value of each bit. That is,

$$\hat{x}_i = \begin{cases} 0, & \text{if } \beta_i \geq 0, \\ 1, & \text{otherwise,} \end{cases} \tag{1.74}$$

where $\beta_i = \sum_{a\in N(i)} L_{ai}$. The algorithms may terminate after some predefined number of iterations. It is also common to terminate the algorithm whenever the estimate variables satisfy all checks. The overall algorithm is summarized in Algorithm 1.3.

Algorithm 1.3 Summary of LDPC decoding using BP

- **Initialization**:

$$L_{ii} = \log \frac{p(0|y_i)}{p(1|y_i)} \tag{1.75}$$

- **Messages update**:
  - Check factor to variable node update:

$$L_{ai} \leftarrow 2\tanh^{-1}\left(\prod_{j\in N(a)\setminus i} \tanh\left(\frac{L_{ja}}{2}\right)\right) \tag{1.76}$$

  - Variable to check factor node update:

$$L_{ia} \leftarrow \sum_{b\in N(i)\setminus a} L_{ia}, \tag{1.77}$$

- **Belief update**: denote the net belief in terms of log-likelihood ratio as β_i for variable node i, then

$$\beta_i \leftarrow \sum_{a\in N(i)} L_{ai} \tag{1.78}$$

- **Stopping criteria**: repeat message update and belief update steps until maximum number of iterations is reached or all checks are satisfied. Output the hard threshold values of net beliefs as the estimated coded bits.

1.4.4 Soft Decoding of Convolutional Codes using BP

While the Viterbi Algorithm as described in Section 1.2.1 can find the maximum likelihood sequence, it cannot tell how likely that a particular bit of the input is

1 or 0. That is why Viterbi algorithm is sometimes known to be a *hard-decoding* algorithm. In this section, we will describe a *soft-decoding* algorithm that can output the probability of each input bit to be 1 or 0 based on BP algorithm.

The key is to realize the correlation structure of the trellis can be represented by a hidden Markov model. Given the two states before and after an input bit is coded, the output bits are conditional independent of the states and the output bits at all other times. Thus, the correlations among the states and true output bits at all times can be depicted by a factor graph as shown in Figure 1.8. Note that the factor graph is actually a tree in this case. And thus the probabilities estimated by belief propagation is actually exact.

Let us revisit the $[5, 7]$ rate-1/2 convolution code. Assume that the output bits go through a binary symmetric channel with cross over probability p. Denote y as the actual output at a particular time instance and s_1 and s_2 are states before and after y is processed. The factor function $f(s_1, s_2, y)$ for the factor node connecting s_1, s_2, and y can be easily constructed and is listed in Table 1.1. For example, for $y = 00$, $s_1 = 00$, and $s_2 = 01$, $f(s_1, s_2, y) = 0$ since transition from state 00 to state 01 is not allowed (see Figure 1.2). On the other hand, if $s_2 = 10$, $f(s_1, s_2, y) = p^2$ is non-zero now as the transition from s_1 to s_2 is valid. Moreover, given 11 as the anticipated output for this state transition, both received bits must be wrong and hence the probability is p^2.

Belief propagation can be applied directly to the factor graph to compute the probability distribution of each state. When these probabilities are known, the probability of the input bit to be x will be the sum probability of the transitions that results from the input x.

Example 1.4 (Soft-decoding using Belief Propagation)**.** Let us continue with Example 1.1. Assuming that the fourth and the eighth bits are corrupted. That is, the decoder receives $00, 10, 10, 00, 10, 11$. Given these bits, the decoder can estimate the probabilities of the received states using belief propagation as in Figure 1.8. Figure 1.8(a) shows the message received by each state variable node from the factor node to its right. Note that the first state is known to be 00 by construction and thus $Pr(s = 00) = 1$. This probability is passed to the first factor node from the left and the message to second state variable node is computed and passed. The messages are computed one after another with factor nodes on the left started first. The probability distribution of each state is shown after all factor nodes are updated. We call this the forward propagation.

Similarly, since the last state has to be 00 by designed also (the encoder is flushed with extra 0 bits after encoding the entire input bitstream), $Pr(s = 00) = 1$ holds for the last state. This probability distribution is passed to the neighboring factor node to its left. The factor nodes will be updated one by one with the nodes on the right finish first. The probability distribution of each state after the updates are shown In Figure 1.8(b). We call this the backward propagation.

In Figure 1.8(c), the combined probabilities of both forward and backward updates are shown. With the known state probabilities, the decoder can compute the probability of each input bit to be 0 (or 1) as the sum probability of the transition that occurs when 0 (or 1) received. For example, there are only two transitions ($00 \rightarrow 00$ and $00 \rightarrow 10$) with non-zero probability (0.988 and 0.012). The first transition occurs when a 0 is received and the second occurs when a 1 is received. Therefore, $Pr(X_1 = 0) = 0.988$

(a) Received output $y = 00$

		s_1			
		00	01	10	11
s_2	00	$(1-p)^2$	p^2	0	0
	01	0	0	$p(1-p)$	$p(1-p)$
	10	p^2	$(1-p)^2$	0	0
	11	0	0	$p(1-p)$	$p(1-p)$

(b) Received output $y = 01$

		s_1			
		00	01	10	11
s_2	00	$p(1-p)$	$p(1-p)$	0	0
	01	0	0	$(1-p)^2$	p^2
	10	$p(1-p)$	$p(1-p)$	0	0
	11	0	0	p^2	$(1-p)^2$

(c) Received output $y = 10$

		s_1			
		00	01	10	11
s_2	00	$p(1-p)$	$p(1-p)$	0	0
	01	0	0	p^2	$(1-p)^2$
	10	$p(1-p)$	$p(1-p)$	0	0
	11	0	0	$(1-p)^2$	p^2

(d) Received output $y = 11$

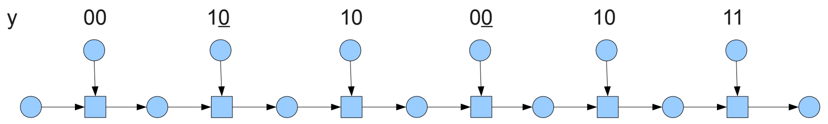
		s_1			
		00	01	10	11
s_2	00	p^2	$(1-p)^2$	0	0
	01	0	0	$p(1-p)$	$p(1-p)$
	10	$(1-p)^2$	p^2	0	0
	11	0	0	$p(1-p)$	$p(1-p)$

Table 1.1: $f(s_1, s_2, y)$ for different received output y.

and $Pr(X_1 = 1) = 0.012$. For the second input bit, a 0 occurs for transitions $00 \to 00$, $01 \to 00$, $10 \to 01$ and $11 \to 01$. The sum probability for these transitions are $0.988 \times 0.1 + 0 \times 0.1 + 0.012 \times 0 + 0 \times 0 = 0.099$. Similarly, the probability of input bit to be 1 computed by sum probability is $0.988 \times 0.888 + 0 \times 0.888 + 0.012 \times 0.012 + 0 \times 0.012 = 0.878$. Note that the two probabilies do not add up to zero and need normalization since any two neighboring states are not really independent. After normalization, we have $Pr(x = 0) = 0.1011$ and $Pr(x = 1) = 0.8989$. Similarly, we can compute the probabilities of all other input bits. The probabilities for them to be 0 are 0.988, 0.101, 0.014, 0.001, 1.000, and 1.000. Thus the decoded message will be $[0, 1, 1, 1, 0, 0]$, which is the same as the true input.

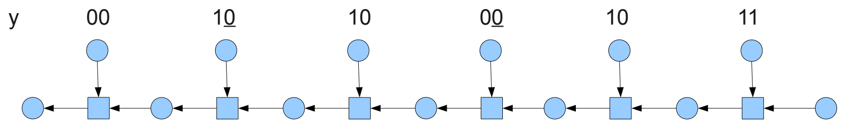
1.4.5 IRA Codes

LDPC codes are generally not systematic codes (see Section 1.1). Therefore, we need an extra step to further translate the corrected codeword at the decoder



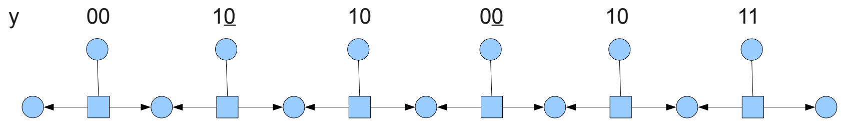
Pr(s=00)	1	0.989	0.473	0.083	0.233	0.086	0.670
Pr(s=01)	0	0	0.001	0.091	0.255	0.412	0.091
Pr(s=10)	0	0.122	0.473	0.083	0.258	0.086	0.149
Pr(s=11)	0	0	0.053	0.743	0.255	0.416	0.091

(a) State probabilities from forward propagation



Pr(s=00)	0.412	0.254	0.083	0.006	0.001	0.012	1
Pr(s=01)	0.415	0.254	0.083	0.052	0.001	0.988	0
Pr(s=10)	0.087	0.257	0.742	0.471	0.012	0	0
Pr(s=11)	0.087	0.235	0.091	0.471	0.985	0	0

(b) State probabilities from backward propagation



Pr(s=00)	1	0.988	0.100	0.001	0.001	0.003	1
Pr(s=01)	0	0	0.000	0.012	0.001	0.997	0
Pr(s=10)	0	0.012	0.888	0.099	0.012	0	0
Pr(s=11)	0	0	0.012	0.888	0.985	0	0

(c) Combined state probabilities

Figure 1.8: Computed probabilites of the forward-backward soft decoding (belief propagation) algorithm. The underline received bits are corrupted.