Chapter 5 General Belief Propagation Decoding Algorithms for Low-Density Parity-Check Codes

5.1 Message-Passing Decoder: Belief Propagation Algorithm by Probabilities and Likelihood Ratios

Having investigated the essential knowledge of the graph theory, Bayesian networks and Tanner graphs, we now introduce a generic and important decoding algorithm, i.e., Belief Propagation (BP) algorithm, which is applicable for decoding of a wide variety of error-correcting codes based on the sparse Tanner graph of long length, including the tree codes, CT codes and LDPC codes, etc. This sort of codes family is simply called as Tanner graph codes [26]. Actually, BP algorithm is not a completely new concept, which was adopted by Gallager in decoding of his celebrated regular LDPC codes in this early thesis [1] about four decades ago. Recent intensive studies [5][26][41] have unveiled that Gallager's iterative approach is only an approximate BP algorithm. The turbo decoding algorithm may also be viewed as an instance of the BP algorithm [22].

In order to perfectly understand the decoding procedures by BP algorithm, we need the following lemma:

Lemma 5.1. Consider a sequence of L independent binary digits in which the l-th digit is 1 with probability $P_l^{(1)}$, and thus 0 with probability $P_l^{(0)}=1-P_l^{(1)}$. Then, the probability that an even number of digits are 1 is

$$Pr(\text{An even number of 1's})=\frac{1+\prod_{l=1}^{L}(1-2P_l^{(1)})}{2}=\frac{1+\prod_{l=1}^{L}(P_l^{(0)}-P_l^{(1)})}{2} \tag{5-1}$$

The proof of the Lemma can be referred to Appendix A [1, Chapter 4, Lemma 4.1]. Hence, the probability that an odd number of digits are 1 is

$$Pr(\text{An odd number of 1's}) = 1 - Pr(\text{An even number of 1's})$$

$$= \frac{1 - \prod_{l=1}^{L} (P_l^{(0)} - P_l^{(1)})}{2} \tag{5-2}$$

Now we investigate the BP algorithm for decoding of LDPC codes. The codeword bits are transmitted using BPSK modulation scheme over an AWGN channel, the received channel output r_n at n-th epoch becomes

$$r_n = s_n + w_n \tag{5-3}$$

where s_n is the n-th transmitted symbol taking value $+a$ or $-a$ corresponding to the codeword bit d_n as 0 and 1 with equal probability, and w_n is an independent Gaussian variable with zero mean and variance σ^2, respectively.

Consider a Tanner graph, denoted by $\{v_n : n = 1, 2, \ldots, N\}$ and $\{c_m : m = 1, 2, \ldots, m_{\max}\}$ the sets of the variable and check nodes, where N and $m_{\max}$ represent the maximum numbers of variable and check nodes of the Tanner graph, respectively. For a regular LDPC code of length N with d_v and d_c, i.e., A(N, d_v, d_c), $m_{\max}$ is equal to $N d_v / d_c$ determined by the definition 4.1. Let $C(x)$ be the set of check nodes linked by a variable node $x \in \{v_n : n = 1, 2, \ldots, N\}$. Similarly, let $V(y)$ be the set of variable nodes linked by a check node $y \in \{c_m : m = 1, 2, \ldots, m_{\max}\}$. The ultimate goal of the decoder is to evaluate the *a posterior* likelihood ratio for each codeword bit d_n ($n = 1, 2, \ldots, N$).

$$R_n = \frac{\Pr(d_n = 0 \left| \{r\right| \}, S(C(v)) = 1)}{\Pr(d_n = 1 \left| \{r\right| \}, S(C(v)) = 1)} \tag{5-4}$$

Note that the event, denoted by the received sequence $\{r_l \big|_{l=1}^{N}\}$, means that all the channel soft outputs are obtained by the decoder. Another event, denoted by a binary function $S(C(v_n))$, is assigned by 1 if all the check nodes in the set $C(v_n)$ are satisfied simultaneously by the transmitted codeword bits; otherwise, assigned by 0. The decoder decides the estimated output by the following rule

$$\hat{d}_n = \begin{cases} 0 & R_n > 1 \\ 1 & R_n \le 1 \end{cases} \tag{5-5}$$

The task of BP algorithm is to iteratively compute the *a posterior* probability likelihood ratio as indicated by (5-4). The whole procedure can be divided into the following steps:

The Belief Propagation Algorithm for LDPC Codes

Preparations: Let us define the two conditional probabilities as:

$$f_n^{(0)} = Pr(d_n = 0 \left| \{r_l \big|_{l=1}^{N}\}\right.) \tag{5-6a}$$

$$f_n^{(1)} = Pr(d_n = 1 \left| \{r_l \big|_{l=1}^{N}\}\right.) = 1 - f_n^{(0)} \tag{5-6b}$$

The likelihood ration of the *a posterior* probabilities is

$$f_n = \frac{f_n^{(0)}}{f_n^{(1)}} \tag{5-6c}$$

Initially, the decoder only receives the channel outputs (associated with the variable nodes of the graph) and has no any prior information from the parity checks (related to

the check nodes). Hence, the codeword bit d_n is only relevant with the individual channel output r_n, and (5-6) can be rewritten as follows over an AWGN channel.

$$f_n^{(0)} = Pr(d_n = 0 | r_n) = \frac{\Pr(r_n | d_n = 0)}{\Pr(r_n | d_n = 0) + \Pr(r_n | d_n = 1)} = \frac{1}{1 + \exp(-2ar_n / \sigma^2)} \tag{5-7a}$$

$$f_n^{(1)} = Pr(d_n = 1 | r_n) = 1 - f_n^{(0)} = \frac{1}{1 + \exp(2ar_n / \sigma^2)} \tag{5-7b}$$

$$f_n = \frac{f_n^{(0)}}{f_n^{(1)}} = \exp(2ar_n / \sigma^2) \tag{5-7c}$$

Based on (5-6) and Tanner graph, we further define two *a posterior* probabilities with respect to d_n, given the channel soft outputs $\{r_l|_{l=1}^N\}$ and all the checks related to v_n other than c_m, i.e., $C(v_n) \setminus c_m$,

$$q_{m,n}^{(0)} = Pr\left(d_n = 0 \left| \{r_l|_{l=1}^N\}, S\left(C(v_n) \setminus c_m\right) = 1\right.\right) \tag{5-8a}$$

$$q_{m,n}^{(1)} = Pr\left(d_n = 1 \left| \{r_l|_{l=1}^N\}, S\left(C(v_n) \setminus c_m\right) = 1\right.\right) = 1 - q_{m,n}^{(0)} \tag{5-8b}$$

Step 1. **Initialization**: Before commencing the iterative process, $q_{m,n}^{(0)}$ and $q_{m,n}^{(1)}$ can be initialized as $f_n^{(0)}$ and $f_n^{(1)}$ as indicated by (5-7a) and (5-7b), respectively [5][26].

Step 2. **Message from a check node c_m (Horizontal step)**: Let variable Δ_m, concerning with a check node c_m in the Tanner graph, be defined as

$$\Delta_m = \prod_{v_{n'} \in V(c_m)} (q_{m,n'}^{(0)} - q_{m,n'}^{(1)}) \tag{5-9}$$

If the value of d_n is assumed by the decoder, saying d_n=0, and the decoder has obtained all the channel outputs $\{r_l|_{l=1}^N\}$, an event that the check node $c_m \in C(v_n)$ is satisfied by the

transmitted codeword bits conditioned on d_n=0 is equivalent to that of an even number of parity bits with d_n excluded being 1 in the *m*-th parity check. Based on the Lemma 5.1, the conditional probability of this event is

$$Pr(S(\{c_m\})=1 \mid d_n=0,\ \{r_l\big|_{l=1}^{N}\})=\frac{1+\prod_{v_{\bar{n}}\in V(c_m)\backslash v_n}(q_{m,\bar{n}}^{(0)}-q_{m,\bar{n}}^{(1)})}{2}$$

$$=\frac{1+\Delta_m/(q_{m,n}^{(0)}-q_{m,n}^{(1)})}{2} \tag{5-10a}$$

Similarly, we have

$$Pr(S(\{c_m\})=1 \mid d_n=1,\ \{r_l\big|_{l=1}^{N}\})=\frac{1-\prod_{v_{\bar{n}}\in V(c_m)\backslash v_n}(q_{m,\bar{n}}^{(0)}-q_{m,\bar{n}}^{(1)})}{2}$$

$$=\frac{1-\Delta_m/(q_{m,n}^{(0)}-q_{m,n}^{(1)})}{2} \tag{5-10b}$$

The ratio of the two probabilities in (5-10) is

$$r_{m,n}=\frac{\Pr(S(\{c_m\})=1\big|d_n=0,\{r_l\,|_{l=1}^{N}\})}{\Pr(S(\{c_m\})=1\big|d_n=1,\{r_l\,|_{l=1}^{N}\})}=\frac{1+\Delta_m/(q_{m,n}^{(0)}-q_{m,n}^{(1)})}{1-\Delta_m/(q_{m,n}^{(0)}-q_{m,n}^{(1)})} \tag{5-11}$$

The message, regarded as extrinsic information, is sent to a variable node v_n (n=1, 2, …, N). The illustrative flow diagram associated with *Step* 2 is shown in Fig. 5.1, where a check node c_m is linked by four variable nodes v_n, v_{n_1}, v_{n_2} and v_{n_3}, respectively. The solid directed lines indicate the messages sent from the check node to the variable nodes in the *current* round, while the dashed directed lines indicate the messages sent from the variable nodes to the check node in the *last* round. For the variable node v_n, it receives the message $r_{m,n}$ from the check node c_m, which only depends on three groups of

messages ($q_{m,n_1}^{(0)}$, $q_{m,n_1}^{(1)}$), ($q_{m,n_2}^{(0)}$, $q_{m,n_2}^{(1)}$) and ($q_{m,n_3}^{(0)}$, $q_{m,n_3}^{(1)}$) provided by variable nodes v_{n_1}, v_{n_2} and v_{n_3} in the last round, respectively. In other words, $r_{m,n}$ is irrelevant with ($q_{m,n}^{(0)}$, $q_{m,n}^{(1)}$) supplied by v_n itself. The scenarios for v_{n_1}, v_{n_2} and v_{n_3} are the similar as that for v_n. This crucial strategy guarantees that the message sent by a check node is extrinsic information for the connected variable nodes if the graph is a *tree*. A check node sends message to the variable nodes, which participate in the parity check related to that check node, is virtually the process of exchange message between the variable nodes via the check node, and equivalent to column process for the parity check matrix. Hence, we also call *Step* 2 as *horizontal* step.

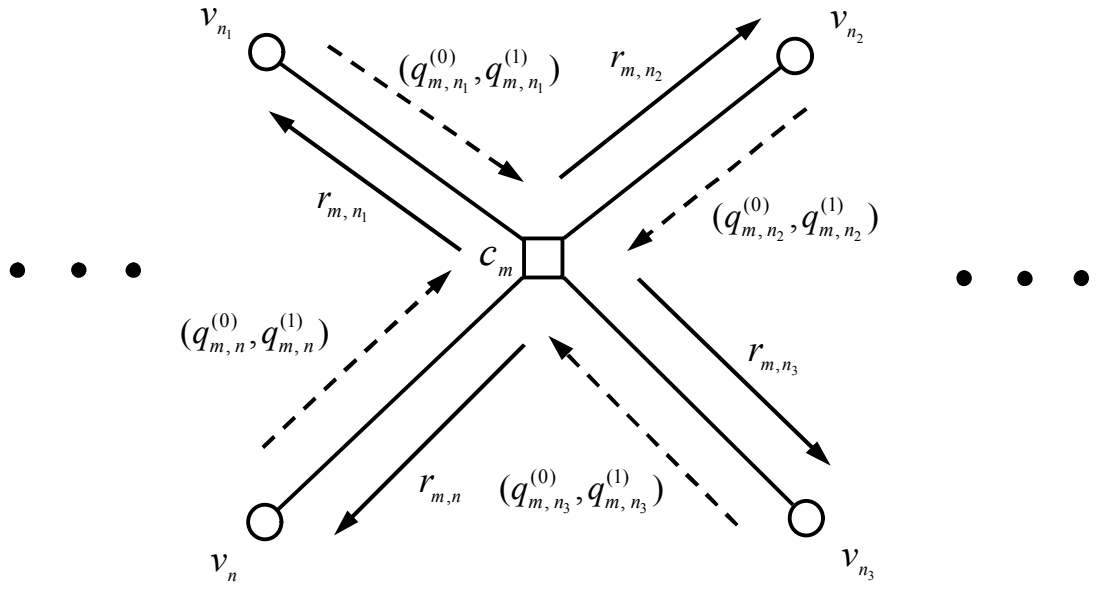

Fig. 5.1 The illustrative flow diagram of a local Tanner graph conveying the messages associated with *Step* 2 of the BP algorithm.

Step 3. **Message from a variable node (Vertical step)**:

Let $\overline{R}_n$, in relation to a variable node v_n of the Tanner graph, be defined as

$$\overline{R}_n = \prod_{c_{m'} \in C(v_n)} r_{m',n} \tag{5-12}$$

An event, which all the checks related to v_n except for c_m, i.e., $C(v_n)\backslash c_m$, are satisfied simultaneously by the transmitted codeword bits with the given value of d_n (0 or 1), is equivalent to that every such a check is satisfied individually. Based on (5-10) the probability of such an event given d_n=0 is

$$Pr\left(S\left(C(v_n)\backslash c_m\right)=1 \,\middle|\, d_n=0, \{r_l|_{l=1}^{N}\}\right) = \prod_{c_{\bar{m}} \in C(v_n)\backslash c_m} \Pr\left(S(\{c_{\bar{m}}\})=1 \,\middle|\, d_n=0, \{r_l|_{l=1}^{N}\}\right) \tag{5-13a}$$

Similarly, the probability of such an event on d_n=1 is

$$Pr\left(S\left(C(v_n)\backslash c_m\right)=1 \,\middle|\, d_n=1, \{r_l|_{l=1}^{N}\}\right) = \prod_{c_{\bar{m}} \in C(v_n)\backslash c_m} \Pr\left(S(\{c_{\bar{m}}\})=1 \,\middle|\, d_n=1, \{r_l|_{l=1}^{N}\}\right) \tag{5-13b}$$

The ratio of two probabilities in (5-13) is

$$\begin{aligned} \overline{\overline{R}}_{m,n} &= \frac{\Pr\left(S(C(v_n)\backslash c_m)=1 \,\middle|\, d_n=0, \{r_l|_{l=1}^{N}\}\right)}{\Pr\left(S(C(v_n)\backslash c_m)=1 \,\middle|\, d_n=1, \{r_l|_{l=1}^{N}\}\right)} = \frac{\prod_{c_{\bar{m}} \in C(v_n)\backslash c_m} \Pr\left(S(\{c_{\bar{m}}\})=1 \,\middle|\, d_n=0, \{r_l|_{l=1}^{N}\}\right)}{\prod_{c_{\bar{m}} \in C(v_n)\backslash c_m} \Pr\left(S(\{c_{\bar{m}}\})=1 \,\middle|\, d_n=1, \{r_l|_{l=1}^{N}\}\right)} \\ &= \prod_{c_{\bar{m}} \in C(v_n)\backslash c_m} r_{\bar{m},n} = \frac{\overline{R}_n}{r_{m,n}} \end{aligned} \tag{5-14}$$

Let $s_{m,n}$ be defined as

$$s_{m,n} = f_n\, \overline{\overline{R}}_{m,n} = f_n\, \frac{\overline{R}_n}{r_{m,n}} \tag{5-15a}$$

It is easy to verify that $s_{m,n}$ can be further expressed in terms of the updated $q_{m,n}^{(0)}$ and $q_{m,n}^{(1)}$ originally defined by (5-8), namely,

$$s_{m,n}=\frac{\Pr\left(d_n=0\left|\{r_l|_{l=1}^{N}\},S(C(v_n)\setminus c_m)=1\right.\right)}{\Pr\left(d_n=1\left|\{r_l|_{l=1}^{N}\},S(C(v_n)\setminus c_m)=1\right.\right)}=\frac{q_{m,n}^{(0)}}{q_{m,n}^{(1)}} \tag{5-15b}$$

Thus, the renewed $q_{m,n}^{(0)}$ and $q_{m,n}^{(1)}$ are

$$q_{m,n}^{(0)}=\frac{s_{m,n}}{1+s_{m,n}} \quad \text{and} \quad q_{m,n}^{(1)}=1-q_{m,n}^{(0)}=\frac{1}{1+s_{m,n}} \tag{5-16}$$

Note that two quantities above are irrelevant with the ratio $r_{m,n}$ provided by the check node c_m in the last round. The newly updated $q_{m,n}^{(0)}$ and $q_{m,n}^{(1)}$, also viewed as extrinsic information, are sent back to a check node c_m (m=1, 2, …, $m_{\max}$) if the maximum number of iterations is not finished. Otherwise, go to the next step. The flow diagram associated with *Step* 3 is illustrated in Fig. 5.2, where a variable node v_n is connected by three check nodes c_m, c_{m_1} and c_{m_2}, respectively. The solid directed lines indicate the messages sent from the variable node to the related check nodes in the *current* round, while the dashed directed lines indicate the messages sent from the check nodes to the variable node in the *last* round. For the check node c_m, it receives the message group ($q_{m,n}^{(0)}$, $q_{m,n}^{(1)}$) from the variable node v_n, which only depends on the messages $r_{m_1,n}$ and $r_{m_2,n}$ provided by the check nodes c_{m_1} and c_{m_2} in *last* round. In other words, ($q_{m,n}^{(0)}$, $q_{m,n}^{(1)}$) is irrelevant with $r_{m,n}$ supplied by c_m itself in the *last* round. It is the similar scenario for the check nodes c_{m_1} and c_{m_2}. Like the message from check node as stated in *Step* 2, this strategy also can fulfil that the messages sent by a variable node is extrinsic information for the check nodes linked under the condition of the Tanner graph being a tree. A

variable node sends message to the check nodes, which involve the codeword bit corresponding to that variable node, is virtually the process of exchange message between the check nodes via the variable node, and equivalent to row process for the parity check matrix. Thus, *Step* 3 is also named as horizontal step.

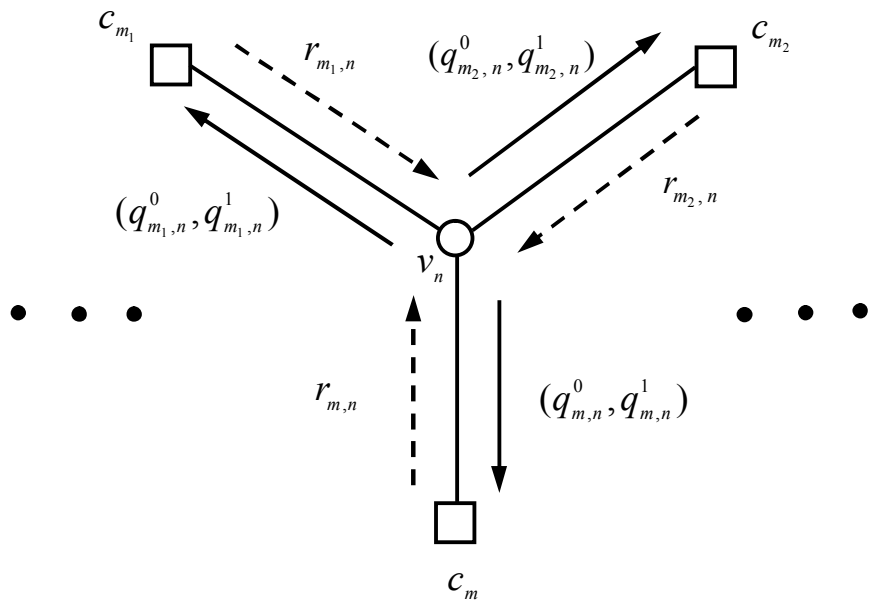

Fig. 5.2 The flow diagram of a local Tanner graph conveying the messages associated with *S*tep 3 of the BP algorithm.

In the decoding by BP algorithm, *Step* 2 and *Step* 3 are evaluated iteratively. With a parallel (flooding) schedule, one iterative process contains the computation of $r_{m,n}$ by (5-11) concurrently for all the check nodes, which is sent to variable nodes as shown in Fig. 5.1, then evaluating $q^0_{m,n}$ and $q^1_{m,n}$ by (5-17) simultaneously for all the variable nodes, which is fed back to the check nodes as shown in Fig. 5.2. With a serial (two-way) schedule, (5-11) and (5-16) are evaluated alternatively following a proper sequential order within an iteration according to the Tanner graph structure. The interested reader

can refer to the references [23][25] for more details. Generically, the decoder employing the BP algorithm is also regarded as message-passing decoder since message (extrinsic information) travels along the edges connecting the variable and check nodes in a bidirectional means.

Step 4. Final decision: Compute the *a posterior* likelihood ratio R_n, formulated by (5-4), for each codeword bit d_n. We can easily verify that R_n can be expressed as follows based on (5-4) and (5-12)

$$R_n = f_n\, \overline{R}_n = f_n \prod_{c_{m'} \in C(v_n)} r_{m',n} \tag{5-17}$$

Then the final decision is made by (5-5). Hence, the decoder can generate an estimated block of N bits as

$$\hat{\mathbf{d}} = (\hat{d}_1,\ \hat{d}_2,\ \ldots,\ \hat{d}_N)^T \tag{5-18}$$

If $\mathbf{H}\hat{\mathbf{d}} = \mathbf{0}$, where $\mathbf{H}$ is parity check matrix of the LDPC code, then the decoding algorithm halts, and $\hat{\mathbf{d}}$ is considered as a valid decoding result. Otherwise, a failure is declared if maximum number of iterations occurs without a valid decoding. □

The algorithm studied above is a standard BP algorithm. To reduce the complexity of the iterative decoding, two other simplified approaches are also introduced based on the standard one, i.e., APP-based decoding algorithm and BP-based decoding algorithm. They both offer an attractive solution to implement iterative decoding of LDPC codes with a negligible performance deficiency compared with the standard BP algorithm. For more details the interested reader can see the reference [42].

Note that so far we have studied the BP algorithm as above from the viewpoint of graph, i.e., this algorithm is suitable for a board category of error-correcting codes that can be alternatively represented by sparse Tanner graphs, not only restricted to some limited sorts of codes, like turbo codes, LDPC codes, etc. This may be the crucial reason why BP algorithm can explain many common puzzled questions encountered by the individual cases.

5.2 Extrinsic Information on Message-Passing Decoding

The problem of extrinsic information are encountered in all the iterative systems, not only for the previously studied iteratively decoding of the concatenated codes, including the well-known turbo codes, serially concatenated code, etc., but for the generic Tanner graph codes family as well, such as LDPC codes, CT codes, etc., which apply the BP algorithm to their iterative message-passing decoders. The study of the extrinsic information is thus an attractive and will be also a never-ending theme.

Recall that in order to guarantee the message to be true extrinsic information, a crucial hypothesis of Tanner graph being a tree must be made for the BP algorithm. More specifically, the exchange messages between the neighboring nodes of different types as described in *Step* 2 and 3 of the BP algorithm should be pure extrinsic information from theoretical viewpoint. For instance, in Fig. 5.1 the message $r_{m,n}$ sent from check node c_m is extrinsic information for the variable node v_n since the computation of $r_{m,n}$ has already excluded the quantities ($q_{m,n}^{(0)}$, $q_{m,n}^{(1)}$) from v_n in last round as claimed by *Step* 2, and meanwhile as a tree with loop-free structure, no any other way than the edge, which

connects c_m and v_n directly, can directly or indirectly convey the messages initially starting from v_n and eventually ending at c_m. It is the same reason for the extrinsic information sent from variable to check node as shown in Fig. 5.2.

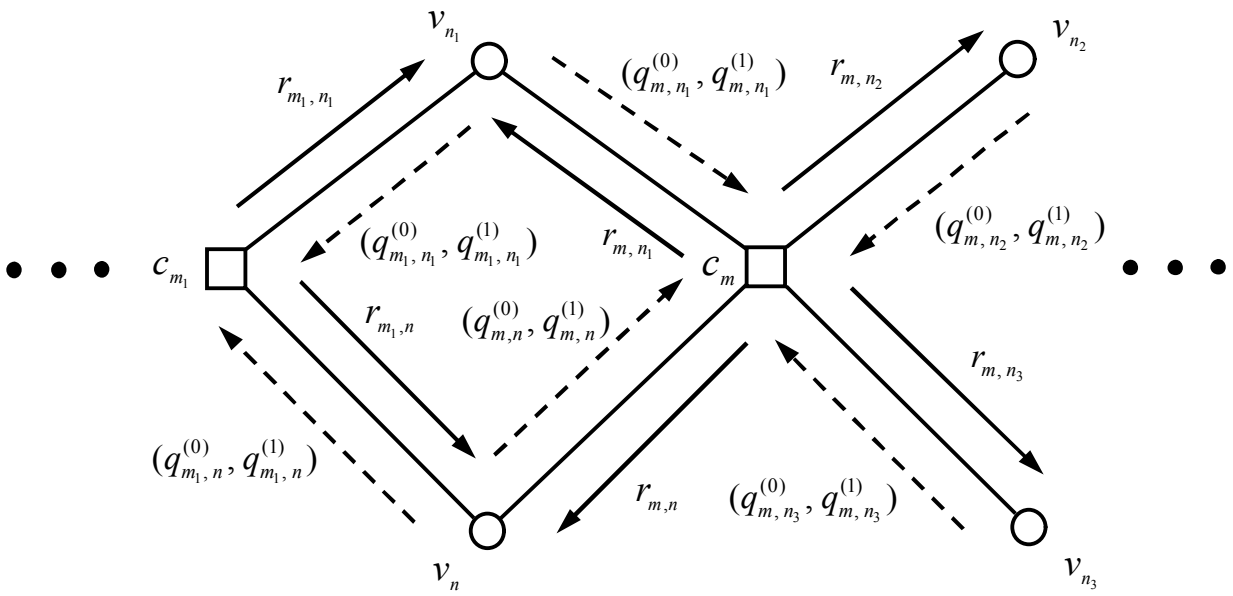

Fig. 5.3 The flow diagram of a local Tanner graph with a girth-4 loop.

However, we may encounter a quite different scene if the Tanner graph is not a tree. Based the scenario given in Fig. 5.1, a local flow diagram of another Tanner graph is shown in Fig. 5.3, where a girth-4 loop bearing the messages is denoted by $v_n \to c_{m_1} \to v_{n_1} \to c_m \to v_n$. Initially, the variable node v_n sends message $(q^{(0)}_{m_1,n}, q^{(1)}_{m_1,n})$ to check node c_{m_1} in the first round, then c_{m_1} "read" it and provides the processed message r_{m_1,n_1} to the variable node v_{n_1}, which is surely dependant on $(q^{(0)}_{m_1,n}, q^{(1)}_{m_1,n})$. Finally, the check node c_m receives the message $(q^{(0)}_{m,n_1}, q^{(1)}_{m,n_1})$ relevant with r_{m_1,n_1}, which, in turn, leads to indirectly dependent

on ($q_{m_1,n}^{(0)}$, $q_{m_1,n}^{(1)}$). Thus, the message $r_{m,n}$ sent back to v_n will inevitably dependent on the message ($q_{m_1,n}^{(0)}$, $q_{m_1,n}^{(1)}$) previously generated from the same node. This message definitely cannot be recognized as the *pure* extrinsic information for v_n due to the loopy graph. Strictly saying, the message sent to a variable/check node is called extrinsic information if this message is completely independent on any message that was previously sent out from the same node in the past. The only condition that can fully satisfy this stringent requirement is that the Tanner graph is a tree.

Actually, we have known from the previous parts of this chapter that most of Tanner graph codes contain many loops, including the turbo codes, serially concatenated codes, CT codes and LDPC codes, etc. Consequently, the BP decoding algorithm applied to the aforementioned error-correction codes is only an ideal loop-free approach and cannot give exact solutions for the loopy ones [22]. For the latter cases, we have to resort it to a loopy BP algorithm [43] with more computational complexity, and the interested reader may further study on it.

On the other hand, one may have already been aware of an intuitive fact that can seriously affect the performance of the BP algorithm, i.e., the girth of a Tanner graph. If the loop indicated in Fig. 5.3 is the shortest length circle in the whole graph, then the girth of the graph is 4. Suppose a girth-8 graph shown in Fig. 5.4 with a longer cycle of the shortest length than that depicted in Fig. 5.3. We may try to intuitively understand that the message $r_{m,n}$, received by the variable node v_n from the check node c_m, should be less dependent on the message ($q_{m_1,n}^{(0)}$, $q_{m_1,n}^{(1)}$) previously sent by v_n than the

corresponding case in Fig. 5.3. It is further informally reported [42] that BP on graph with girth 6 or lager still works well although it becomes suboptimum. However, it suffers a serious performance degrade when the girth is reduced to 4 and such a short loop occurs frequently in the graph. Therefore, increasing the girth of the graph is a general design criterion for the construction of "good" Tanner graph codes even if when the loops are unavoidable.

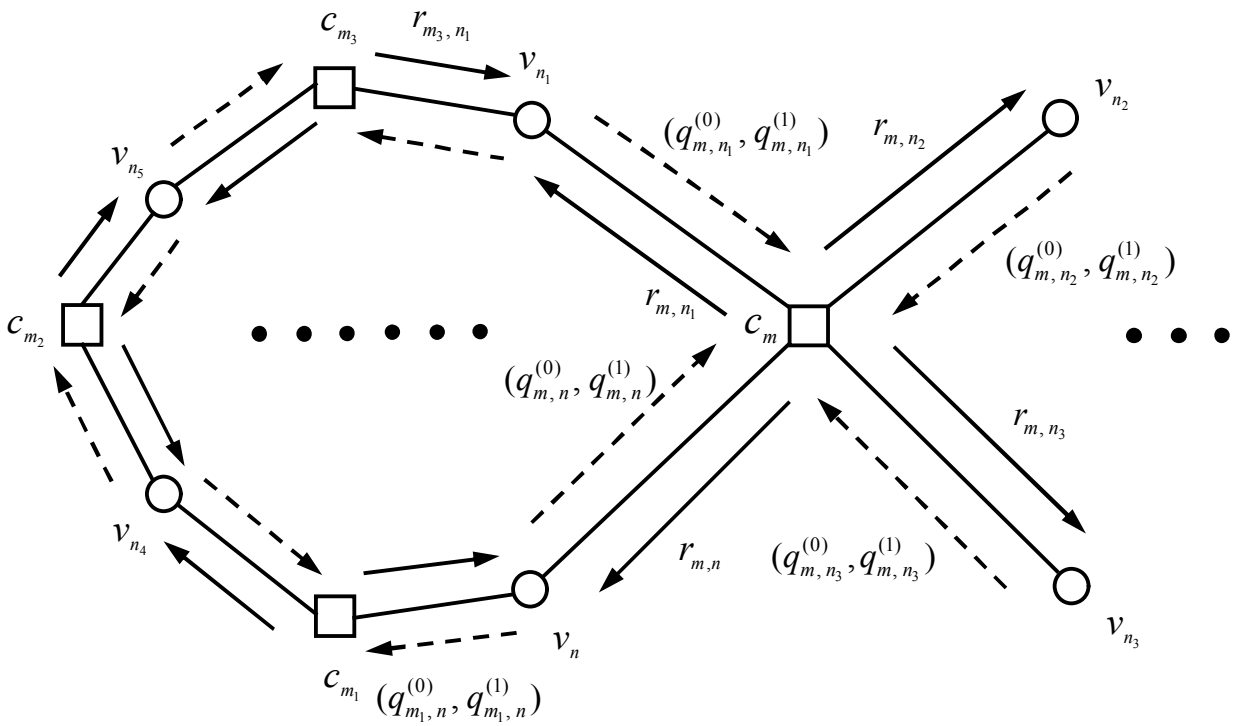

Fig. 5.4 The flow diagram of a local Tanner graph with a girth-8 loop.

5.3 Probabilities Density Evolutions and Thresholds of Message-Passing Decoding Algorithms with Discrete Message Alphabets

After studying the BP algorithm by probabilities and likelihood ratios for LDPC codes in Section 5.2, one may eagerly try to know the details of evolution process of an iterative decoding. However, it is really extremely difficult to offer an exact mathematical analysis

for the soft-decision decoding using the BP algorithm. For binary symmetric channel (BSC) (or hard-decision), Gallager suggested a pragmatic iterative approach, i.e., probabilistic decoding [1] that was an early version of the message-passing decoding based on *check-sum* and *majority-logic* decoding, for the regular LDPC codes. Even though these decoders do not perform as well as the BP algorithm as one might expect, such schemes are still very useful in practice since they are simpler and require less memory. Moreover, these simplified decoders may be helpful for the people to gain an in-depth view at iterative decoding of general graph-based LDPC codes while employing BP algorithm, including the messages passing through the incident nodes of different kinds, convergent properties, performance gaps between the thresholds and the ultimate Shannon limit, etc.

For the simplicity in the theoretically derivation of various decoding algorithms, some important and reasonable hypotheses are necessarily made.

(1) All-zero (or all-one) bits transmitted over the noisy channels.

(2) The Tanner graph of an LDPC code is a cycle-free graph (tree).

(3) The decoding of a long code exhibits a threshold phenomenon that clearly separates the region where reliable (error-free) transmission is possible from where it is not.

5.3.1 Gallager's Decoding Algorithm A over a BSC

We now investigate a very simple decoding method originally proposed by Gallager in his early thesis [1] over BSC, which is illustrated in Fig. 5.5 with input and output alphabet M =GF(2) and crossover probability denoted by p_0. In particular, the message

alphabet set is also GF(2), which implies that only 0 or 1 is passing through the edges between the incident nodes of different kinds.

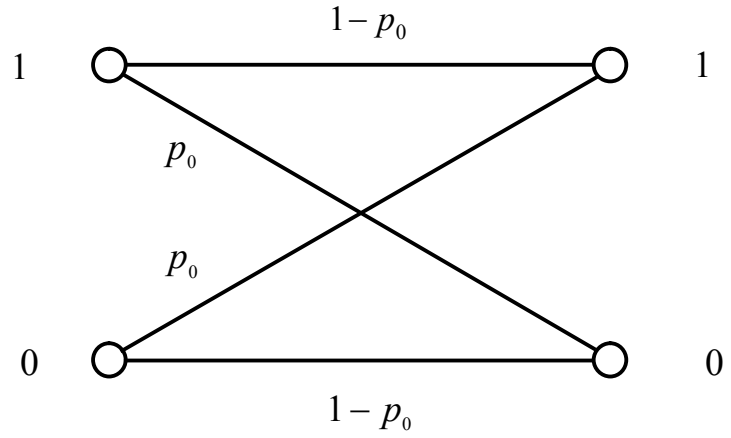

Fig. 5.5 A basic channel model of BSC with a crossover probability p_0.

Compared with the BP algorithm, Gallager decoding algorithm A is quite simple approach and can be similarly summarized as the following steps:

Step 1. **Initialization**: The decoder directly gets hard-decision output from the BSC, which is simply given as the initially received values to the corresponding variable nodes. This is different from the scenario that channel soft output is received by the decoder using BP algorithm.

Step 2. **Message from a check node (Horizontal step)**: For an arbitrary check node, it sends a message to the incident variable node indicating the modulo-2 sum of $d_c - 1$ incoming binary messages from the other incident variable nodes except the one along the same edge as the outgoing message, i.e.,

$$\Phi_c(m_1, m_2, ..., m_{d_c-1}) = \sum_{i=1}^{d_c-1} \oplus m_i \tag{5-19}$$

where $\Phi_c(\cdot)$ is the mapping function of a check node, and $m_i \in \mathrm{GF}(2)$ (i=1, ….., d_c-1) are the messages sent from d_c-1 incident variable nodes in the last decoding iteration. An illustrative diagram for *Step 2* is shown in Fig. 5.6.

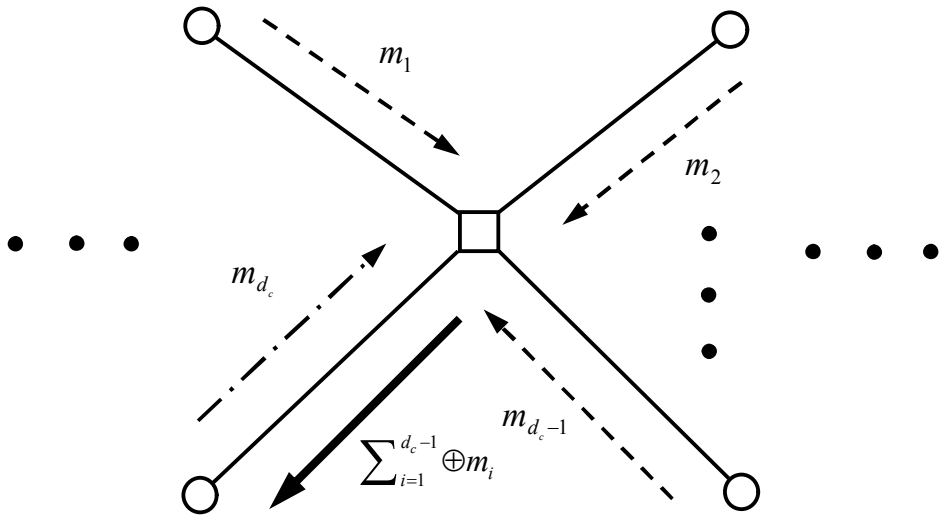

Fig. 5.6 The illustrative diagram for Gallager's decoding algorithm A for an outgoing extrinsic message from a check node to an incident variable node with d_c-1 input extrinsic messages m_i (i=1, 2,…., d_c-1).

Step 3. **Message from a variable node (Vertical step)**: For an arbitrary variable node, it sends a message to the incident check node by the given decision as follows:

$$\Phi_v(m_0, m_1', \ldots, m_{d_v-1}') = \begin{cases} \bar{m}_0 & \text{if } m_1' = m_2' = \ldots = m_{d_v-1}' = \bar{m}_0 \\ m_0 & \text{if otherwise} \end{cases} \tag{5-20}$$

where $\Phi_v(\cdot)$ and m_0 are the mapping function of a variable node and BSC hard-decision output received by this node, respectively, and $m_i' \in \mathrm{GF}(2)$ (i=1, 2, ….., d_v-1) are d_v-1 incoming binary messages from the other incident check nodes except the one along the

same edge as the outgoing message, $\bar{m}_0$ is the inverse element of m_0 in GF(2). An illustrative diagram for *Step* 3 is shown in Fig. 5.7.

The decoding strategy in (5-20) implies that the variable node sends its received hard-decision output from BSC to the incident check node unless *all* the incoming messages are *unanimous* in opposition to it, in which case the resulting value is the same as the incoming messages. In other words, the received value m_0 for a variable node cannot be rejected by the incident check node if the case that all the incoming messages give their opposite votes does not appear. Therefore, it is a majority-logic decoding method with highest degree.

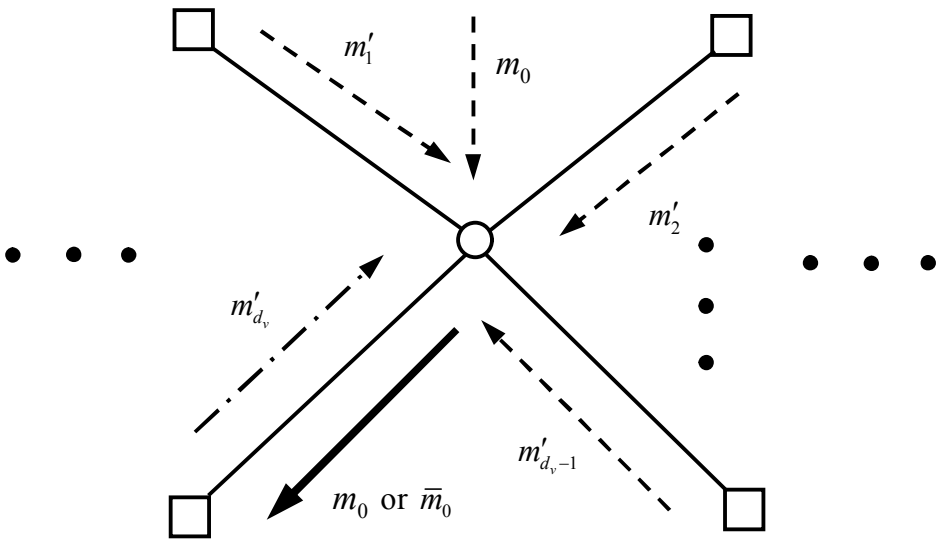

Fig. 5.7 The illustrative diagram for Gallager's decoding algorithm A for an outgoing extrinsic message from a variable node to an incident check node with d_v-1 input extrinsic messages m'_i (i=1, 2,…., d_v–1) and an initial value m_0.

Step 4. Final decision: The final hard decision is made simply based on (5-20) for every codeword bit after the maximum number of decoding iterations is performed. □

Next we will derive the density evolution of Gallager's decoding algorithm A for a regular LDPC code, which are still characterized by Tanner graph as a unified approach. The derivation is performed by an iteration-to-iteration basis.

The 1st decoding iteration: If a variable node, saying v_n, initially receives a wrong coded bit from BSC output (an event of probability of p_0), then an incident check node, saying c_m, will be unsatisfied if and only if an even number (including possible zero) of associated variable nodes other than v_n sending c_m the wrong (incorrect) coded bits (BSC outputs). As each coded bit is correctly sent to c_m with probability $1-p_0$, based on the Lemma 12.1 and *Step* 2, it is easy to check that the probability that c_m receives an even number of errors is

$$\frac{1+(1-2p_0)^{d_c-1}}{2} \tag{5-21}$$

Based on *Step* 3 an error will be *corrected* only if *all* the incident check nodes other than c_m are unsatisfied. Hence, the probability that an error bit is first received by v_n and then *corrected* by its d_v-1 incident check nodes is

$$p_0\left[\frac{1+(1-2p_0)^{d_c-1}}{2}\right]^{d_v-1} \tag{5-22}$$

On the other hand, if the variable node v_n initially receives a correct bit from BSC output, then the incident check node c_m will be unsatisfied if and only if an odd number of

variable nodes other than v_n sending c_m the wrong bits. Similarly, the probability that the bit is first *correctly* received by v_n but *incorrectly* decoded because of $d_v - 1$ unsatisfied incident check nodes is

$$(1-p_0)\left[\frac{1-(1-2p_0)^{d_c-1}}{2}\right]^{d_v-1} \tag{5-23}$$

By (5-22) and (5-23), the error probability of a coded bit after the 1st iteration (first two decoding rounds) by using Gallager's decoding algorithm A is

$$p_1 = p_0\left\{1-\left[\frac{1+(1-2p_0)^{d_c-1}}{2}\right]^{d_v-1}\right\}+(1-p_0)\left[\frac{1-(1-2p_0)^{d_c-1}}{2}\right]^{d_v-1} \tag{5-24}$$

The 2nd decoding iteration: Similar as (5-22) for the 1st decoding iteration, the probability that an error bit is first received by v_n and then *corrected* by its $d_v - 1$ incident check nodes in the 2nd decoding iteration is

$$p_0\left[\frac{1+(1-2p_1)^{d_c-1}}{2}\right]^{d_v-1} \tag{5-25}$$

Similar as (5-23) for the 1st decoding iteration, the probability that the bit is first received correctly by v_n but incorrectly decoded because of $d_v - 1$ unsatisfied incident check nodes in the 2nd decoding iteration is

$$(1-p_0)\left[\frac{1-(1-2p_1)^{d_c-1}}{2}\right]^{d_v-1} \tag{5-26}$$

By (5-25) and (5-26), the error probability of a coded bit after the 2nd iteration by using Gallager's decoding algorithm A is

$$p_2 = p_0\left\{1-\left[\frac{1+(1-2p_1)^{d_c-1}}{2}\right]^{d_v-1}\right\}+(1-p_0)\left[\frac{1-(1-2p_1)^{d_c-1}}{2}\right]^{d_v-1} \tag{5-27}$$

It immediately follows by induction that p_l with respect to l-th decoding iteration can be expressed in terms of p_{l-1} for $(l-1)$th decoding iteration and p_0 as:

$$p_l = p_0\left\{1-\left[\frac{1+(1-2p_{l-1})^{d_c-1}}{2}\right]^{d_v-1}\right\}+(1-p_0)\left[\frac{1-(1-2p_{l-1})^{d_c-1}}{2}\right]^{d_v-1} \tag{5-28}$$

For a fixed value of p_{l-1}, p_l in (5-28) is an *increasing* function of p_0. Likewise, for a fixed value of p_0, p_l is an *increasing* function of p_{l-1}. By induction we conclude that p_l is an increasing function of p_0. Obviously, p_l =0 for any l >0 if p_0 =0. Thus, the *supremum* ε^* of all values of $p_0 \in [0,1]$ must exist such that $\lim_{l\to+\infty} p_l$=0 for $0 \le p_0 < \varepsilon^*$. The value ε^* is called *threshold* of Gallager's decoding algorithm A for the codes ensemble $C^n(d_v, d_c)$ over BSC when block length n and number of decoding iterations both go to infinity. The threshold can be determined by an exact evaluation [60] in theory as well as the approximated one by numerical methods [1][8]. These values, denoted by $\varepsilon^*(A)$, are given and listed in Table 5.1 for various (d_v, d_c) pairs, where the most of the entries were always given by Gallager [1, Ch. 4.3]. The maximum allowed crossover probability ε_{opt}, corresponding to BSC capacity as code rate, is also included for an explicit comparison.

Example 5.1: Consider a rate-1/2 regular LDPC codes ensemble with d_v=5 and d_c=10 over a BSC with the crossover probability p_0=0.02. Evaluate the bit error probability in

the 3rd iteration while Gallager's algorithm A is adopted to decode the regular LDPC codes with infinite block length.

Based on (5-28) the bit error probability after the 1st iteration (l=1) is

$$p_1 = p_0 - p_0\left[\frac{1+(1-2p_0)^{d_c-1}}{2}\right]^{d_v-1} + (1-p_0)\left[\frac{1-(1-2p_0)^{d_c-1}}{2}\right]^{d_v-1}$$

$$=0.02-0.02\left[\frac{1+(1-2\times 0.02)^{9}}{2}\right]^{4} + (1-0.02)\left[\frac{1-(1-2\times 0.02)^{9}}{2}\right]^{4}$$

$$=1.03\times 10^{-2} \tag{5-29}$$

The bit error probability after the 2nd iteration (l=2) is

$$p_2 = p_0 - p_0\left[\frac{1+(1-2p_1)^{d_c-1}}{2}\right]^{d_v-1} + (1-p_0)\left[\frac{1-(1-2p_1)^{d_c-1}}{2}\right]^{d_v-1}$$

$$=0.02-0.02\left[\frac{1+(1-2\times 1.03\times 10^{-2})^{9}}{2}\right]^{4} + (1-0.02)\left[\frac{1-(1-2\times 1.03\times 10^{-2})^{9}}{2}\right]^{4}$$

$$=6.05\times 10^{-3} \tag{5-30}$$

The bit error probability after the 3rd iteration (l=3) is

$$p_3 = p_0 - p_0\left[\frac{1+(1-2p_2)^{d_c-1}}{2}\right]^{d_v-1} + (1-p_0)\left[\frac{1-(1-2p_2)^{d_c-1}}{2}\right]^{d_v-1}$$

$$=0.02-0.02\left[\frac{1+(1-2\times 6.05\times 10^{-3})^{9}}{2}\right]^{4} + (1-0.02)\left[\frac{1-(1-2\times 6.05\times 10^{-3})^{9}}{2}\right]^{4}$$

$$=3.85\times 10^{-3} \tag{5-31}$$

□

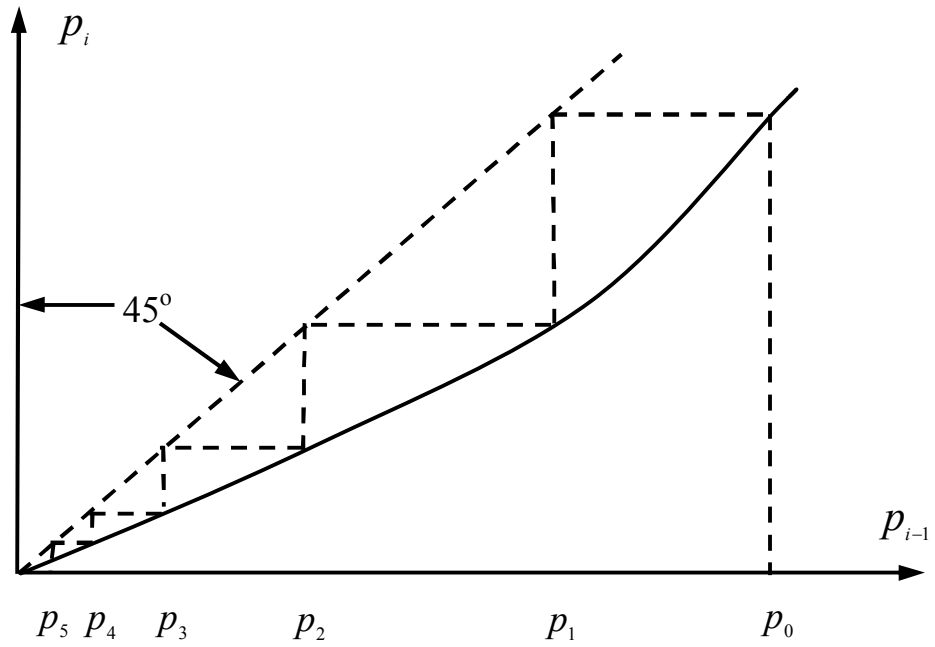

Fig. 5.8. An illustrative diagram of convergent process of p_i by adopting Gallager algorithm A under the condition of $p_0<\varepsilon^*(A)$.

The convergent process of $\left\{p_i\middle|_{i=1}^{+\infty}\right\}$ is also roughly sketched in Fig. 5.8 for $0\le p_0<\varepsilon^*$, where the ordinate (p_i) for one value of i is the abscissa (p_{i+1}) for the next, the dashed zigzag line [1] illustrates a effective method of finding p_i for each successive value of i.

More importantly, the density evolution can characterize the asymptotical behavior of the ensemble of LDPC codes with the same (d_v, d_c) and code rate using this decoding algorithm. The value ε^* clearly defines the maximum crossover probability of the BSC, greater than which the bit (or block) error probability of the decoder is strictly bounded away from zero by a constant independent of the number of decoding iterations and block length. This critical condition guarantees a decoder to achieve successful decoding with an asymptotic zero bit (or block) error probability. One can also compare ε^* with the

corresponding ε_{opt} based on the ultimate Shannon limit under the same code rate and, the gap between the two limits, i.e., theoretical potentials for the regular LDPC codes ensemble using the desired decoding algorithm, is unveiled. For example, as shown in Table 5.1, $\varepsilon(A)$ is 0.04 and 0.047 for the codes ensembles $C^n(3,6)$ and $C^n(4,8)$, respectively, over a BSC with infinite block n while ε_{opt} is 0.11 for the BSC capacity as 1/2. Thus, we conclude that $C^n(4,8)$ outperforms $C^n(3,6)$ asymptotically and statistically over a BSC since the former threshold is closer to the ultimate Shannon limit than that of the latter one.

5.3.2 Gallager's Decoding Algorithm B over a BSC

We have known that, in *Step* 3 of Gallager's decoding algorithm A, all of the other incident check nodes of a variable node must disagree with the received bit so that its value can be changed. This majority logic decoding rule is extremely strict and guarantees the received bits to be corrected with a very high probability of correctness. However, on the hand, it may not correct some of wrong bits received that only cause partial check nodes involved against them. Thus, Gallager proposed an improved version based on the algorithm A, which is now called Gallager's decoding algorithm B. Only the decision rule in *Step* 3 is slightly modified as

$$\Phi_v^{(l)}(m_0, m_1', m_2', ..., m_{d_v-1}') = \begin{cases} \bar{m}_0 & if \quad \left|\{i : m_i' = \bar{m}_0\}\right| \geq \lambda_l \\ m_0 & if \quad otherwise \end{cases} \tag{5-32}$$

Note that the above mapping function for a variable node depends on the specified decoding iteration and crossover probability p_0 of BSC. Hence, the parameter λ_l should

be determined for the l-th iteration. For an arbitrary variable node v_n with the received value m_0 from BSC and a related check node c_m, m_0 is changed in current iteration if λ_l or more check nodes on v_n except c_m are violated, otherwise, v_n simply gives m_0 to c_m. Using this criterion and the similar analysis as (5-23), we can find a recursive formulation of p_l for Gallager's decoding algorithm B

$$p_l = p_0\left(1-\sum_{u=\lambda_l}^{d_v-1}\binom{d_v-1}{u}\left[\frac{1+(1-2p_{l-1})^{d_c-1}}{2}\right]^u\left[\frac{1-(1-2p_{l-1})^{d_c-1}}{2}\right]^{d_v-1-u}\right)$$

$$+(1-p_0)\sum_{u=\lambda_l}^{d_v-1}\binom{d_v-1}{u}\left[\frac{1-(1-2p_{l-1})^{d_c-1}}{2}\right]^u\left[\frac{1+(1-2p_{l-1})^{d_c-1}}{2}\right]^{d_v-1-u} \quad (5\text{-}33)$$

where the parameter λ_l is the smallest integer satisfying the criterion that the probability of a wrong received bit corrected by the decoder is greater than or equal to the probability of a right received bit corrected by the decoder, i.e.,

$$p_0\left[\frac{1+(1-2p_{l-1})^{d_c-1}}{2}\right]^{\lambda_l}\left[\frac{1-(1-2p_{l-1})^{d_c-1}}{2}\right]^{(d_v-1)-\lambda_l}$$

$$\geq(1-p_0)\left[\frac{1-(1-2p_{l-1})^{d_c-1}}{2}\right]^{\lambda_l}\left[\frac{1+(1-2p_{l-1})^{d_c-1}}{2}\right]^{(d_v-1)-\lambda_l} \quad (5\text{-}34a)$$

Equivalently,

$$\frac{1-p_0}{p_0}\leq\left[\frac{1+(1-2p_{l-1})^{d_c-1}}{1-(1-2p_{l-1})^{d_c-1}}\right]^{2\lambda_l-(d_v-1)} \quad (5\text{-}34b)$$

Note that λ_l is an increasing function of p_{l-1}; intuitively, as p_{l-1} decreases, smaller majorities are needed to get an accurate assessment of the bit value related to the variable node v_n. The Gallager algorithm B differs from the algorithm A for d_v>3.

Example 5.2: Evaluate the bit error probability in the 3rd iteration for the same regular LDPC code and BSC with crossover probability p_0 =0.02 as Example 5.1 while Gallager's decoding algorithm B is employed for the case of infinite block length.

According to (12.65c) for the 1st decoding iteration (l=1), we have

$$\frac{1-p_0}{p_0} \le \left[\frac{1+(1-2p_0)^9}{1-(1-2p_0)^9}\right]^{2\lambda_1-4} \tag{5-35a}$$

The smallest integer λ_1 that satisfies (5-35a) is 4. Thus, based on (5-34a) the bit error probability p_1 is

$$\begin{aligned} p_1 &= p_0\left(1-\sum_{u=\lambda_1}^{d_v-1}\binom{d_v-1}{u}\left[\frac{1+(1-2p_0)^{d_c-1}}{2}\right]^u\left[\frac{1-(1-2p_0)^{d_c-1}}{2}\right]^{(d_v-1)-u}\right) \\ &+(1-p_0)\sum_{u=\lambda_1}^{d_v-1}\binom{d_v-1}{u}\left[\frac{1-(1-2p_0)^{d_c-1}}{2}\right]^u\left[\frac{1+(1-2p_0)^{d_c-1}}{2}\right]^{(d_v-1)-u} \\ &= 0.02-0.02\sum_{u=4}^{4}\binom{4}{u}\left[\frac{1+(1-2\times0.02)^9}{2}\right]^u\left[\frac{1-(1-2\times0.02)^9}{2}\right]^{4-u} \\ &+(1-0.02)\sum_{u=4}^{4}\binom{4}{u}\left[\frac{1-(1-2\times0.02)^9}{2}\right]^u\left[\frac{1+(1-2\times0.02)^9}{2}\right]^{4-u} = 1.03\times10^{-2} \end{aligned} \tag{5-35b}$$

Actually, in the 1st decoding iteration Gallager's decoding algorithms A and B are the same since λ_1=4. In the 2nd decoding iteration (l=2), we have

$$\frac{1-p_0}{p_0} \le \left[\frac{1+(1-2p_1)^9}{1-(1-2p_1)^9}\right]^{2\lambda_2-4} \tag{5-36a}$$

The smallest integer λ_2, which satisfies (5-36a), is 3 that is less than λ_1=4. Similarly, the bit error probability p_2 is

$$p_2 = p_0\left(1-\sum_{u=\lambda_2}^{d_v-1}\binom{d_v-1}{u}\left[\frac{1+(1-2p_1)^{d_c-1}}{2}\right]^u\left[\frac{1-(1-2p_1)^{d_c-1}}{2}\right]^{(d_v-1)-u}\right)$$

$$+(1-p_0)\sum_{u=\lambda_2}^{d_v-1}\binom{d_v-1}{u}\left[\frac{1-(1-2p_1)^{d_c-1}}{2}\right]^u\left[\frac{1+(1-2p_1)^{d_c-1}}{2}\right]^{(d_v-1)-u}$$

$$=0.02-0.02\sum_{u=3}^{4}\binom{4}{u}\left[\frac{1+(1-2\times1.03\times10^{-2})^9}{2}\right]^u\left[\frac{1-(1-2\times1.03\times10^{-2})^9}{2}\right]^{4-u}$$

$$+(1-0.02)\sum_{u=3}^{4}\binom{4}{u}\left[\frac{1-(1-2\times1.03\times10^{-2})^9}{2}\right]^u\left[\frac{1+(1-2\times1.03\times10^{-2})^9}{2}\right]^{4-u}=3.06\times10^{-3} \quad (5\text{-}36b)$$

In the 3rd decoding iteration (l=3), we have

$$\frac{1-p_0}{p_0}\le\left[\frac{1+(1-2p_2)^9}{1-(1-2p_2)^9}\right]^{2\lambda_3-4} \quad (5\text{-}37a)$$

The smallest integer λ_3, which satisfies (5-37a), is 3 that equals to λ_2=3. Similarly, the bit error probability p_3 is

$$p_3 = p_0\left(1-\sum_{u=\lambda_3}^{d_v-1}\binom{d_v-1}{u}\left[\frac{1+(1-2p_2)^{d_c-1}}{2}\right]^u\left[\frac{1-(1-2p_2)^{d_c-1}}{2}\right]^{(d_v-1)-u}\right)$$

$$+(1-p_0)\sum_{u=\lambda_3}^{d_v-1}\binom{d_v-1}{u}\left[\frac{1-(1-2p_2)^{d_c-1}}{2}\right]^u\left[\frac{1+(1-2p_2)^{d_c-1}}{2}\right]^{(d_v-1)-u}$$

$$=0.02-0.02\sum_{u=3}^{4}\binom{4}{u}\left[\frac{1+(1-2\times3.06\times10^{-3})^9}{2}\right]^u\left[\frac{1-(1-2\times3.06\times10^{-3})^9}{2}\right]^{4-u}$$

$$+(1-0.02)\sum_{u=3}^{4}\binom{4}{u}\left[\frac{1-(1-2\times3.06\times10^{-3})^9}{2}\right]^u\left[\frac{1+(1-2\times3.06\times10^{-3})^9}{2}\right]^{4-u}=1.58\times10^{-4} \quad (5\text{-}37b)$$

Compared with the bit error probabilities obtained from the above two examples using Gallager's decoding algorithm A and B, the results with algorithm B are smaller than

those with algorithm A except the same in the 1st iteration. Obviously, Gallager's algorithm B exhibits a faster convergent process in contrast to the algorithm A under the same conditions. □

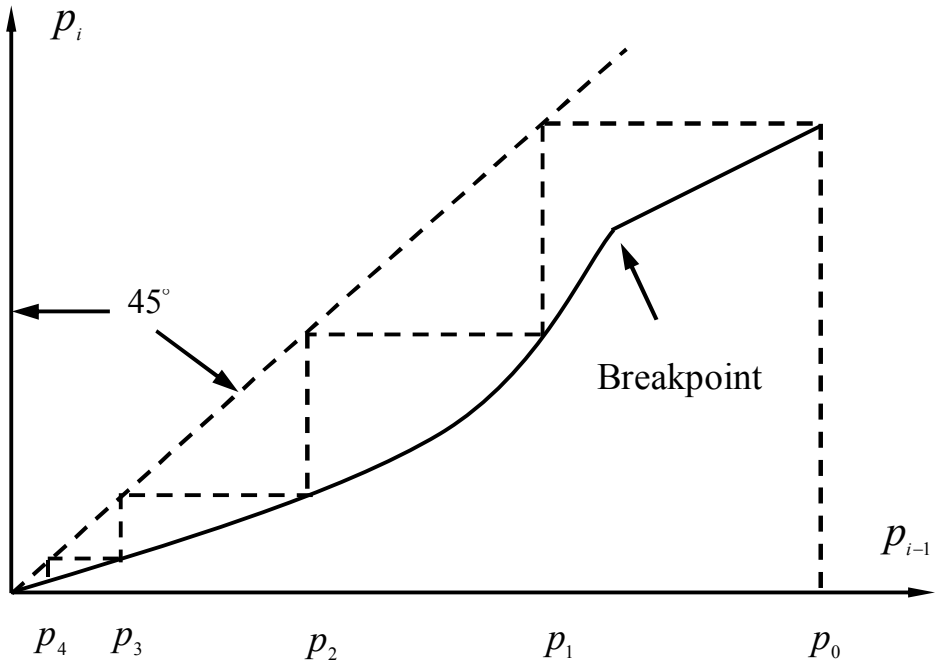

Fig. 5.9. An illustrative diagram of convergent process of p_i by using Gallager algorithm B under the condition of d_v>3 and $p_0<\varepsilon^*(B)$.

Similar as for the algorithm A, there also exists the supremum ε^* of all values of $p_0 \in [0, 1]$ such that $\lim_{l\to+\infty} p_l = 0$ for $p_0 < \varepsilon^*$. The convergent process of $\{p_l \mid_{l=1}^{+\infty}\}$ is also approximately depicted in Fig. 5.9 for $p_0<\varepsilon^*$ and when λ_l is dynamically determined subject to the constraint (5-34b), and the breakpoint may be caused by the changes of λ_l during the iterations [1]. To distinguish the threshold $\varepsilon^*(A)$ for Gallager's decoding algorithm A, $\varepsilon^*(B)$ is denoted for the threshold for Gallager's decoding algorithm B. The significant importance of $\varepsilon^*(B)$ is quite similar as that of $\varepsilon^*(A)$ that has been explained

explicitly. The values $\varepsilon^*(B)$ obtained by numerical simulations for the same ensembles of regular LDPC codes are also given and listed in Table 5.1.

5.3.3 BSC with Erasures Inside the Iterative Decoder

Now we extend ideas of Gallager's decoding algorithms by allowing erasures in the decoder. In order to be well presented, the binary symbols +1 and −1, corresponding to 0 and 1 in GF(2), respectively, are used for the input and output of BSC as shown in Fig. 5.5. However, the message alphabet set is M_E={+1, 0, −1}, where the symbol 0 output from a variable or check node implies the low reliability for the node that cannot make any clear decision on +1 and −1 based on the incoming messages. Note that BSC itself does not perform any operation of erasure. In the 1st decoding iteration, a variable initially receives outputs (+1 or −1) from BSC and perform the same decoding as Gallager's decoding algorithm A or B. Compared with the Gallager's well-known algorithms, there has some significant changes with erasures inside the iterative decoder.

In *Step* 2, the message from an arbitrary check node c_m to an incident variable node v_n is equal to the product of the other d_c −1 incoming messages except the message from v_n, i.e.,

$$\Phi_c^{(l)}\left(m_1, m_2, ..., m_{d_c-1}\right) = \prod_{i=1}^{d_c-1} m_i \tag{5-38}$$

Note that $m_i \in M_E$ (i=1, 2, ….., d_c −1) for lth ($l \geq 2$) iteration, which means that c_m will give value 0, i.e., an erasure, to the incident variable node v_n if an erasure exists in the

d_c-1 incoming messages generated by d_c-1 variable nodes in the last iteration. However, in the 1st iteration m_i is either +1 or −1 and thus c_m sends to v_n without erasure.

In *Step* 3, the message from a variable node v_n to an incident check node c_m takes the real sum of the other d_v-1 incoming messages, except the message from c_m, and plus w_l times the initially received message $m_0 \in \{+1, -1\}$ by v_n, i.e.,

$$\Phi_v^{(l)}(m_0, m_1, ..., m_{d_v-1}) = sign\left(w_l m_0 + \sum_{i=1}^{d_v-1} m_i\right) \tag{5-39}$$

where the parameter w_l (w_l>0), involving with the l-th decoding iteration, is to be specified. The check node c_m can only receive an opposite value $-m_0$ if both conditions $(w_l \mid m_0 \mid)\sum_{i=1}^{d_c-1} m_i$ <0 and $\sum_{i=1}^{d_c-1} m_i > w_l \mid m_0 \mid$ are satisfied, while c_m is given an erasure (value 0) if $\sum_{i=1}^{d_c-1} m_i = -w_l \mid m_0 \mid$. Note that both check and variable node may result in erasures in the course of decoding iterations while using this algorithm, which differs greatly from Gallager's decoding algorithm A and B.

For check node processing with erasures in the decoder, let $q_k^{(l)}$ ($k = -1, +1, 0$) be the probabilities of message sent from a check node to a related variable node in the lth decoding iteration, which can be derived based on the lemma 5.1 and taking erasures into account,

$$q_{+1}^{(l)} = \frac{1}{2}\left[\left(p_{+1}^{(l-1)} + p_{-1}^{(l-1)}\right)^{d_c-1} + \left(p_{+1}^{(l-1)} - p_{-1}^{(l-1)}\right)^{d_c-1}\right] \tag{5-40a}$$

$$q_{-1}^{(l)} = \frac{1}{2}\left[\left(p_{+1}^{(l-1)} + p_{-1}^{(l-1)}\right)^{d_c-1} - \left(p_{+1}^{(l-1)} - p_{-1}^{(l-1)}\right)^{d_c-1}\right] \tag{5-40b}$$

$$q_0^{(l)} = 1 - \left(1 - p_0^{(l-1)}\right)^{d_c-1} \tag{5-40c}$$

where $p_k^{(l-1)}$ ($k=-1, +1, 0$) are the probabilities of message sent from a variable node to a related check node in the $(l-1)$th decoding iteration. In particular, $p_0^{(0)}=0$, $p_{+1}^{(0)}=p_{-1}^{(0)}=1/2$ since no erasures for BSC output and equal likelihood for binary source information symbols. Hence, $p_{+1}^{(l)}$, $p_{-1}^{(l)}$ and $p_0^{(l)}$ ($l\geq 0$) can be expressed as

$$\begin{aligned}
p_{+1}^{(l)} &= p_{+1}^{(0)} \sum_{(i,j):i>j-w_l} \binom{d_v-1}{(d_v-1)-i-j} \left(q_{+1}^{(l-1)}\right)^i \left(q_{-1}^{(l-1)}\right)^j \left(q_0^{(l-1)}\right)^{(d_v-1)-i-j} \\
&\quad + p_{-1}^{(0)} \sum_{(i,j):i>j+w_l} \binom{d_v-1}{(d_v-1)-i-j} \left(q_{+1}^{(l-1)}\right)^i \left(q_{-1}^{(l-1)}\right)^j \left(q_0^{(l-1)}\right)^{(d_v-1)-i-j} \\
&= \frac{1}{2} \sum_{(i,j):i>j-w_l} \binom{d_v-1}{(d_v-1)-i-j} \left(q_{+1}^{(l-1)}\right)^i \left(q_{-1}^{(l-1)}\right)^j \left(q_0^{(l-1)}\right)^{(d_v-1)-i-j} \\
&\quad + \frac{1}{2} \sum_{(i,j):i>j+w_l} \binom{d_v-1}{(d_v-1)-i-j} \left(q_{+1}^{(l-1)}\right)^i \left(q_{-1}^{(l-1)}\right)^j \left(q_0^{(l-1)}\right)^{(d_v-1)-i-j}
\end{aligned} \tag{5-41a}$$

$$\begin{aligned}
p_{-1}^{(l)} &= p_{+1}^{(0)} \sum_{(i,j):i<j-w_l} \binom{d_v-1}{(d_v-1)-i-j} \left(q_{+1}^{(l-1)}\right)^i \left(q_{-1}^{(l-1)}\right)^j \left(q_0^{(l-1)}\right)^{(d_v-1)-i-j} \\
&\quad + p_{-1}^{(0)} \sum_{(i,j):i<j+w_l} \binom{d_v-1}{(d_v-1)-i-j} \left(q_{+1}^{(l-1)}\right)^i \left(q_{-1}^{(l-1)}\right)^j \left(q_0^{(l-1)}\right)^{(d_v-1)-i-j} \\
&= \frac{1}{2} \sum_{(i,j):i<j-w_l} \binom{d_v-1}{(d_v-1)-i-j} \left(q_{+1}^{(l-1)}\right)^i \left(q_{-1}^{(l-1)}\right)^j \left(q_0^{(l-1)}\right)^{(d_v-1)-i-j} \\
&\quad + \frac{1}{2} \sum_{(i,j):i<j+w_l} \binom{d_v-1}{(d_v-1)-i-j} \left(q_{+1}^{(l-1)}\right)^i \left(q_{-1}^{(l-1)}\right)^j \left(q_0^{(l-1)}\right)^{(d_v-1)-i-j}
\end{aligned} \tag{5-41b}$$

$$\begin{aligned}
p_0^{(l)} &= 1 - p_{+1}^{(l)} - p_{-1}^{(l)} \\
&= p_{+1}^{(0)} \sum_{(i,j):i=j-w_l} \binom{d_v-1}{(d_v-1)-i-j} \left(q_{+1}^{(l-1)}\right)^i \left(q_{-1}^{(l-1)}\right)^j \left(q_0^{(l-1)}\right)^{(d_v-1)-i-j}
\end{aligned}$$

$$+p_{-1}^{(0)}\sum_{(i,j):i=j+w_l}\binom{d_v-1}{(d_v-1)-i-j}\left(q_{+1}^{(l-1)}\right)^i\left(q_{-1}^{(l-1)}\right)^j\left(q_0^{(l-1)}\right)^{(d_v-1)-i-j}$$

$$=\frac{1}{2}\sum_{(i,j):i=j-w_l}\binom{d_v-1}{(d_v-1)-i-j}\left(q_{+1}^{(l-1)}\right)^i\left(q_{-1}^{(l-1)}\right)^j\left(q_0^{(l-1)}\right)^{(d_v-1)-i-j}$$

$$+\frac{1}{2}\sum_{(i,j):i=j+w_l}\binom{d_v-1}{(d_v-1)-i-j}\left(q_{+1}^{(l-1)}\right)^i\left(q_{-1}^{(l-1)}\right)^j\left(q_0^{(l-1)}\right)^{(d_v-1)-i-j}$$

$$=\frac{1}{2}\sum_{(i,j):|i-j|=w_l}\binom{d_v-1}{(d_v-1)-i-j}\left(q_{+1}^{(l-1)}\right)^i\left(q_{-1}^{(l-1)}\right)^j\left(q_0^{(l-1)}\right)^{(d_v-1)-i-j} \quad (5\text{-}41c)$$

Thus, the probability of message sent from a variable node to an incident check node can be evaluated by using (5-40) and (5-41) iteratively. Then, the bit error probability denoted by $p_e^{(l)}$ in variable node prospect in l-th decoding iteration can be derived by (5-41a) and (5-41b), respectively.

$$p_e^{(l)}=\Pr\{\text{“+1” is decoded/“–1” is sent to BSC}\}\times\Pr\{\text{“–1” is sent to BSC}\}$$
$$+\Pr\{\text{“–1” is decoded/“+1” is sent to BSC}\}\times\Pr\{\text{“+1” is sent to BSC}\} \quad (5\text{-}42)$$

where the 1st probability in (5-42) corresponding to (5-41a) is

$$\Pr\{\text{“+1” is decoded/“–1” is sent to BSC}\}\times\Pr\{\text{“–1” is sent to BSC}\}$$
$$=\frac{p}{2}\sum_{(i,j):i>j-w_l}\binom{d_v-1}{(d_v-1)-i-j}\left(q_{+1}^{(l-1)}\right)^i\left(q_{-1}^{(l-1)}\right)^j\left(q_0^{(l-1)}\right)^{(d_v-1)-i-j}$$

$$+\frac{1-p}{2}\sum_{(i,j):i>j+w_l}\binom{d_v-1}{(d_v-1)-i-j}\left(q_{+1}^{(l-1)}\right)^i\left(q_{-1}^{(l-1)}\right)^j\left(q_0^{(l-1)}\right)^{(d_v-1)-i-j} \quad (5\text{-}43a)$$

where p is the crossover probability of BSC and the 2nd probability in (5-42) related to (5-41b) is

$$\Pr\{\text{“–1” is decoder/“+1” is sent to BSC}\}\times\Pr\{\text{“+1” is sent to BSC}\}$$

$$=\frac{1-p}{2}\sum_{(i,j):i<j-w_l}\binom{d_v-1}{(d_v-1)-i-j}\left(q_{+1}^{(l-1)}\right)^i\left(q_{-1}^{(l-1)}\right)^j\left(q_0^{(l-1)}\right)^{(d_v-1)-i-j}$$

$$+\frac{p}{2}\sum_{(i,j):i<j+w_l}\binom{d_v-1}{(d_v-1)-i-j}\left(q_{+1}^{(l-1)}\right)^i\left(q_{-1}^{(l-1)}\right)^j\left(q_0^{(l-1)}\right)^{(d_v-1)-i-j} \tag{5-43b}$$

Hence, using (5-43a) and (5-43b), (5-42) can be further expressed as

$$p_e^{(l)}=\frac{1-p}{2}\sum_{(i,j):|i-j|>w_l}\binom{d_v-1}{(d_v-1)-i-j}\left(q_{+1}^{(l-1)}\right)^i\left(q_{-1}^{(l-1)}\right)^j\left(q_0^{(l-1)}\right)^{(d_v-1)-i-j}$$

$$+\frac{p}{2}\sum_{(i,j):|i-j|<w_l}\binom{d_v-1}{(d_v-1)-i-j}\left(q_{+1}^{(l-1)}\right)^i\left(q_{-1}^{(l-1)}\right)^j\left(q_0^{(l-1)}\right)^{(d_v-1)-i-j} \tag{5-44}$$

The erasure probability denoted by $p_E^{(l)}$ in variable node prospect regarding the l-th decoding iteration can be derived from (5-41c), i.e.,

$$p_E^{(l)}=\Pr\{\text{“0” is decoded}/\text{“–1” is sent to BSC}\}\times\Pr\{\text{“–1” is sent to BSC}\}$$

$$+\Pr\{\text{“0” is decoded}/\text{“+1” is sent to BSC}\}\times\Pr\{\text{“+1” is sent to BSC}\} \tag{5-45}$$

where

$$\Pr\{\text{“0” is decoded}/\text{“–1” is sent to BSC}\}\times\Pr\{\text{“–1” is sent to BSC}\}$$

$$=\frac{p}{2}\sum_{(i,j):i=j-w_l}\binom{d_v-1}{(d_v-1)-i-j}\left(q_{+1}^{(l-1)}\right)^i\left(q_{-1}^{(l-1)}\right)^j\left(q_0^{(l-1)}\right)^{(d_v-1)-i-j}$$

$$+\frac{1-p}{2}\sum_{(i,j):i=j+w_l}\binom{d_v-1}{(d_v-1)-i-j}\left(q_{+1}^{(l-1)}\right)^i\left(q_{-1}^{(l-1)}\right)^j\left(q_0^{(l-1)}\right)^{(d_v-1)-i-j} \tag{5-46a}$$

$$\Pr\{\text{“0” is decoded}/\text{“+1” is sent to BSC}\}\times\Pr\{\text{“+1” is sent to BSC}\}$$

$$=\frac{1-p}{2}\sum_{(i,j):i=j-w_l}\binom{d_v-1}{(d_v-1)-i-j}\left(q_{+1}^{(l-1)}\right)^i\left(q_{-1}^{(l-1)}\right)^j\left(q_0^{(l-1)}\right)^{(d_v-1)-i-j}$$

$$+\frac{p}{2}\sum_{(i,j):i=j+w_l}\binom{d_v-1}{(d_v-1)-i-j}\left(q_{+1}^{(l-1)}\right)^i\left(q_{-1}^{(l-1)}\right)^j\left(q_0^{(l-1)}\right)^{(d_v-1)-i-j} \tag{5-46b}$$

Hence, using (5-46a) and (5-46b), (5-45) can be further expressed as

$$p_E^{(l)} = \frac{1}{2}\sum_{(i,j):i=j-w_l}\binom{d_v-1}{(d_v-1)-i-j}\left(q_{+1}^{(l-1)}\right)^i\left(q_{-1}^{(l-1)}\right)^j\left(q_0^{(l-1)}\right)^{(d_v-1)-i-j}$$

$$+\frac{1}{2}\sum_{(i,j):i=j+w_l}\binom{d_v-1}{(d_v-1)-i-j}\left(q_{+1}^{(l-1)}\right)^i\left(q_{-1}^{(l-1)}\right)^j\left(q_0^{(l-1)}\right)^{(d_v-1)-i-j}$$

$$=\frac{1}{2}\sum_{(i,j):|i-j|=w_l}\binom{d_v-1}{(d_v-1)-i-j}\left(q_{+1}^{(l-1)}\right)^i\left(q_{-1}^{(l-1)}\right)^j\left(q_0^{(l-1)}\right)^{(d_v-1)-i-j} = p_0^{(l)} \tag{5-47}$$

The erasure probability $p_E^{(l)}$ is the same as the probability $p_0^{(l)}$ in the same decoding iteration. This result is not surprising because an event, where an erasure is resulted in from the decoder output, is equivalent to an event, where a binary symbol (+1 or –1) transmitted over BSC is decoded as neither +1 or –1 by the decoder.

Determining the sequence of weights w_l ($l \geq 1$): At $l=0$ the incoming messages (binary symbol +1 or –1) at the variable nodes are all from BSC and hence no weight is used. However, at any given step $l \geq 1$, the weight w_l is needed in order to evaluate the density evolution ($p_{+1}^{(l)}$, $p_{-1}^{(l)}$, $p_0^{(l)}$) and error and erasure probabilities ($p_e^{(l)}$, $p_E^{(l)}$). Note that different choices of the weights definitely results in different results. There are two basic ways to determine the weights [9]

(1) **Minimizing the linear combination of bit error and erasure probabilities**

Assuming all +1's symbols (all-zero codeword bits) transmitted over BSC, a good sequence of weights can be determined by *minimizing* the following function at l th decoding iteration

$$f(p_{-1}^{(l)}, p_0^{(l)}) = p_{-1}^{(l)} + \alpha p_0^{(l)} \tag{5-48}$$

where α is a positive number, e.g., α=1/2. The optimum weights w_1^*, w_2^*, …., w_l^* are determined by means of dynamic programming. For the (3, 6) regular LDPC code, the optimum weight sequence [9] found by this method is w_1^*=2, and w_l^*=1 for $l \geq 2$ The advantage of such an approach is that it is widely applicable regardless of how many alternative maps at any step and regardless of how many levels of quantization we have. The major drawback for this scheme is computationally intensive and it is quickly infeasible if the size of the message alphabet set becomes large.

(2) Maximizing the capacity of binary erasure channel (BEC)

Consider a binary erasure channel (BEC) with a crossover probability of p and an erasure probability of ε, the capacity of BEC is expressed as follows

$$C_{BEC}(\varepsilon, p)=1-\varepsilon+h(\varepsilon)-\bar{h}(\varepsilon, p) \tag{5-49}$$

where $h(\varepsilon)$ and $\bar{h}(\varepsilon, p)$ are binary entropy functions and defined as

$$h(\varepsilon)=-\varepsilon \log_2 \varepsilon-(1-\varepsilon)\log_2(1-\varepsilon) \tag{5-50a}$$

$$\bar{h}(\varepsilon, p)=-\varepsilon \log_2 \varepsilon - p \log_2 p-(1-p-\varepsilon)\log_2(1-p-\varepsilon) \tag{5-50b}$$

The derivation of $C_{BEC}(\varepsilon, p)$ can be referred to Appendix F. Assuming that w_l ($l \geq 1$) is a positive integer that belong to {0, 1, 2, ….., d_v-1}, the optimum weights are specified by maximizing (5-49) as [9]

$$C_{BEC}\left(p_0^{(l)}, p_{-1}^{(l)}\right)\Big|_{w_l^*} = \max_{w_l \in \{0,1,\ldots,d_v-1\}} C_{BEC}\left(p_0^{(l)}, p_{-1}^{(l)}\right) \tag{5-51}$$

The above criterion is equal effectiveness but less computationally intensive as the first method. For the (3, 6) regular LDPC code this lead to the same sequence of optimum weights as those by the first method.

d_v	d_c	Rate	$\varepsilon^*(A)$	$\varepsilon^*(B)$	$\varepsilon^*(E)$	$\varepsilon^*(BP)$	ε_{opt}
3	6	0.5	0.04	0.04	0.07	0.084	0.11
4	8	0.5	0.047	0.051	0.059	0.076	0.11
5	10	0.5	0.027	0.041	0.055	0.068	0.11
3	5	0.4	0.061	0.061	0.096	0.113	0.146
4	6	0.333	0.066	0.074	0.09	0.116	0.174
3	4	0.25	0.106	0.106	0.143	0.167	0.215

Table 5.1 Threshold ε^* for various regular LDPC codes using Gallager Algorithm A and B over BSC, Algorithm E over BEC, BP Algorithm over BSC and maximum allowed threshold value ε_{opt}.

Table 5.1 also gives the thresholds, denoted by $\varepsilon^*(E)$ and obtained by numerical simulations using the 2nd approach for optimum weights w_l, for decoding of regular LDPC codes with erasures in the iterative decoder over BSC, which are significantly larger than their corresponding entries for Gallager's decoding algorithm A and B. Particularly impressive is the performance of the regular (3, 6) codes ensemble with a threshold 0.07, which is very close to 0.084, i.e., the threshold of BP algorithm over BSC.

This gives us a very useful and important conclusion that a very simple decoder with erasures performs almost as well as a belief-propagation decoder.

For BP algorithm with continuous message alphabets, we will analyze its density evolution and obtain the thresholds in Section 5.5.

5.3.4 Bit error rate (BER) performance comparison for various decoding algorithms over BSC

The performance of the regular LDPC codes using various decoding algorithms over BSC can be demonstrated by means of bit error probability versus crossover probability. The simulation results are shown in Fig. 5.10, where a rate-1/2 (3, 6) regular LDPC code with Gallager's decoding algorithm A, the decoding algorithm using inside erasures and BP decoding algorithm are adopted with block lengths 10^3, 10^4 and 10^5 for each algorithm. The thresholds 0.04, 0.07 and 0.084 are clearly indicated for the three algorithms, respectively. For each algorithm the curves gradually move to the related ultimate threshold with the increasing block length. The BP decoding algorithm is the best one of the three approaches while a decoder with inside erasures is a very effective way to improve the performance of the decoding algorithm. These properties are also presented by another view listed in Table 5.1.

Further discussions on the quantized continuous channels with erasures in both channel and decoder: Motivated by the significant increase in the threshold for the BSC when allowing the erasures in the decoder, we can apply the similar decoder for LDPC codes to generic continuous output channels. First, a symmetric threshold τ around zero is used and finalized to quantize the continuous output of the channel into negative values,

erasures and positive values, depending on whether the received channel soft output r fulfills $r \le -\tau$, $-\tau < r < \tau$ or $r \ge \tau$, respectively. In other words, the binary-input continuous -output channel becomes a BEC. Then we can now apply exactly the same algorithm to determine the optimum weight w_l that maximizes (12.82) at each decoding iteration. Note that in this scenario the erasures exist in both channel and iterative decoder. For the binary-input AWGN channel with ternary quantization and erasures and (3,6) regular LDPC code, the threshold σ^* [9] is 0.743, which corresponds to a raw bit error rate of about $Q(1/\sigma^*)$=0.089 if the continuous channel output is quantized by 1-bit and decided by hard-decision. Note that the threshold for the BSC with erasures only inside the decoder is 0.07 as list in Table 5.1 under the same conditions.

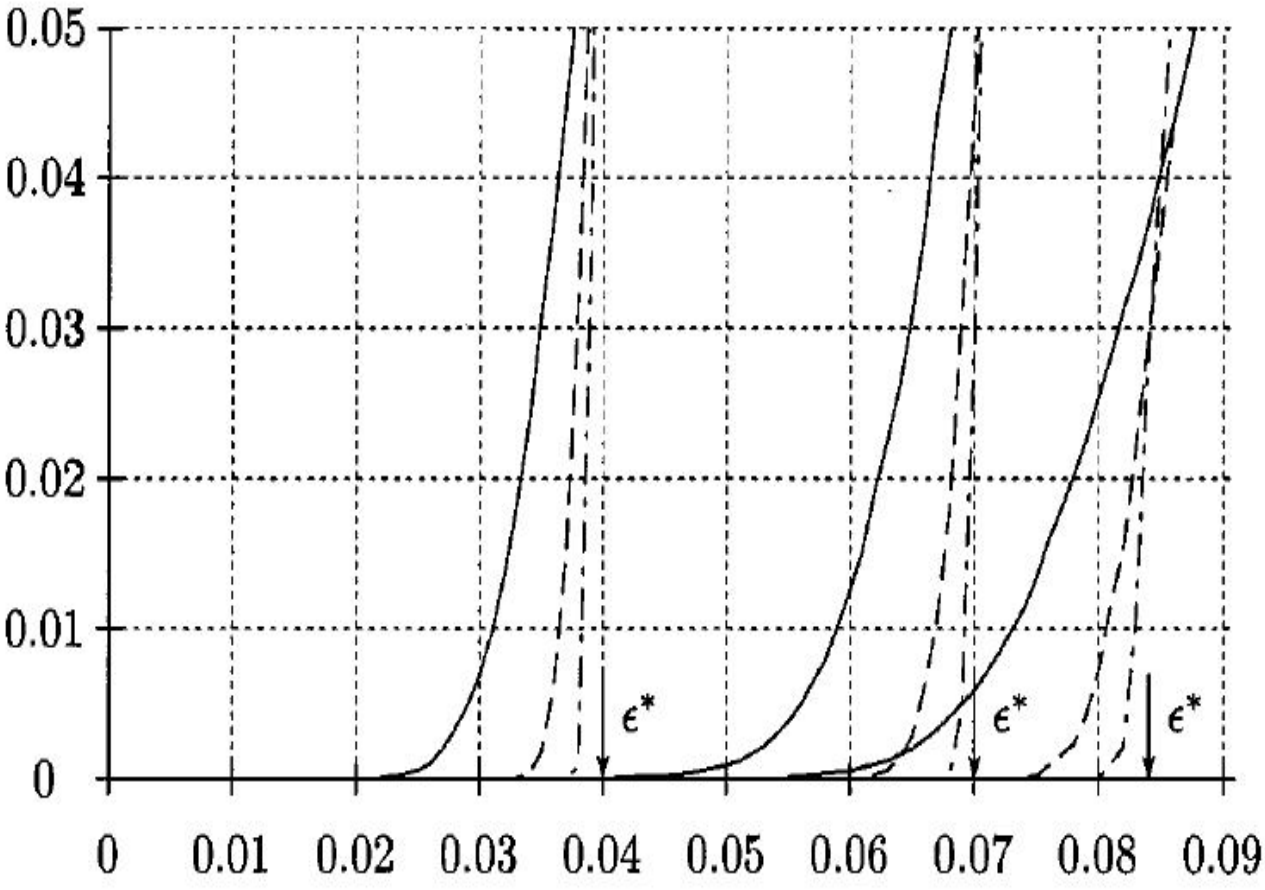

Fig. 5.10 The bit error probability versus crossover probability for rate-1/2 (3, 6) regular LDPC codes ensemble using Gallager's algorithm A and BP algorithm shown by leftmost and rightmost curves in a BSC, and the algorithm with inside erasures shown by the middle curves in a BSC. The solid, dotted-dashed and dashed lines correspond to block lengths

5.4 Accurate Representations of LLR-Based Belief Propagation Algorithms

In practice, using LLRs as extrinsic (incoming or outgoing) messages offers great implementation advantages over using probabilities or likelihood ratios, which have been adopted for the standard BP algorithm in Section 5.1. The reason is that multiplications in vertical *step* (*step* 3) are replaced by additions and the normalization step is completely eliminated. In this section, the standard BP algorithm is first described in terms of LLR for its generated extrinsic messages. This modified version is called LLR-BP algorithm [42][46][49][50] in order to slightly distinguish it from the aforementioned standard BP algorithm for the sake of description convenience, although both mean the iterative LDPC decoding algorithm employing belief propagation principle. Several different accurate representations [42][46] of LLR-BP approach are given in this section. We then describe LLR-BP decoding alternatives that are much easy to implement and can significantly reduce the decoding complexity, delay and processing storage, etc.

The LLRs, associated with the extrinsic message from a check node c_m to the related variable node v_n and the extrinsic message from v_n to c_m, are given as follows based on (12.42) and (12.46b), respectively.

$$L_{m,n} = \log r_{m,n} = \log\left[\frac{\Pr(S(\{c_m\}) = 1 | d_n = 0, \{r_l;|_{l=1}^{N}\})}{\Pr(S(\{c_m\}) = 1 | d_n = 1, \{r_l;|_{l=1}^{N}\})}\right] \tag{5-52}$$

and

$$Z_{m,n}=\log s_{m,n}=\log\left\{\frac{\Pr\left[d_n=0\left|\{r_l;|_{l=1}^{N}\},S\left(C(v_n)\setminus c_m\right)=1\right.\right]}{\Pr\left[d_n=1\left|\{r_l;|_{l=1}^{N}\},S\left(C(v_n)\setminus c_m\right)=1\right.\right]}\right\}$$

$$=L\left(d_n\left|\{r_l;|_{l=1}^{N}\},S\left(C(v_n)\setminus c_m\right)=1\right.\right) \tag{5-53}$$

where $L(x)$ denotes the log-likelihood ration (LLR) of a binary random variable x (see the Appendix B). Therefore, $Z_{m,n}$ can be expressed as a conditional LLR of d_n.

5.4.1 LLR-BP Algorithm Based on the tanh rule

Based on the standard BP algorithm, the LLR-BP algorithm is summarized as follows:

Step1. Initialization: Following the relations in (5-7), each variable node v_n is given an initial *a posterior* LLR F_n only based on the channel soft output r_n

$$F_n=L\left(d_n\left|r_n\right.\right)=\log\frac{\Pr\left(d_n=0\left|r_n\right.\right)}{\Pr\left(d_n=1\left|r_n\right.\right)}$$

$$=\log\frac{f_n^0}{f_n^1}=2ar_n/\sigma^2 \tag{5-54}$$

For each position (m,n) related to a nonzero entry in the sparse parity-check matrix of an LDPC code, $Z_{m,n}$ is given an initial value F_n before commencing the iterative decoding

$$Z_{m,n}=F_n \tag{5-55}$$

Step 2. Message from a check node c_m (Horizontal step): The calculation of $L_{m,n}$, which is the updated outgoing LLR extrinsic messages from a check node c_m to a related variable node $v_n\in V(c_m)$, is performed in this step. Note that the probability

$\Pr\left(S\left(\{c_m\}\right)=1\middle|d_n=i,\{r_l;|_{l=1}^{N}\}\right)$ in (5-52) is equivalent to the probability of an event such that $\sum_{v_{\underline{n}}\in V(c_m)\backslash v_n}\oplus d_{\underline{n}}=i$ (i=0, 1) based on the two conditions:

(A) The channel soft outputs $\{r_l;|_{l=1}^{N}\}$ are obtained by the decoder.

(B) The checks in each set $C(v_{\underline{n}})\backslash c_m$ corresponding to every $v_{\underline{n}}\in V(c_m)\backslash v_n$ are satisfied simultaneously.

Hence, $L_{m,n}$ can be expressed as

$$L_{m,n}=\log\left[\frac{\Pr\left(S(\{c_m\})=1\middle|d_n=0,\{r_l;|_{l=1}^{N}\}\right)}{\Pr\left(S(\{c_m\})=1\middle|d_n=1,\{r_l;|_{l=1}^{N}\}\right)}\right]$$

$$=\log\left(\frac{\Pr\left[\left(\sum_{v_{\underline{n}}\in V(c_m)\backslash v_n}\oplus d_{\underline{n}}=0\right)\middle|\{r_l;|_{l=1}^{N}\},\underset{v_{\underline{n}}\in V(c_m)\backslash v_n}{S}\left(C(v_{\underline{n}})\backslash c_m\right)=1\right]}{\Pr\left[\left(\sum_{v_{\underline{n}}\in V(c_m)\backslash v_n}\oplus d_{\underline{n}}=1\right)\middle|\{r_l;|_{l=1}^{N}\},\underset{v_{\underline{n}}\in V(c_m)\backslash v_n}{S}\left(C(v_{\underline{n}})\backslash c_m\right)=1\right]}\right)$$

$$=L\left(\left(\sum_{v_{\underline{n}}\in V(c_m)\backslash v_n}\oplus d_{\underline{n}}\right)\middle|\{r_l;|_{l=1}^{N}\},\underset{v_{\underline{n}}\in V(c_m)\backslash v_n}{S}\left(C(v_{\underline{n}})\backslash c_m\right)=1\right) \tag{5-56}$$

Theorem 5.1 Prove that

$$L\left(\sum_{j=1}^{J}\oplus u_j\right)=\log\frac{1+\prod_{j=1}^{J}\tanh\left(L(u_j)/2\right)}{1-\prod_{j=1}^{J}\tanh\left(L(u_j)/2\right)}=2\tanh^{-1}\left[\prod_{j=1}^{J}\tanh\left(L(u_j)/2\right)\right]$$

$$=2\prod_{j=1}^{J}sign\left(L\left(u_j\right)\right)\tanh^{-1}\prod_{j=1}^{J}\tanh\left[\left|L(u_j)\right|/2\right] \tag{5-57}$$

where u_j and $L(u_j)$ (j =1, 2, ……, J) are statistically independent binary variables in GF(2) and the corresponding LLRs, respectively, and the tanh function $y=\tanh(x/2)=(e^x-1)/(e^x+1)$ $(-\infty<x<+\infty)$ with its inverse function $x=2\tanh^{-1}(y)$ for $y\in(-1\ +1)$. □

The proof of the theorem can be referred to the Appendix B. Based on the theorem 5.1 and (5-53), we can further rewrite $L_{m,n}$ as

$$L_{m,n}=\left(\prod_{v_{\bar{n}}\in V(c_m)\backslash v_n} sign\left[L\left(d_{\bar{n}}\middle|\{r_l;|_{l=1}^{N}\},S\left(C(v_{\bar{n}})\backslash c_m\right)=1\right)\right]\right)$$

$$\times 2\tanh^{-1}\left(\prod_{v_{\bar{n}}\in V(c_m)\backslash v_n}\tanh\left[\frac{\left|L\left(d_{\bar{n}}\middle|\{r_l;|_{l=1}^{N}\},S\left(C(v_{\bar{n}})\backslash c_m\right)=1\right)\right|}{2}\right]\right)$$

$$=\left(\prod_{v_{\bar{n}}\in V(c_m)\backslash v_n} sign\left(Z_{m,\bar{n}}\right)\right)\times 2\tanh^{-1}\left[\prod_{v_{\bar{n}}\in V(c_m)\backslash v_n}\tanh\left(\frac{\left|Z_{m,\bar{n}}\right|}{2}\right)\right] \tag{5-58}$$

Clearly as in (5-58), it is the condition (B) that enables to establish the crucial relationship between the LLR messages in (5-52) and (5-53). The renewed $L_{m,n}$ will be used as outgoing LLR extrinsic messages from check node c_m to variable node $v_n\in V(c_m)$ in order to execute the processing in *step* 3. Thus, this step is also called *check-node update*.

Step 3. Message from a variable node v_n (Vertical step): The calculation of $Z_{m,n}$, which is the updated outgoing LLR extrinsic message from a variable node v_n to a related check node $c_m\in C(v_n)$, is carried out in this step. Based on (5-14), (5-15a) and (5-52), $Z_{m,n}$ can be further expressed as

$$Z_{m,n} = \log s_{m,n} = \log\left(f_n\right) + \log\left(\overline{\overline{R}}_{m,n}\right)$$

$$= L\left(d_n \middle| r_n\right) + \log\left(\frac{\prod_{\bar{c}_m \in C(v_n)\setminus c_m} \Pr\left(S\left(\{\bar{c}_m\}\right) = 1 \middle| d_n = 0, \{r_l;|_{l=1}^{N}\}\right)}{\prod_{\bar{c}_m \in C(v_n)\setminus c_m} \Pr\left(S\left(\{\bar{c}_m\}\right) = 1 \middle| d_n = 1, \{r_l;|_{l=1}^{N}\}\right)}\right)$$

$$= L\left(d_n \middle| r_n\right) + \sum_{\bar{c}_m \in C(v_n)\setminus c_m} \log\left(\frac{\Pr\left(S\left(\{\bar{c}_m\}\right) = 1 \middle| d_n = 0, \{r_l;|_{l=1}^{N}\}\right)}{\Pr\left(S\left(\{\bar{c}_m\}\right) = 1 \middle| d_n = 1, \{r_l;|_{l=1}^{N}\}\right)}\right)$$

$$= F_n + \sum_{\bar{c}_m \in C(v_n)\setminus c_m} L_{\bar{m},n} \tag{5-59}$$

The renewed $Z_{m,n}$ will be used as outgoing LLR extrinsic messages from variable node v_n to check node $c_m \in C(v_n)$ in order to perform the *step* 2 recursively. Thus, this step is also called *variable-node update*. Meanwhile, Z_n is evaluated for each codeword bits d_n (n=1, 2, ….., N) to make an estimation in the next step.

$$Z_n = F_n + \sum_{\bar{c}_m \in C(v_n)} L_{\bar{m},n} \tag{5.60}$$

Step 4. Final decision: Quantize $\tilde{\mathbf{d}} = [\tilde{d}_1, \tilde{d}_2, ..., \tilde{d}_N]^T$ such that $\tilde{d}_n$=0 if $Z_n > 0$, and $\tilde{d}_n$=1 if $Z_n \le 0$. If $\mathbf{H}\tilde{\mathbf{d}} = \mathbf{0}$, the iterative decoding process halts and $\tilde{\mathbf{d}}$ is treated as the decoder output; otherwise go back to *step* 2. If the algorithm does not halt within the maximum number of decoding iterations, a decoding failure is then declared. □

For LLR-BP algorithm, the multiplication operations are completely replaced by addition operations in *step* 3 due to the use of LLR messages, while they are still required in *step* 2. Hence, the BP algorithm is also called "sum-product" algorithm [7][42][46]. Note that we need to evaluate hyperbolic tangents in *step* 2. For a regular LDPC code

with d_v and d_c, the implementation of (5-58), regarding $L_{m,n}$ for all the variable nodes checked by c_m, i.e., $v_n \in V(c_m)$, requires d_c hyperbolic tangent evaluations, d_c multiplications to obtain the entire product $\prod_{v_n \in V(c_m)} \tanh\left(\left|Z_{m,n}\right|/2\right)$, d_c divisions and d_c inverse hyperbolic tangent evaluations to obtain $\left|L_{m,n}\right|$ for all $v_n \in V(c_m)$. The sign of $L_{m,n}$ is resulted in by first multiplying the signs of the incoming extrinsic LLR messages $Z_{m,n}$ for all $v_n \in V(c_m)$, and then multiplying the individual item $sign\left(Z_{m,n}\right)$ again.

5.4.2 LLR-BP Algorithm Based on Gallager's Approach

The LLR-BP based on Gallager's approach [1] is a modified version of the LLR-BP algorithm by using the tanh rule, where the outgoing LLR extrinsic messages in *step* 2 are computed based on the following theorem with a so-called involution transform [1].

Theorem 5.2 Prove that

$$L\left(\sum_{j=1}^{J} \oplus u_j\right) = \prod_{j=1}^{J} sign\left(L(u_j)\right) f\left[\sum_{j=1}^{J} f\left(\left|L(u_j)\right|\right)\right] \tag{5-61}$$

where u_j and $L(u_j)$ (j =1, 2, ….., J) are the same as those in the theorem 5.1, and $L\left(u_j\right) \neq 0$ for all j and $f(x) = \log\left[\left(e^x+1\right)/\left(e^x-1\right)\right]$ $(0 < x < +\infty)$ is an involution transform with the property $f\left[f(x)\right] = x$. □

The proof of the theorem can be referred to the Appendix B. Using the theorem 5.2, (5-58) can be formulated in a more simple form as

$$L_{m,n} = \left(\prod_{v_{\bar{n}} \in V(c_m)\backslash v_n} sign\left(Z_{m,\bar{n}}\right)\right) \times f\left(\sum_{v_{\bar{n}} \in V(c_m)\backslash v_n} f\left(\left|Z_{m,\bar{n}}\right|\right)\right) \tag{5-62}$$

The involution transform is first done on all incoming LLR messages Z_{m,n_1} for all the variable nodes checked by c_m, i.e., all $v_{n_1} \in V(c_m)$. Then the individual term $f\left(\left|Z_{m,n}\right|\right)$ is subtracted from $\sum_{v_{n_1} \in V(c_m)} f\left(\left|Z_{m,n_1}\right|\right)$, and the resulted one is used as the input for another transform with the overall sign to generate the outgoing LLR extrinsic message.

Clearly, the calculation of $L_{m,n}$, formulated by Gallager's approach, for all the variable nodes checked by c_m requires $2\,d_c$ evaluations of the involution transform and $2\,d_c$ additions, which is relatively simple and amenable to a parallel implementation, and therefore promising for use in many extremely high-speed applications.

5.4.3 LLR-BP Algorithm Based on Jacobian Approach

The LLR-BP based on Jacobian approach [51] is another modified version of the LLR-BP algorithm based on the following theorem 5.3 concerning a very useful identity, where the outgoing LLR extrinsic messages in *step* 2 are computed in a recursive manner.

Theorem 5.3 Prove that

$$L(u_1 \oplus u_2) = sign(L(u_1))\ sign(L(u_2))\ \min\left(\left|L(u_1)\right|, \left|L(u_2)\right|\right) + \Delta \tag{5-63a}$$

and Δ is

$$\Delta = \log\left(1 + e^{-|L(u_1)+L(u_2)|}\right) - \log\left(1 + e^{-|L(u_1)-L(u_2)|}\right) \tag{5-63b}$$

where u_j and $L(u_j)$ (j=1, 2) are the same as those in the theorem 5.1. □

The proof of the theorem can be referred to the Appendix B. Consider check node c_m with d_c edges from variable nodes in $V(c_m) = \{v_n;\ n = n_1,\ n_2,\ \ldots..,\ n_{d_c}\}$. The incoming LLR

messages are Z_{m,n_1}, Z_{m,n_2}, ….., $Z_{m,n_{d_c}}$. Let us first define a set of auxiliary binary random variables in a *forward* manner as.

$$f_1 = d_{n_1} \tag{5-64a}$$

$$f_2 = f_1 \oplus d_{n_2} \tag{5-64b}$$

$$f_3 = f_2 \oplus d_{n_3} \tag{5-64c}$$

$$\vdots$$

$$f_{d_c-1} = f_{d_c-2} \oplus d_{n_{d_c-1}} \tag{5-64d}$$

$$f_{d_c} = f_{d_c-1} \oplus d_{n_{d_c}} \tag{5-64e}$$

Similarly for a *backward* manner, we have another set of auxiliary binary random variables

$$g_{d_c} = d_{n_{d_c}} \tag{5-65a}$$

$$g_{d_c-1} = g_{d_c} \oplus d_{n_{d_c-1}} \tag{5-65b}$$

$$g_{d_c-2} = g_{d_c-1} \oplus d_{n_{d_c-2}} \tag{5-65c}$$

$$\vdots$$

$$g_2 = g_3 \oplus d_{n_2} \tag{5-65d}$$

$$\vdots$$

$$g_1 = g_2 \oplus d_{n_1} \tag{5-65e}$$

Using the theorem 5.3 repeatedly, we obtain $L(f_1)$, $L(f_2)$, ……, $L(f_{d_c-1})$ and $L(g_{d_c})$, $L(g_{d_c-1})$, ……, $L(g_2)$ in a *recursive* manner from Z_{m,n_1}, Z_{m,n_1}, ..., $Z_{m,n_{d_c}}$ that are evaluated by variable-node update of *step* 3 in the preceding iterative round. It immediately results in the following relation due to the constraint of zero check-sum, i.e., $\sum_{i=1}^{d_c} \oplus d_{n_i} = 0$.

$$d_{n_i} = f_{i-1} \oplus g_{i+1} \qquad (i=2, 3, ..., d_c-1) \tag{5-66}$$

Therefore, the outgoing LLR extrinsic message from check node c_m becomes

$$L_{m,n_i} = L\left(\sum_{v_{\underline{n}} \in V(c_m)\backslash v_{n_i}} \oplus d_{\underline{n}} \middle| \{r_l;|_{l=1}^{N}\}, \underset{v_{\underline{n}} \in V(c_m)\backslash v_{n_i}}{S} \left(C(v_{\underline{n}})\backslash c_m\right)=1 \right)$$

$$= L\left(f_{i-1} \oplus g_{i+1} \middle| \{r_l;|_{l=1}^{N}\}, \underset{v_{\underline{n}} \in V(c_m)\backslash v_{n_i}}{S} \left(C(v_{\underline{n}})\backslash c_m\right)=1 \right) \quad (i=2, 3, ..., d_c-1) \tag{5-67a}$$

and

$$L_{m,n_i} = \begin{cases} L\left(g_2 \middle| \{r_l;|_{l=1}^{N}\}, \underset{v_{\underline{n}} \in V(c_m)\backslash v_{n_1}}{S} \left(C(v_{\underline{n}})\backslash c_m\right)=1 \right) & if \quad i=1 \\ L\left(f_{d_c-1} \middle| \{r_l;|_{l=1}^{N}\}, \underset{v_{\underline{n}} \in V(c_m)\backslash v_{n_{d_c}}}{S} \left(C(v_{\underline{n}})\backslash c_m\right)=1 \right) & if \quad i=d_c \end{cases} \tag{5-67b}$$

Thus, the resulted LLR message $L_{m,n}$ is sent to the variable nodes checked by c_m. This is essentially the forward-backward algorithm, which is quite similar to the BCJR (or MAP) [12] algorithm applied to decode the component recursive systematic convolutional (RSC) codes of a turbo code.

Actually, the calculation of $L_{m,n}$ for all the variable nodes checked by c_m in *step* 2 requires altogether 3(d_c −2) computations of the core operation $L(u_1 \oplus u_2)$. More specifically, for (5-64), (5-65) and (5-67a), each needs d_c −2 such operations. Note that $L(f_1)=Z_{m,n_1}$ and $L(g_{d_c})=Z_{m,n_{d_c}}$ while $L(f_{d_c})$ and $L(g_1)$ are unnecessary for (5-67a) and (5-67b). The underlying function $h(x) = \log\left(1+e^{-|x|}\right)$ can be adopted as a lookup table [42][46]. Therefore, the core operation can be realized by using four additions, one comparison and two corrections, each of which can be implemented by a table lookup.

5.4.4 Computational Complexity for LLR-BP Algorithms Based on Different Approaches

For LLR-BP algorithm based on tanh rule, Gallager's approach and Jacobian Approach, the *variable-node* updates in *step* 3 are completely the same. Hence, we only need to compare the *check-node* updates for the three cases. Table 5.2 [46] summarizes the computational complexity involving the two operations of multiplication (division) and addition (subtraction), and some special operations, such as evolution function $f(x)$ and $L(U+V)$.

Algorithm	Multiplications	Additions	Special Operations
LLR-BP (tanh rule)	$2d_c$	None	$d_c \tanh(x)$, $d_c \tanh^{-1}(x)$
LLR-BP (Gallager)	None	$2d_c$	$2d_c f(x)$
$L(U \oplus V)$ LLR-BP (Jacobian)	None None	5 None	2 table loo-ups 3(d_c-2) $L(U \oplus V)$

Table 5.2 Computational complexity of check-node updates for LLR-BP decoding algorithm of various representations.

Apparently, the check-node updates by Gallager's approach and Jacobian method are less computationally intensive than LLR-BP by tanh rule since the evolution function $f(x)$ and $L(U \oplus V)$ in the two modified versions can be effectively implemented either by a systematic way using its elegant properties or by simple table look-ups. Both methods

avoid the high complexity evaluations of $\tanh(x)$ and $\tanh^{-1}(x)$ used by LLR-BP by tanh rule.

5.5 Probability Density Evolutions of Belief Propagation Decoding Algorithms with Continuous Message Alphabets

Recall that in Section 5.3 we have derived the probability density function for Gallager's decoding algorithm A and B and the iterative decoding algorithm with erasures inside the decoder over BSC. In this section, we will investigate the most general and important case of a message-passing decoder with an *infinite* message alphabet for *regular* LDPC codes, i.e., the belief-propagation or sum-product decoder that employs LLR-BP decoding algorithm introduced in last section.

Similar as the scenarios for Gallager's decoding algorithm A and B, the derivation of density evolution function for the belief-propagation decoder with an *infinite* message alphabet can be generally summarized as two basic steps, corresponding to the check-node update and variable-node update, respectively.

Step A for check-node update: Derive the probability distribution of outgoing LLR extrinsic message from a check node based on the incoming LLR extrinsic messages generated by the incident variable nodes in the last decoding round, whose incoming messages are independent and identically distributed (i.i.d). The outgoing LLR message with the known distribution is sent for variable-node update in the current decoding round.

Step B for variable-node update: Derive the probability distribution of outgoing LLR extrinsic message from a variable node based on the i.i.d incoming LLR extrinsic

messages given by the incident check nodes in the current decoding round. The outgoing LLR message with the known distribution is sent for check-node update in the next decoding round.

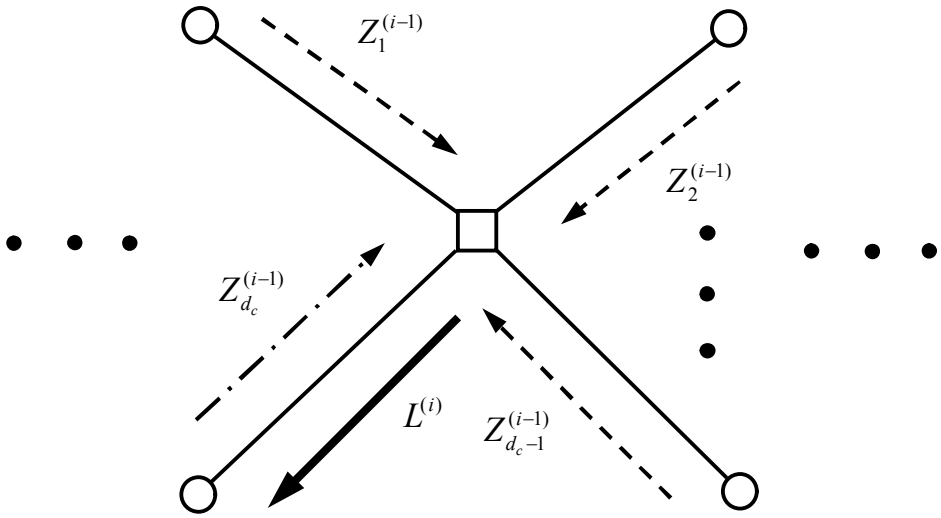

Fig. 5.11 The illustrative diagram of a Tanner graph for an outgoing LLR extrinsic message $L^{(i)}$ from a check node to a related variable node with $d_c - 1$ input LLR messages $Z_j^{(i-1)}$ (j=1, 2, …., $d_c - 1$).

For a *regular* LDPC code with check node of degree d_c in its Tanner graph, each outgoing LLR extrinsic message $L^{(i)}$ from a check node to a corresponding variable node in the i th iteration can be generally expressed by a function of $d_c - 1$ incoming LLR extrinsic messages $Z_1^{(i-1)}$, $Z_2^{(i-1)}$, ….., $Z_{d_c-1}^{(i-1)}$ generated by the related variable nodes in the $(i-1)$ th iteration, but one incoming LLR message $Z_{d_c}^{(i-1)}$ along the same edge as the outgoing message is excluded. The illustrative diagram is shown in Fig. 5.11, where the check node and the related variable nodes are represented by a square and circles, respectively, and the dashed lines denote $d_c - 1$ input LLR messages from the variable

nodes, and one long dash-dot line denotes the excluded input LLR message along the same edge as the outgoing LLR message $L^{(i)}$. Without danger of confusions, we will henceforth skip the indices for the incoming and outgoing LLR messages. Thus, the check-node update is

$$L = \Phi_c\left(Z_1, Z_2, ..., Z_{d_c-1}\right) \tag{5-68}$$

where $\Phi_c(.)$ depends on the decoding algorithms employed. In this section it refers to the check-node update using (5-58) for the LLR-BP algorithm.

Similarly, each outgoing LLR extrinsic message Z from a variable node to a related check node in the ith iteration is the sum of the initial *a posterior* LLR F and $d_v - 1$ incoming LLR extrinsic messages L_1, L_2, ….., L_{d_v-1} from the check nodes, each of which is evaluated by (5-68) in the ith iteration. One incoming LLR message Z_{d_c} along the same edge as the outgoing message is also excluded. The illustrative diagram is shown in Fig.5.12, where the dashed lines denote the *a posterior* LLR F and $d_v - 1$ input LLR messages from the check nodes, and one long dash-dot line denotes the excluded input LLR message along the same edge as the outgoing LLR message Z. Thus, we have function

$$Z = \Phi_v\left(F, L_1, L_2, ..., L_{d_v-1}\right) = F + \sum_{j=1}^{d_v-1} L_j \tag{5-69}$$

where the initial *a posterior* LLR F only depends on the channel soft output, and $\Phi_v(.)$ is simply a function of summation for the LLR-BP algorithm.

In order to efficiently analyze the density evolution for the BP-based algorithms, the following assumptions are necessarily made and listed as follows:

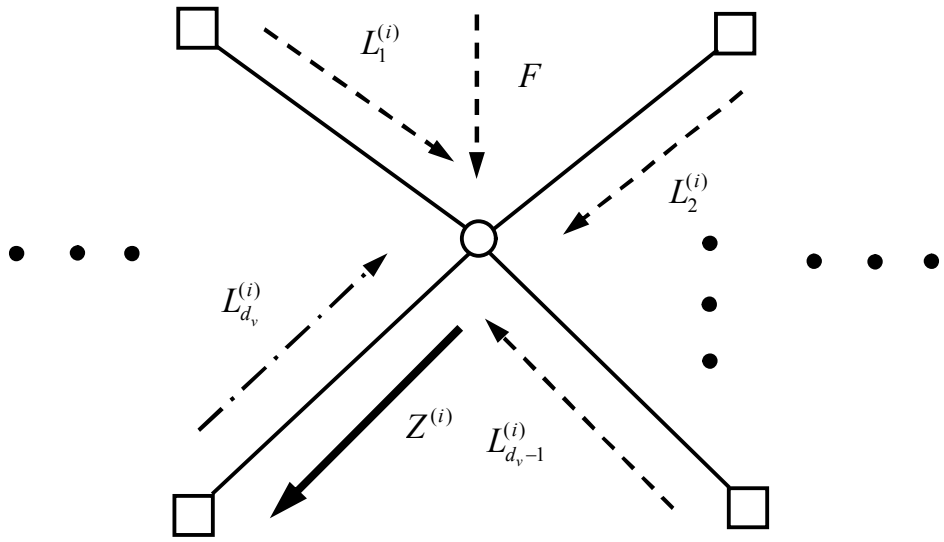

Fig. 5.12 The illustrative diagram of a Tanner graph for an outgoing LLR extrinsic message $Z^{(i)}$ from a variable node to a related check node with $d_v - 1$ input LLR messages $L_j^{(i)}$ (j =1, 2, ….., $d_v - 1$) and an initial *a posterior* LLR F .

(i) All-one binary symbols transmitted over the noise channel, or equivalently all-zero information bits stream for the encoder input.

(ii) The Tanner graph of a regular LDPC code is cycle-free.

(iii) The decoding of long code exhibits a threshold phenomenon that clearly separates the region where reliable transmission is possible from where it is not.

Consequently, all messages entering one node, regardless of check or variable nodes, are independent and identically distributed (i.i.d) based on the above assumptions. The key problem is to calculate the probability density function (pdf) of the messages of the

current iteration based on the pdf of the messages from the preceding processing. More specifically, for check-node update, given the pdf $P_Z(z_i)$ of Z_i (i=1, 2, ….., d_c −1) evaluated in the last round iteration and the mapping function $\Phi_c(.)$, we need to derive the pdf $P_L(l)$ of L for the current round iteration, which is stated in *step* A. For variable-node update, given the pdf $P_F(f)$ of F and the updated $P_L(l_j)$ of L_j (j=1, 2, ….., d_v −1), we need to evaluate the pdf $P_Z(z)$ for the next round iteration according to the mapping function $\Phi_v(.)$, which is stated in *step* B.

(1) Calculation of the pdf $P_L(l)$ of L in check-node update

Based on (5-58) for LLR-BP algorithm, the mapping function in (12.99) for a check node is

$$L=\log\left(\frac{1+\prod_{i=1}^{d_c-1}\tanh(Z_i/2)}{1-\prod_{i=1}^{d_c-1}\tanh(Z_i/2)}\right) \tag{5-70}$$

where Z_i (i=1, 2, ….., d_c−1) are independent and identically distributed LLRs, whose probability density function (pdf) is determined by the variable-node update in the last decoding round and denoted by $P_Z(z_i)$. Let the variables Y_i, Ω and L be defined as follows with the probability density functions $P_Y(y_i)$, $P_X(x)$ and $P_L(l)$, respectively.

$$Y_i=\tanh(Z_i/2)=\frac{e^{Z_i}-1}{e^{Z_i}+1} \tag{5-71a}$$

$$X=\prod_{i=1}^{d_c-1}Y_i \tag{5-71b}$$

$$L=\log\left(\frac{1+X}{1-X}\right) \tag{5-71c}$$

Hence, based on the known $P_Z(z_i)$ we first determine the pdf $P_Y(y_i)$ of Y_i by making an appropriate variable transformation that expresses Z_i in terms of Y_i, then specify the pdf $P_X(x)$ of X, and finally obtain the pdf $P_L(l)$ of L.

(a) Determine the pdf $P_Y(y_i)$ based on pdf $P_Z(z_i)$: From (5-71a) it yields

$$z_i=\log\left(\frac{1+y_i}{1-y_i}\right) \qquad (-\infty<z_i<+\infty,\ -1<y_i<1) \tag{5-72a}$$

Thus, we have

$$P_Y(y_i)=P_Z(z_i)\left|\frac{dz_i}{dy_i}\right|=\frac{2}{1-y_i^2}\,P_Z\left(\log\left(\frac{1+y_i}{1-y_i}\right)\right) \tag{5-72b}$$

Note that for the first decoding iteration in an AWGN channel, the LLR Z_i can be directly evaluated based on the channel soft output as specified by (5-54) and (5-55), which is distributed by $Z_i \sim N(\mu_Z,\ \sigma_Z^2)$ with $\mu_Z=2a^2/\sigma^2$ and $\sigma_Z^2=4a^2/\sigma^2$ if the binary symbols all a's are transmitted over the channel. Thus, pdf of Y_i is

$$\begin{aligned} P_Y(y_i)&=\frac{2}{1-y_i^2}\,\frac{1}{\sqrt{2\pi}\sigma_Z}\exp\left\{-\frac{(z_i-\mu_Z)^2}{2\sigma_Z^2}\right\} \\ &=\frac{\sigma}{\sqrt{2\pi}a(1-y_i^2)}\exp\left\{-\frac{\left[\log[(1+y_i)/(1-y_i)]-2(a^2/\sigma^2)\right]^2}{8(a^2/\sigma^2)}\right\} \end{aligned} \tag{5-73}$$

(b) Find the pdf $P_X(x_i)$ based on pdf $P_Y(y_i)$: It immediately follows from (5-71b) that

$$y_{d_c-1}=\frac{x}{\prod_{i=1}^{d_c-2}y_i} \tag{5-74}$$

Hence, the joint probability density of X, Y_1, Y_2, …., Y_{d_c-2} is given by

$$P_{X,\mathbf{Y}}\left(x,y_1,y_2,....,y_{d_c-2}\right)=P_{\mathbf{Y}}\left(y_1,y_2,.....,y_{d_c-1}\right)\left|\frac{\partial y_{d_c-1}}{\partial x}\right|$$

$$=\frac{1}{\left|\prod_{i=1}^{d_c-2}y_i\right|}\underbrace{P_Y\left(y_1\right)P_Y\left(y_2\right).....P_Y\left(y_{d_c-2}\right)P_Y\left(y_{d_c-1}\right)}_{d_c-1}$$

$$=\frac{1}{\left|\prod_{i=1}^{d_c-2}y_i\right|}\underbrace{P_Y\left(y_1\right)P_Y\left(y_2\right).....P_Y\left(y_{d_c-2}\right)}_{d_c-2}P_Y\left(\frac{x}{\prod_{i=1}^{d_c-2}y_i}\right) \tag{5-75}$$

where $P_{\mathbf{Y}}\left(y_1,y_2,.....,y_{d_c-1}\right)$ is the joint pdf of the independent and identically distributed random variables Y_1, Y_2, …., Y_{d_c-1}. Hence, we can integrate out d_c–2 variables Y_1, Y_2, …., Y_{d_c-2} to get the marginal density of X

$$P_X\left(x\right)=\underbrace{\int\int.....\int}_{y_1\,y_2\qquad y_{d_c-2}}P_{X,\mathbf{Y}}\left(x,y_1,y_2,.....,y_{d_c-2}\right)dy_1dy_2\,.....\,dy_{d_c-2}$$

$$=\underbrace{\int_{-1}^{1}\int_{-1}^{1}.....\int_{-1}^{1}}_{d_c-2}\frac{1}{\left|\prod_{i=1}^{d_c-2}y_i\right|}\underbrace{P_Y\left(y_1\right)P_Y\left(y_2\right).....P_Y\left(y_{d_c-2}\right)}_{d_c-2}\times$$

$$P_Y\left(\frac{x}{\prod_{i=1}^{d_c-2}y_i}\right)dy_1dy_2\,.....\,dy_{d_c-2}\qquad(-1<x<1) \tag{5-76}$$

The above expression mathematically indicates a way to calculate the pdf of random variable X in theory, which can be efficiently computed by numerical methods thanks to d_c usually as a small number.

(c) Specify the pdf $P_L(l)$ based on pdf $P_X(x)$: From (5-71c) we find a very interesting result in contrast to (5-71a), which is

$$x=\frac{e^l-1}{e^l+1}=\tanh\left(l/2\right)\qquad(-\infty<l<+\infty\,,\,-1<x<1) \tag{5-77}$$

Hence, $P_L(l)$ is easily derived as follows

$$P_L(l)=P_X(x)\left|\frac{dx}{dl}\right|=\frac{2e^l}{\left(1+e^l\right)^2}\,P_X\left(\tanh(l/2)\right) \tag{5-78}$$

Note that another effective way was proposed in [8][9] where the evaluation of $P_L(l)$ is realized by using Fourier transform and inverse Fourier transform over the direct product of two fields. The $P_L(l)$ is then used for the next step in the evaluation of the pdf $P_Z(z)$ in the current decoding iteration.

(2) Calculation of the pdf $P_Z(z)$ of Z in variable-node update

Based on (5-59) for LLR-BP algorithm, the mapping function in (5-69) for a variable node is

$$Z=F+\sum_{j=1}^{d_v-1}L_j \tag{5-79}$$

where L_i (i=1, 2, …., d_v–1) are independent and identically distributed LLRs, whose probability density function $P_L(l_i)$ is given by (5-78) in the check-node update in the current decoding round, and F is the *a posterior* LLR initially from channel soft output. Obviously, the probability density $P_Z(z)$ of Z, which is the sum of the independently distributed random variables, is simply the convolution of the densities $P_F(f)$ of F and $P_L(l_i)$ of L_i, i.e.,

$$P_Z(z)=P_F(f)*\underbrace{P_L(l_1)*P_L(l_2)*.....*P_L(l_{d_v-1})}_{d_v-1} \tag{5-80}$$

where the symbol "$*$" denotes convolution. Alternatively, $P_Z(z)$ can be formulated by the inverse Fourier transform (F^{-1}) of the multiplication of the characteristic functions of variables F and L_i (i=1, 2, …., d_v−1)

$$P_Z(z)=\mathsf{F}^{-1}\{\varphi_F(jv)[\varphi_L(jv)]^{d_v-1}\}$$

$$=\frac{1}{2\pi}\int_{-\infty}^{+\infty}\varphi_F(jv)[\varphi_L(jv)]^{d_v-1}\exp(-jvz)dv \tag{5-81}$$

where the variable v is real and $j=\sqrt{-1}$, and the two characteristic functions $\varphi_F(jv)$ and $\varphi_L(jv)$ are the Fourier transform (F) of the pdf's $P_F(f)$ and $P_L(l_i)$, respectively

$$\varphi_F(jv)=\mathsf{F}\{P_F(f)\}=\int_{-\infty}^{+\infty}P_F(f)\exp(jvf)df \tag{5-82a}$$

$$\varphi_L(jv)=\mathsf{F}\{P_L(l_i)\}=\int_{-\infty}^{+\infty}P_L(l_i)\exp(jvl_i)dl_i \tag{5-82b}$$

Clearly, $P_Z(z)$ in (5-81) can be efficiently implemented by the well-known fast Fourier transform (FFT) and inverse FFT techniques. In particular, for AWGN channel F is a Gaussian variable distributed by $N(\mu_F, \sigma_F^2)$ ($\mu_F=2a^2/\sigma^2$, $\sigma_F^2=4a^2/\sigma^2$) if all-one binary symbols are transmitted, i.e.,

$$P_F(f)=\frac{1}{\sqrt{2\pi}\sigma_F}\exp\left[-\frac{(f-\mu_F)^2}{2\sigma_F^2}\right] \tag{5-83a}$$

Accordingly, its characteristic function is

$$\varphi_F(jv)=\frac{1}{\sqrt{2\pi}\sigma_F}\int_{-\infty}^{+\infty}\exp\left[-\frac{(f-\mu_F)^2}{2\sigma_F^2}\right]\exp(jvf)df$$

$$=\exp(j\mu_F v-(1/2)\sigma_F^2v^2)=\exp\left[\frac{2a^2}{\sigma^2}(jv-v^2)\right] \tag{5-83b}$$

Finally, the obtained $P_Z(z)$ is offered to the check-node update to evaluate the pdf $P_L(l)$ in the next decoding round. The bit error rate after the k th decoding iteration, conditioned on all-one binary symbols transmitted over the noisy channel, is

$$p_e^{(k)} = 1 - \int_{-\infty}^{0} P_Z(z) dz \tag{5-84}$$

Now we can briefly summarize the main steps to evaluate the density evolution for BP algorithm with continuous message alphabets for (d_v, d_c) regular LDPC codes.

(i) In check-node update: Computing the updated pdf $P_L(l)$ based on the pdf's $P_Z(z_i)$ (i= 1, 2, ….., d_c–1) evaluated in the preceding iteration and the variable transformations given by (5-71a), (5-71b) and (5-71c). In the first iteration, the distribution of LLR Z_i only depends on the channel noise. In particular, Z_i is a Gaussian variable and distributed by $Z_i \sim N(\mu_Z, \sigma_Z^2)$ with $\mu_Z = 2a^2/\sigma^2$ and $\sigma_Z^2 = 4a^2/\sigma^2$ if all-one binary symbols are transmitted over an AWGN channel. □

(ii) In variable-node update: Computing the updated pdf $P_Z(z)$ based on the pdf's $P_L(l_i)$ (i=1, 2, ….., d_v–1) evaluated by the check-node update in the current iteration and the pdf $P_F(f)$ of the LLR with respective to the received symbol. $P_Z(z)$ is simply the convolution of $P_L(l_i)$ and $P_F(f)$, which is numerically implemented by FFT and inverse FFT techniques. In particular, for AWGN channel F is a Gaussian variable and distributed by $Z_i \sim N(\mu_F, \sigma_F^2)$ with $\mu_F = 2a^2/\sigma^2$ and $\sigma_F^2 = 4a^2/\sigma^2$ if all-one binary symbols are sent over the channel. □

5.6 Thresholds of Belief Propagation Decoding Algorithms with Continuous Message Alphabets over Memoryless Binary-Input Continuous-Output Channels

Similar as the thresholds for regular LDPC codes using Gallager's decoding algorithm A and B and iterative decoding algorithm with erasures in the decoder over BSC, the thresholds for regular LDPC codes employing BP algorithm with continuous message alphabets also exist for different noisy channels. In this section we will study two typical memoryless binary-input continuous-output channels, i.e., additive white Gaussian noise (AWGN) and Laplace channels, whose output alphabet is equal to the message alphabet in the BP decoder. These two channels belong to the memoryless binary-input output-symmetric (BIOS) [62] category. The thresholds obtained by numerical methods are also compared with the corresponding maximum allowed values related to the capacities of the channels.

5.6.1 Binary-input Additive white Gaussian noise (BIAWGN) channel:

Consider a memoryless binary-input continuous-output additive Gaussian noise channel modeled by $r_n = s_n + w_n$ as (5-3), where the n th transmitted s_n is either +1 or –1 corresponding to binary information bit 0 and 1, respectively, w_n is a Gaussian variable with zero mean and standard deviation σ, i.e.,

$$p_{BIG}(w) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{w^2}{2\sigma^2} \right) \tag{5-85}$$

The capacity of BIAWGN channel is a function of standard deviation σ and given by

$$C_{BIG}(\sigma)=-\int_{-\infty}^{+\infty}\varsigma_\sigma(r)\log_2[\varsigma_\sigma(r)]dr-\frac{1}{2}\log_2\left(2\pi e\sigma^2\right) \tag{5-86}$$

where the function $\varsigma_\sigma(r)$ is

$$\varsigma_\sigma(r)=\frac{1}{\sqrt{8\pi}\sigma}\left[\exp\left(-\frac{(r-1)^2}{2\sigma^2}\right)+\exp\left(-\frac{(r+1)^2}{2\sigma^2}\right)\right] \tag{5-87}$$

d_v	d_c	Rate	σ^*	σ_{opt}
3	6	0.5	0.88	0.9787
4	8	0.5	0.83	0.9787
5	10	0.5	0.79	0.9787
3	5	0.4	1.0	1.148
4	6	0.333	1.01	1.295
4	4	0.25	1.26	1.549

Table 5.3 Threshold σ^* for various regular LDPC codes using BP algorithm with continuous message alphabets over BIAWGN channel and the maximum allowed threshold value σ_{opt}.

The derivation of the BIAWGN channel capacity can be referred to Appendix G. Table 5.3 lists the thresholds in terms of standard deviation for various regular LDPC codes by numerical simulations using the BP algorithm with continuous message alphabets, introduced in last section, over the BIAWGN channel. Let σ^* denote the supremum of all values of $\sigma \in (0, +\infty)$ such that $p_e^{(k)}(\sigma)$, evaluated by (5-84) as a function of σ in

BIAWGN channel, satisfied that $\lim_{k\to+\infty} p_e^{(k)}(\sigma)=0$ if $\sigma<\sigma^*$ and the number of decoding iterations asymptotically goes to infinity. The maximum allowed value σ_{opt} given in this Table is associated with the capacity of BIAWGN in (5-86), which is the same as the code rate of the individual regular LDPC code. The most impressive one of all rate-1/2 codes is the (3, 6) LDPC code with the threshold σ^* equal to 0.88 and close to the related optimum value σ_{opt}=0.9787 resulted in by letting C_{BIG}=1/2 in (5-86).

(B) Binary-input Laplace (BIL) channel: Consider another memoryless binary-input channel with continuous output and additive Laplacian noise, which is modeled by $r_n=s_n+w_n$ with w_n distributed by

$$p_{BIL}(w)=\frac{1}{2\lambda}\exp\left(-\frac{|w|}{\lambda}\right) \tag{5-88}$$

The capacity of the BIL channel is a function of parameter λ and given by

$$C_{BIL}(\lambda)=1-\log_2\left[1+\exp(-2/\lambda)\right]-\frac{\arctan\left[sh(1/\lambda)\right]}{\exp(1/\lambda)\ln 2} \tag{5-89}$$

The derivation of the BIL channel capacity can be referred to Appendix G. Table 5.4 gives the thresholds in terms of parameter λ for the same regular LDPC codes as Table 5.3 by numerical simulations using the BP algorithm with continuous message alphabets over the BIL channel. Similarly, let λ^* denote the supremum of all values of $\lambda\in(0,+\infty)$ such that $p_e^{(k)}(\lambda)$ satisfied that $\lim_{k\to+\infty} p_e^{(k)}(\lambda)=0$ if $\lambda<\lambda^*$ and the infinite number of decoding iterations. The optimum value σ_{opt} listed in this Table is given by letting the ultimate capacity C_{BIL} in (5-89) be the code rate of the respective regular LDPC code.

The best one of all rate-1/2 codes is still the (3, 6) LDPC code with threshold λ^* equal to 0.65 and close to the respective optimum value λ_{opt}=0.752 obtained by setting C_{BIL}=1/2 in (5-89).

d_v	d_c	Rate	λ^*	λ_{opt}
3	6	0.5	0.65	0.752
4	8	0.5	0.62	0.752
5	10	0.5	0.58	0.752
3	5	0.4	0.77	0.914
4	6	0.333	0.78	1.055
4	4	0.25	1.02	1.298

Table 5.4 Threshold λ^* for various regular LDPC codes using BP algorithm with continuous message alphabets over BIL channel and the maximum allowed threshold value λ_{opt}.

Chapter 6 Reduced-Complexity Belief-Propagation-Based Iterative Decoding Algorithms for LDPC Codes

This section focuses on simplifying the check-node updates (in *step* 2) of LLR-BP algorithm to obtain reduced-complexity LLR-BP derivatives [42][46][49-52] that can achieve near-optimum performance. Meanwhile, a simplified variable-node update of LLR-BP algorithm is also very useful and important for some LDPC codes in practical applications [57[58], where the storage requirements [55][56] can be significantly reduced. The following approximated approaches are introduced in this section, which are all BP-based algorithms using LLR extrinsic messages for check and variable-node updates and thus called BP-based approximations thereafter. Finally, the comparisons of computational complexity are made for the three BP-Based approximations regarding the check-node updates.

6.1.1 BP-Based Approximation

The most critical method to obtain the approximated BP-based algorithms from the accurate LLR-BP algorithm is to simplify its check-node updates in *step* 2. Let us consider an inequality described by the following theorem.

Theorem 6.1 The magnitude of $L\left(\sum_{j=1}^{J}\oplus u_j\right)$ is not greater than the minimum of the magnitudes of $L(u_1)$, $L(u_2)$, ….., $L(u_J)$, namely,

$$\left|L(\sum_{j=1}^{J}\oplus u_j)\right| \le \min\left\{\left|L(u_1)\right|,\left|L(u_2)\right|,.....,\left|L(u_J)\right|\right\} \tag{6-1}$$

where u_j and $L(u_j)$ (j=1, 2, ….., J) are the same as those in the theorem 5.1. □

The proof of the theorem can be referred to the Appendix B. By applying the inequality in (6-1) to (5-58), $L_{m,n}$ can be approximately evaluated as $\tilde{L}_{m,n}$

$$L_{m,n} \approx \tilde{L}_{m,n} = \left(\prod_{v_{\bar{n}} \in V(c_m) \setminus v_n} sign\left(Z_{m,\bar{n}}\right) \right) \times \min_{v_{\bar{n}} \in V(c_m) \setminus v_n} \left| Z_{m,\bar{n}} \right| \tag{6-2}$$

In practice, it requires the determination of two of the incoming LLR extrinsic messages with the smallest magnitudes, as well as of the signs of the outgoing LLR extrinsic messages. More specifically, only two other than d_c resulting values (Z_{m,n_1}, Z_{m,n_2}, ….., $Z_{m,n_{d_c}}$) have to be stored in the decoder since $d_c - 1$ branches share the same magnitude of the outgoing LLR extrinsic message.

In order to make this simplification in a clearer manner, we may investigate the following instance. For instance, the magnitudes of the incoming LLR messages are $\left|Z_{m,n_1}\right|$, $\left|Z_{m,n_2}\right|$, ……, $\left|Z_{m,n_{d_c-1}}\right|$, $\left|Z_{m,n_{d_c}}\right|$ for a regular LDPC code with d_v and d_c. Without loss of generality, suppose two of the smallest ones be $\left|Z_{m,n_1}\right|$ and $\left|Z_{m,n_2}\right|$ with $\left|Z_{m,n_1}\right| > \left|Z_{m,n_2}\right|$. According to (6-2), it yields

$$\left|\tilde{L}_{m,n_2}\right| = \left|Z_{m,n_1}\right| \tag{6-3a}$$

and

$$\left|\tilde{L}_{m,n_i}\right| = \left|Z_{m,n_2}\right| \quad (i = 1, 3, \ldots..., d_c) \tag{6-3b}$$

Hence, BP-based approximation not only results in the reduction of computational complexity, but also in the efficient memory storage. The overall computational

complexity of this step is $d_c+[\log d_c]-2$ additions (comparisons) [42]. In particular, this approximation by the theorem 6.1 immediately results in the following BP-based approximated method to get more accurate soft LLRs generated from check nodes.

6.1.2 Normalized BP-Based Approximation

We have established from the theorem 6.1 that $L_{m,n}$ and $\tilde{L}_{m,n}$ have the same sign and magnitude of $\tilde{L}_{m,n}$ is not less than that of $L_{m,n}$, i.e., sign($L_{m,n}$)=sign($\tilde{L}_{m,n}$) and $|\tilde{L}_{m,n}|\geq|L_{m,n}|$. This conclude motivates us to adopt a normalization method to get more accurate soft LLR values from $\tilde{L}_{m,n}$. In other words, $\tilde{L}_{m,n}$ can be divided by a factor α that is usually greater than one to achieve a better approximation of $L_{m,n}$. In a statistical view, to determine α, we can apply a reasonable criterion of forcing the mean of the normalized magnitude $\left|\tilde{L}_{m,n}\right|/\alpha$ to be equal to the mean of the magnitude $L_{m,n}$, i.e.,

$$\alpha=\frac{E\left(\left|\tilde{L}_{m,n}\right|\right)}{E\left(\left|L_{m,n}\right|\right)} \tag{6-4}$$

The normalization factor α that makes $\tilde{L}_{m,n}/\alpha$ equal to $L_{m,n}$ in magnitude on the probabilistic sense may not be the optimum one, but it seems to be a very reasonable and practical approach. Thus, the BP-based approximation can be further improved by employing a check-node update that employs the above-mentioned α, and is given by

$$\hat{L}_{m,n}=\left(\prod_{v_{\bar{n}}\in V(c_m)\backslash v_n} sign\left(Z_{m,\bar{n}}\right)\right)\times\frac{\min\limits_{v_{\bar{n}}\in V(c_m)\backslash v_n}\left|Z_{m,\bar{n}}\right|}{\alpha} \tag{6-5}$$

Theoretically, α should be finalized with different SNRs and different decoding iterations to achieve the optimum performance since the incoming LLR extrinsic messages for a check node is varied dynamically. The computational complexity for the check-node update by this approach is essentially the same as that of the BP-based approximation except that two divisions with respect to two incoming LLR extrinsic messages with the smallest magnitudes are required for the normalized one.

(1) Theoretical Derivation of the Normalization Factors

Generally, α can be determined by either Monte Carlo simulations or theoretical derivations. A good way to specify the value of α in theory is by density evolution (DE) analysis [50][52] as introduced in section 5.5 for LLR-BP algorithm with continuous message alphabets. Theoretically, the best α_{opt} should enable the normalized BP-based approximation with the closest threshold of all $\alpha \geq 1$ to the maximum allowed value corresponding ultimate limit. However, this method seems to be too complicated in practice for a decoder design. Here we give another theoretical means by probabilistic method and compare the results by simulations and by this theoretical approach.

The accurate determination of the normalization factors is very important for the iterative decoding of LDPC codes using the normalized BP-based algorithm. For convenience, we denote $\{X_i: i=1, 2, \ldots, d_c-1\}=\{Z_{m,\bar{n}}; v_{\bar{n}} \in V(c_m)\backslash v_n\}$, and let $L_1=\hat{L}_{m,n}$ and $L_2=L_{m,n}$. On the basis of cycle-free hypothesis for a regular LDPC code with all-zero information bits stream sent for the encoder (or all-one binary symbols stream transmitted over the channel), then X_i are independent and identically distributed random variables,

whose pdf's depend only on the SNR and the decoding iterations. Thus, we formulate the expectations of the magnitudes of L_1 and L_2 as

$$E\left(|L_1|\right) = E\left[\min\left(|X_1|, |X_2|, ..., |X_{d_c-1}|\right)\right] \tag{6-6a}$$

and

$$E\left(|L_2|\right) = E\left\{\left|\ln\frac{1+\prod_{i=1}^{d_c-1}\tanh\left(X_i/2\right)}{1-\prod_{i=1}^{d_c-1}\tanh\left(X_i/2\right)}\right|\right\} \tag{6-6b}$$

Note that the natural logarithm used in (6-6b) is only for the sake of simplicity in the derivations (in Appendix C). The accuracy of the soft values X_i for the check-node update, which is delivered by the first iteration, will not only directly affect the performance of the second iteration, but also influence the error probability of the overall decoding. Thus, we finalize the normalization factors using (6-6a), (6-6b) and (6-4) from the first iteration. For the LLR-BP algorithm over an AWGN channel, before commencing the first iteration, each $Z_{m,n}$ is given the initial *a posteriori* LLR $2ar_n/\sigma^2$ distributed by Gaussian law $N\left(\mu_0, \sigma_0^2\right)$ based on channel soft output r_n, where $\mu_0 = 2a^2/\sigma^2 > 0$ and $\sigma_0^2 = 4a^2/\sigma^2$ have the same meanings for a and σ^2 as those in (5-54).

(A) Determine α for the first iteration by simulations: The normalization factors can be easily obtained by Monte Carlo simulations. First, the independent random variables X_1, X_2, ….., X_{d_c-1} are generated based on the same distributions, i.e., $N\left(\mu_0, \sigma_0^2\right)$ for the first iteration. Then, the calculation of the means of $|L_1|$ and $|L_2|$ statistically based on (6-6a) and (6-6b). Finally, the factor α is numerically obtained by (6-4).

(B) Determine α for the first iteration by theoretical way: On the other hand, we can also calculate $E(|L_1|)$ and $E(|L_2|)$ by the probabilistic means [53], which offers an efficient way to specify the normalization factor α. $E(|L_1|)$ and $E(|L_2|)$ are numerically computed by the following formulas for the first decoding iteration. The details of the derivations can be referred to the Appendix C. Thus, we have

$$E(|L_1|)=\int_0^{\mu_0}\left[1-Q\left(\frac{\mu_0-y}{\sigma_0}\right)+Q\left(\frac{\mu_0+y}{\sigma_0}\right)\right]^{d_c-1}dy$$

$$+\int_{\mu_0}^{+\infty}\left[Q\left(\frac{y-\mu_0}{\sigma_0}\right)+Q\left(\frac{y+\mu_0}{\sigma_0}\right)\right]^{d_c-1}dy \qquad \text{(6-7a)}$$

where $Q(x)=(1/\sqrt{2\pi})\int_x^{+\infty}e^{-x^2/2}dx$. Let $Y_j=|X_j|$ for $j\in\{1, 2, ..., d_c-1\}$, we get

$$E(|L_2|)=2\left(m_1+\frac{m_3}{3}+\frac{m_5}{5}+\frac{m_7}{7}+.....\right)=2\sum_{k=1}^{+\infty}\left[E\left[(\tanh(Y_j/2))^{2k-1}\right]\right]^{d_c-1} \qquad \text{(6-7b)}$$

where $m_{2k-1}=\left[E\left[(\tanh(Y_i/2))^{2k-1}\right]\right]^{d_c-1}$ (k =1, 2, …..}. Note that $E(|L_2|)$ can be easily computed by numerical method for the first decoding iteration, where X_i is a Gaussian variable distributed by $N(\mu_0,\sigma_0^2)$ ($\mu_0>0$), and thus pdf of Y_i has a very simple expression as in (C–2). Note that this principle of normalization can also be applied to enhance the performance of iterative turbo decoding with Max-Log-MAP algorithm [53].

(2) Comparison for the Results by Theoretical Analysis and Simulations

The first few terms of (6-7b) are usually enough to present a satisfactory estimation of $E(|L_2|)$ in most cases. Combined with the value $E(|L_1|)$ given in (6-7a), we can get the theoretical values of the normalization factors for the first iteration. Figs. 6.1 and 6.2

show the normalization factors obtained for several regular LDPC codes with different code rates and different values of d_c in various SNR values, both by simulation and theoretical analysis. In the evaluation of (6-7b), the first five terms are taken into account, i.e., k=1, 2, 3, 4 and 5. Basically, the results observed from Fig. 6.1 are quite close for the three regular LDPC codes of rates 1/3, 191/273 and 813/1057 by both methods, whose values of d_c are 9, 17 and 33, respectively. However, for the two rate-1/2 regular LDPC codes with d_c as 6 and 8, and SNR values greater than 3 dB as shown in Fig. 6.2, two approaches do not give very close estimations. Therefore, in this case the first five terms in (6-7b) cannot offer a good enough theoretical estimates, and thus more terms have to be added in order to compute $E(|L_2|)$ accurately.

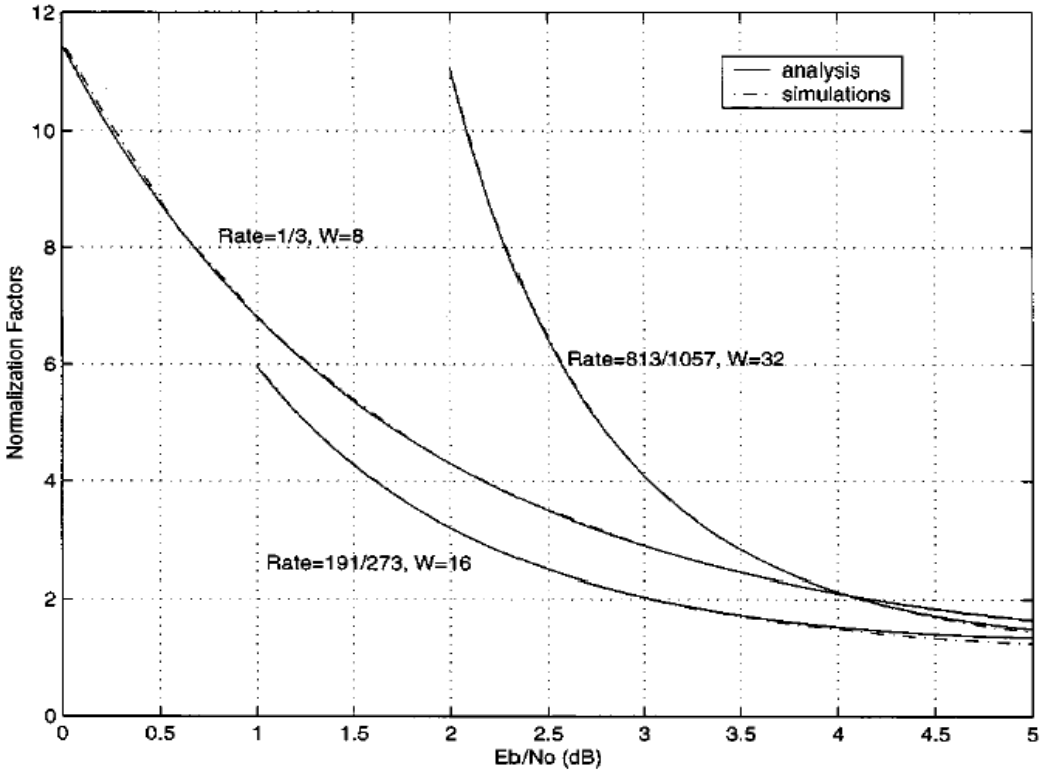

Fig. 6.1 Normalization factors obtained by simulations and theoretical analysis for some regular LDPC codes ($W = d_c - 1$).

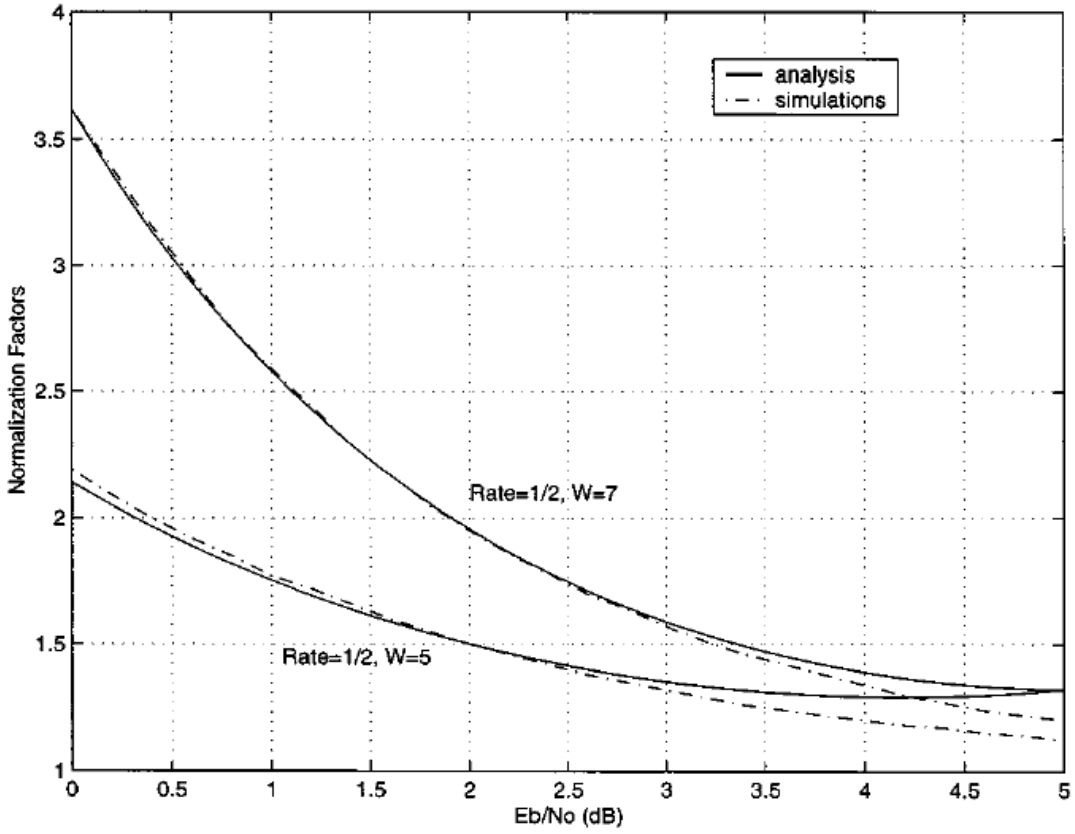

Fig. 6.2 Normalization factors obtained by simulations and theoretical analysis for some rate-1/2 regular LDPC codes ($W = d_c - 1$).

The normalization factors after the first iteration are not so straightforward to derive by the aforementioned theoretical method, while Monte Carlo simulations may be a practical solution to this problem. However, we choose not to perform the determinations from iteration to iteration, since simulations show that the performance remains quite good if the factor acquired from the first decoding iteration is kept for all subsequent iterations, and very little improvement can be achieved no matter how we change the factors from one iteration to another. This important fact has been clearly proved by the subsequent simulation results in Figs. 6.7 and 6.9.

In a recently published paper [59], the message-passing iterative decoding for LDPC codes, in which the decoder can dynamically choose its optimum decoding rule from a set of decoding algorithms at each iteration, is given a very interesting name as "Gear-Shift-Decoding".

6.1.3 Offset BP-Based Approximation

Like the method using a normalization constant α greater than one in a check-node update, another computationally efficient way [46], which can approximately compensate the difference between the values of $L_{m,n}$ and $\hat{L}_{m,n}$ in (6-2) for each check-node update of the normalized BP-based approximation, is obtained by subtracting a correction positive constant value β for the core operation $L(u_1 \oplus u_2)$ as follows:

$$\breve{L}_{m,n} = \left(\prod_{v_{\bar{n}} \in V(c_m) \backslash v_n} sign\left(Z_{m,\bar{n}} \right) \right) \times \max \left\{ \min_{v_{\bar{n}} \in V(c_m) \backslash v_n} \left| Z_{m,\bar{n}} \right| - \beta, 0 \right\} \tag{6-8}$$

where β can be finalized by the method in [49] as well as the density evolution. The use of a fixed β only incurs a negligible loss in performance with respect to LLR-BP decoding algorithm [50]. This approach differs from the normalization scheme and LLR messages smaller in magnitude than β are set to zero, thereby removing their minor contributions in the next step for variable-node update.

Note that the decoders, employing BP-based, normalized BP-based, and offset BP-based reduced-complexity algorithms, still need to know the prior knowledge of SNR at initial stage in order to perform an exact iterative decoding as well as the received channel soft outputs as their inputs. As indicated by (5-54) and (5-55), the evaluated

$L(d_n|r_n)$ based on the channel soft output r_n, which is closely related to SNR, is initially assigned to $Z_{m,n}$ for each m with $c_m \in C(v_n)$.

6.1.4 APP-Based Variable-Node Simplifications

The computation of the outgoing LLR extrinsic messages at a variable node can be carried out in a similar manner as that of the extrinsic messages at a check node. More specifically, first obtain Z_n by computing (5-60), and then get the extrinsic messages $Z_{m,n}$ by individually subtracting $L_{m,n}$ from Z_n. The variable-node update can be further simplified by only evaluating the *a posterior* LLR for each d_n by (5-60). Hence, the resulted outgoing extrinsic messages associated with d_n are the same for each check, in which d_n participates, i.e., $Z_{m,n}=Z_n$ for $c_m \in C(v_n)$. Decoding algorithm with this variable-node update simplification is called APP algorithm because it simplifies the calculation of LLR of the *a posterior* probabilities (APPs). This simplification can greatly reduce not only the computational complexity, but also the storage requirements. Other reduced-storage decoding algorithms for LDPC codes can be referred to the references [55][56].

6.1.5 Other Memory-Efficient Decoding Approaches for LDPC Codes

Besides the decoding complexity for the LLR-BP algorithm, another disadvantage of LDPC code is that the decoder by BP algorithm requires a lot of memory (number of fixed-point numbers that need to be stored). The memory requirement at the decoder increases linearly with the number of edges in the graph of LDPC codes, which could turn out to be a bottleneck in many applications. In [55], a new scheme is proposed to split the LDPC code in two subcodes and decode them in a turbo fashion, instead of

applying the BP algorithm to the entire graph. Splitting a graph into more than two halves will definitely result in more memory saving, but the resulting decoding algorithm will be more suboptimum, as correlated message will have to be exchanged between the subcodes. The tradeoff between the two effects is carefully investigated in [56].

6.1.6 Computational Complexity Comparison for LLR-BP Algorithm and BP-Based Approximations

Table 6.1 summarizes the computational complexity of various check-node updates for the reduced-complexity derivatives, including BP-based, normalized BP-based and offset BP-based approximations.

Algorithm	Multiplications	Additions	Special Operations
BP-based	None	$d_c+\lceil\log d_c\rceil-2$	None
Normalized BP-based	2	$d_c+\lceil\log d_c\rceil-2$	None
Offset BP-based	None	$d_c+\lceil\log d_c\rceil$	None

Table 6.1 Computational complexity of check-node updates for various reduced-complexity BP-based decoding algorithms. [46, Tab. 1]

Compared with the results listed in Table 5.2 for LLR-BP algorithm by different representations, the computational complexity of check-node updates is significantly reduced for the three reduced-complexity BP-based decoding algorithms. The main operations are only basic multiplications and additions for the simplified approaches, and

some computationally intensive operations necessary for LLR-BP algorithm, such as $L(U \oplus V)$, hyperbolic tangent function $\tanh(x)$ and its inverse function $\tanh^{-1}(x)$, are completely avoided.

6.2 Density Evolution Analysis and Performances for Unquantized BP-Based Approximations with Continuous Message Alphabets

Density evolution is a well-known effective method to analyze the performance of message-passing iterative decoding algorithms. In this section, we will focus on analyzing the density evolutions of the BP-based approximated algorithms for regular LDPC codes over an AWGN channel, which are very important for many practical implementations, such as FPGA-based hardware and DSP-based software iterative decoders, etc. Simulation results are also presented to demonstrate the performances of different decoding algorithms with codes of finite lengths.

Recall that we have studied the density evolution for message-passing decoding algorithms (Gallager's decoding algorithm A and B, and iterative algorithm with erasures in the decoder) with discrete message alphabets over BSC in Section 5.3, and LLR-BP decoding algorithm with continuous message alphabets in Section 5.5, which provides a general solution to the unquantized LLR-BP algorithm for regular LDPC codes.

The similar analytical method and assumptions, given in Section 5.5, can be applied to the aforementioned three BP-based approximations with continuous message alphabets (without quantization) for regular LDPC codes in this section, i.e., BP-based, normalized BP-based and offset BP-based approximations without quantization. Note that only

check-node update should be given especially attention since the variable-node update is the same as the LLR-BP algorithm.

6.2.1 Density Evolution for Unquantized BP-Based Approximation

The main steps for the evaluation of the probability density evolution for the BP-based approximated algorithm with continuous message alphabets are categorized as follows, where the noisy channel is binary-input continuous-output without quantization and the channel output alphabet as the decoder input alphabet is equal to the message alphabets for both check-node and variable-node updates.

(1) Calculation of $P_L(l)$ for the BP-based Approximation in the Current Round Decoding Iteration

For check-node update in *step* 2 of LLR-BP algorithm, L is a function of $d_c - 1$ independent and identically distributed random variables Z_1, Z_2, ….., Z_{d_s-1} due to the cycle-free graph for regular LDPC code and distributed by pdf $P_Z(z)$ obtained from the preceding iteration, and the mapping function for a check-node update is $L = \prod_{i=1}^{d_c-1} sign(Z_i) \min_{1\le i\le d_c-1} |Z_i|$ as given by (6-2). Let us define two basic functions for $x > 0$

$$\phi_+(x) = \int_x^{+\infty} P_Z(z)dz \tag{6-9a}$$

and

$$\phi_-(x) = \int_{-\infty}^{-x} P_Z(z)dz \tag{6-9b}$$

Apparently, $\phi_+(x)$ and $\phi_-(x)$ means that the probabilities of Z_i (i=1, 2, ….., d_c-1) with magnitude greater than or equal to x, and sign + and −, respectively. Then, the pdf of L is calculated by

$$P_L(l)=\frac{d_c-1}{2}\left\{\left[\phi_+(|l|)-\phi_-(|l|)\right]^{d_c-2}\left[P_Z(l)-P_Z(-l)\right]\right.$$

$$\left.+\left[\phi_+(|l|)+\phi_-(|l|)\right]^{d_c-2}\left[P_Z(l)+P_Z(-l)\right]\right\} \quad (-\infty<l<+\infty) \tag{6-10}$$

The derivation of $P_L(l)$ can be referred to the Appendix D for both $l\geq 0$ and $l<0$. The evaluated $P_L(l)$ will be used in the derivation of the updated pdf $P_Z(z)$ with respect to the variable-node update (*step* 3 of LLR-BP algorithm) of the BP-based approximation.

In particular, for an AWGN channel and in the first iteration the pdf $P_Z(z)$ is simply distributed by the well-known Gaussian law $N(\mu_Z,\sigma_Z^2)$ based on (5-54) and (5-55) with the mean $\mu_Z=2a^2/\sigma^2$ and the variance $\sigma_Z^2=4a^2/\sigma^2$ due to the assumption of all $+a$'s symbols transmitted over the channel.

(2) Calculation of $P_Z(z)$ for the BP-based Approximation for the Next Round Decoding Iteration

The variable-node updates are the same for LLR-BP algorithm and other reduced-complexity BP-based approximations, which are all based on (5-59). The random variables L_1, L_2, ….., L_{d_v-1} incoming from d_v-1 check nodes are also independent and identically distributed due to the same assumption of cycle-free graph as for check-node update, and each of L_j (j=1, 2, ….., d_v-1) distributed by (6-10) is also independent with

the initial a posteriori LLR F . Hence, according to (5-59) $P_Z(z)$ can be generally expressed as the convolution of the densities of F and L_j

$$P_Z(z)=P_F(f)*\left[P_{L_1}(l_1)*P_{L_2}(l_2)*.....*P_{L_{d_v-1}}(l_{d_v-1})\right]$$

$$=P_F(z)*\underbrace{\left[P_L(z)*P_L(z)*.....*P_L(z)\right]}_{d_v-1} \tag{6-11}$$

Thus, $P_Z(z)$ can be numerically evaluated by FFT (characteristic function) and IFFT (inverse Fourier transform) techniques since only the operation of additions for multiple independent random variables is involved. Then $P_Z(z)$ will be adopted in the calculation of the updated pdf $P_L(l)$ regarding the check-node update in *step* 2 of the BP-based approximation.

In particular, for an AWGN channel F is simply a Gaussian variable by $N(\mu_F, \sigma_F^2)$ with $\mu_F=2a^2/\sigma^2$ and $\sigma_F^2=4a^2/\sigma^2$ based on the assumption that all binary symbols $+a$'s are transmitted.

6.2.2 Density Evolution for Unquantized Normalized and Offset BP-Based Approximations

For the normalized and offset BP-based decoding algorithms, it is straightforward to modify the density evolution for the BP-based algorithm by taking normalization and offset into account. In the variable-node update, the procedure is completely the same as in the BP-based algorithm for both the normalized and the offset BP-based algorithms. In each check-node update, we first calculate the pdf $P_L(l)$ by (6-10) that is the same as for

the BP-based algorithm. Then the pdf's of outgoing messages from a check node regarding the normalized and offset BP-based algorithms are further derived as

(1) Normalized BP-based approximation

Let the random variable L_N be the outgoing message from a check node for the normalized BP-based approximation, we have following relationship based on (6-5)

$$l_N = \frac{l}{\alpha} \tag{6-12}$$

where the random variable L with pdf $P_L(l)$ is the outgoing message from a check node for the BP-based approximation. The pdf of L_N is easily obtained by the above variable transformation as

$$P_{L_N}(l_N) = \left|\frac{dl}{dl_N}\right| P_L(\alpha l_N) = \alpha P_L(\alpha l_N) \tag{6-13}$$

(2) Normalized BP-based approximation

Similarly, let the random variable L_{off} be the outgoing message from a check node for the offset BP-based approximation, the following equation holds based on (6-8) with $\beta > 0$

$$l_{off} = \begin{cases} l - \beta & l > \beta \\ 0 & -\beta \le l \le \beta \\ l + \beta & l < -\beta \end{cases} \tag{6-14}$$

The pdf of L_{off} is easily obtained by the above variable transformation as

$$P_{L_{off}}(l_{off}) = \begin{cases} P_L(l_{off} + \beta) & l_{off} > 0 \\ P_0 & l_{off} = 0 \\ P_L(l_{off} - \beta) & l_{off} < 0 \end{cases} \tag{6-15a}$$

where $P_0 = \int_{-\beta}^{+\beta} P_L(l)dl$ results from setting all L in the range $[-\beta,+\beta]$ to zero in the offset processing. The pdf $P_{L_{off}}(l_{off})$ can be also expressed by a compact single form as

$$P_{L_{off}}(l_{off}) = P_L(l_{off}+\beta)u(l_{off}) + P_L(l_{off}-\beta)u(-l_{off}) + P_0\delta(l_{off}) \tag{6-15b}$$

where the unit-step function $u(l_{off})$=1 for $l_{off} \geq 0$ and $u(l_{off})$=0 otherwise, and $\delta(l_{off})$ is the impulse function. Clearly, $\delta(l_{off})$ function introduced in $P_{L_{off}}(l_{off})$ with offset disappears in the evaluation of the pdf $P_Z(z)$ due to the convolutional operations. Note that both normalization and offsetting preserve the sign of the resulting LLRs and reduce their magnitudes.

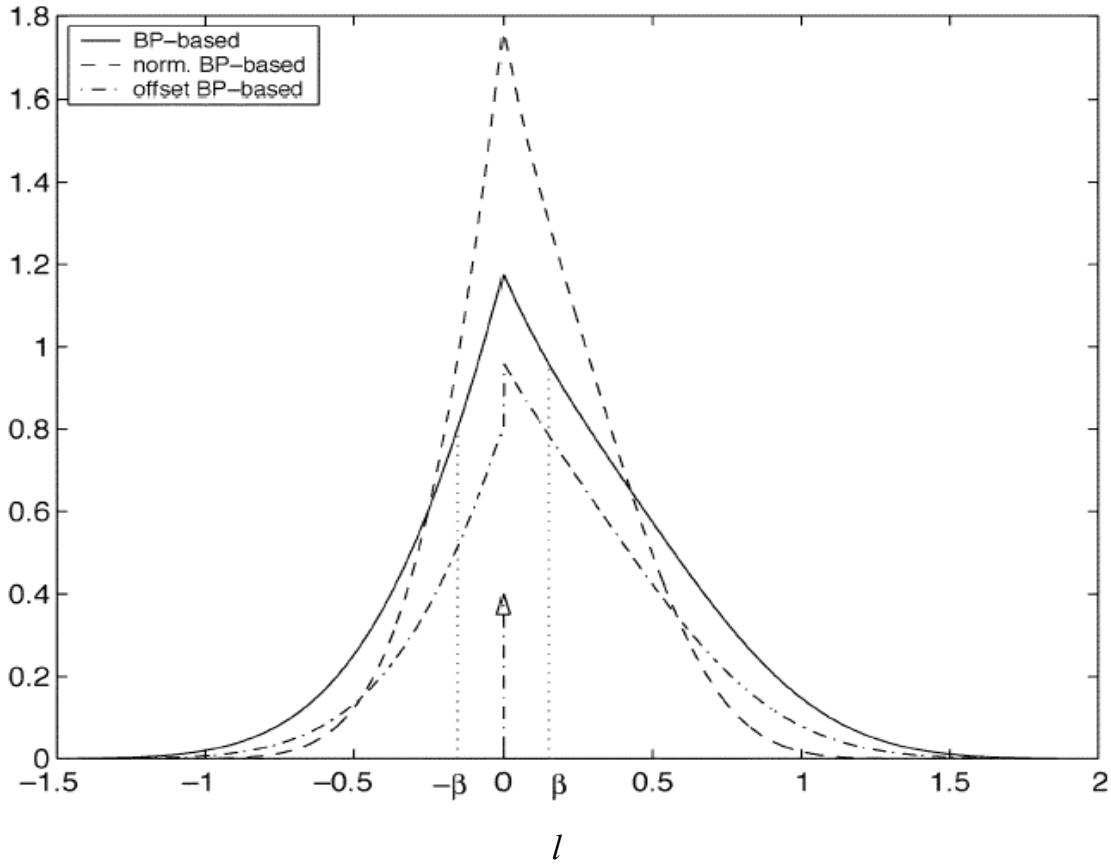

Fig. 6.3 An example of asymptotical performance of updated $P_L(l)$ for BP-based approximation, $P_N(l)$ and $P_{off}(l)$ for normalized and offset BP-based approximations, respectively.

Fig. 6.3 gives an example of $P_L(l)$, $P_N(l)$ and $P_{off}(l)$ for the BP-based, normalized and offset BP-based approximations, where, without danger of confusion, we use variable l for all the three pdf's. It can be observed that with either normalization or offsetting, the densities $P_N(l)$ and $P_{off}(l)$ is more concentrated toward the origin such that the mean of magnitude $|L|$ is thus decreased.

6.2.3 Performances of the Unquantized LLR-BP Algorithm, BP-based, Normalized and offset BP-based Approximations

(1) Thresholds for various unquantized decoding algorithms

The density evolutions for the LLR-BP, BP-based, normalized and offset BP-based algorithms have been theoretically analyzed in the previous sections. Now we perform the numerical calculations of the thresholds for the normalized and offset BP-based algorithms with different values α and β and different code ensembles. Here the threshold is expressed as $\sigma_* = 10\log\left(1/(2R(\sigma^*)^2)\right)$ (in dB) only for the sake of clear presentation, where R is the code rate and σ^*, defined by the previous sections, is the threshold in terms of the standard deviation of channel noise σ with the transmitted signal as either +1 or –1.

The results for three rate-1/2 regular LDPC codes families are given in Figs. 6.4 and 6.5. With a properly chosen α and β, the two improved algorithms can achieve performances very close to that the LLR-BP algorithm.

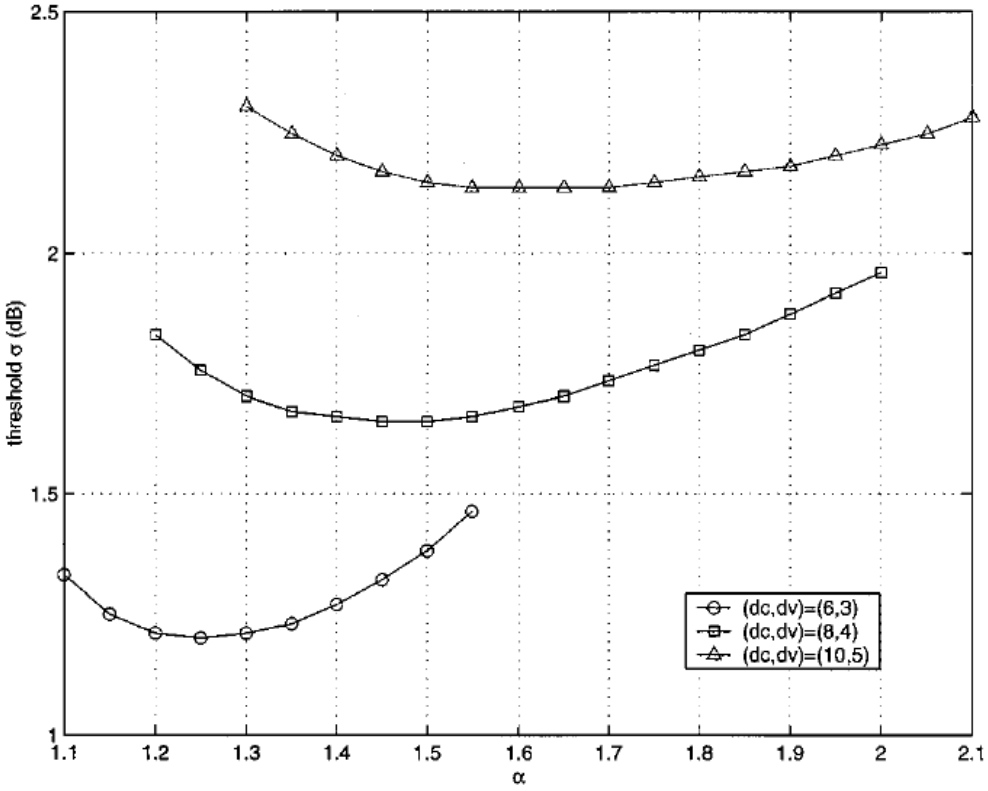

Fig. 6.4 Threshold σ (in dB) for the normalized BP-based algorithm.

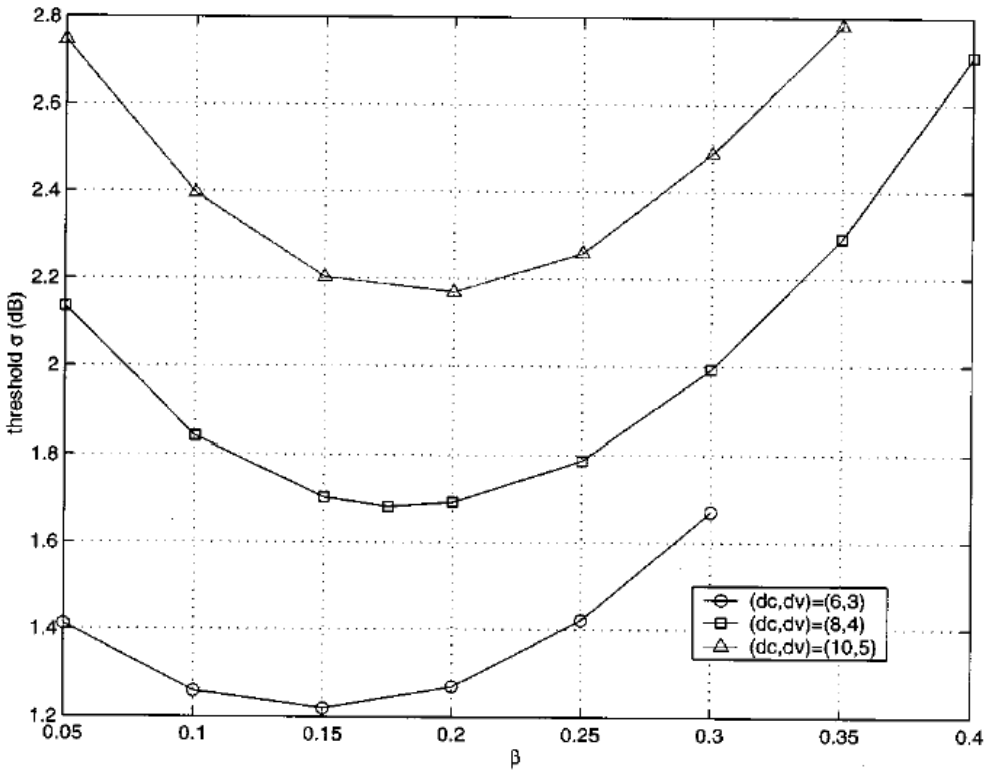

Fig. 6.5 Threshold σ (in dB) for the offset BP-based algorithm.

The best possible thresholds (in dB) of the two improved BP-based algorithms are summarized in Table 6.2 with the thresholds for the LLR-BP and BP-based algorithms included for comparisons. For instance, threshold σ_*=1.11 for (3, 6) LDPC code using LLR-BP algorithm is corresponding to σ^*=0.88 as listed in Table 5.3 under the same conditions. For the same code the thresholds σ_* for the normalized and offset BP-based algorithms are 1.20 and 1.23, which corresponds to α^* as 0.871 and 0.868, respectively, which is very close to that of the LLR-BP algorithm.

(d_v, d_c)	LLR-BP σ_*/σ^*	BP-based σ_*/σ^*	Normalized BP-based		Offset BP-based	
			α	σ_*/σ^*	β	σ_*/σ^*
(3, 6)	1.11/0.88	1.71/0.821	1.25	1.20/0.871	0.15	1.23/0.868
(4, 8)	1.62/0.83	2.50/0.750	1.50	1.65/0.827	0.175	1.70/0.822
(5, 10)	2.04/0.79	3.10/0.700	1.65	2.14/0.782	0.20	2.17/0.779

Table 6.2 Thresholds (in dB) for various decoding algorithms. ([52], Table 1)

(2) Bit error rate performance for various unquantized decoding algorithms

Using density evolution for the normalized BP-based and offset BP-based decoding, we can optimize the decoder parameters by varying α or β in calculation of the thresholds, and selecting those α or β that can offer the best performance. Fig. 6.6 presents the numerical results of the two improved BP-based algorithms, i.e., normalized and offset BP-based algorithms, for a rate-1/2 LDPC code (8000, 4000) with (d_v=3, d_c=6) and 100 decoding iterations, and compares with those obtained from the LLR-BP and the

BP-based algorithms. Although the code length is far from infinite, we can still observe the effectiveness of the two improved BP-based algorithms. For this code, the gap between LLR-BP and BP-based algorithm is about 0.5dB, very close to the 0.6dB gap predicted for infinite code length. However, by properly chosen the best parameters α=1.25 and β=0.15 for this regular LDPC codes family by density evolution, two improved BP-based algorithms exhibit less than 0.1dB degradation in BER performance compared with the LLR-BP algorithm. More importantly, the results strongly suggest that little additional improvements can be achieved by further allowing either α or β to change adaptively with SNRs and decoding iterations. The normalized BP-based algorithm slightly outperforms the offset BP-based algorithm, but may be slightly more complex to implement. For comparison, the performances of these two algorithms with α=1.6, determined by the probabilistic method by (6-4), and β=0.25 are also plotted. Clearly, these parameter choices are not as good as the proposed one in this case, which clearly verifies that density evolution rather than the probabilistic method is the best approach for long code length.

Fig. 6.7 compares the performances for the two improved BP-based algorithms for the same rate-1/2 regular LDPC codes as in Fig. 6.6 and (1008, 504) with (d_v=3, d_c=6). Even for medium and short code length, the values of α and β obtained from density evolution are still excellent to achieve a close-to-BP performance. Interestingly, for the (1008, 504) code two improved algorithms have even a slightly better performance than the BP algorithm at high SNR values. The reason may be the fact that the BP algorithm is

not optimum for medium or short code lengths due to correlation among messages processed during the iteration decoding. However, the two improved BP-based algorithms seem to be able to reduce the negative effect of correlations. Optimizing α and β by density evolution may not be applicable to LDPC codes with many short cycles in their representation, e.g., such Gallager codes or Euclidean geometry codes. An alternative effective method to obtain α for such codes is the probabilistic means by (12.59), which has been proved to be effectiveness by the simulation results in [53].

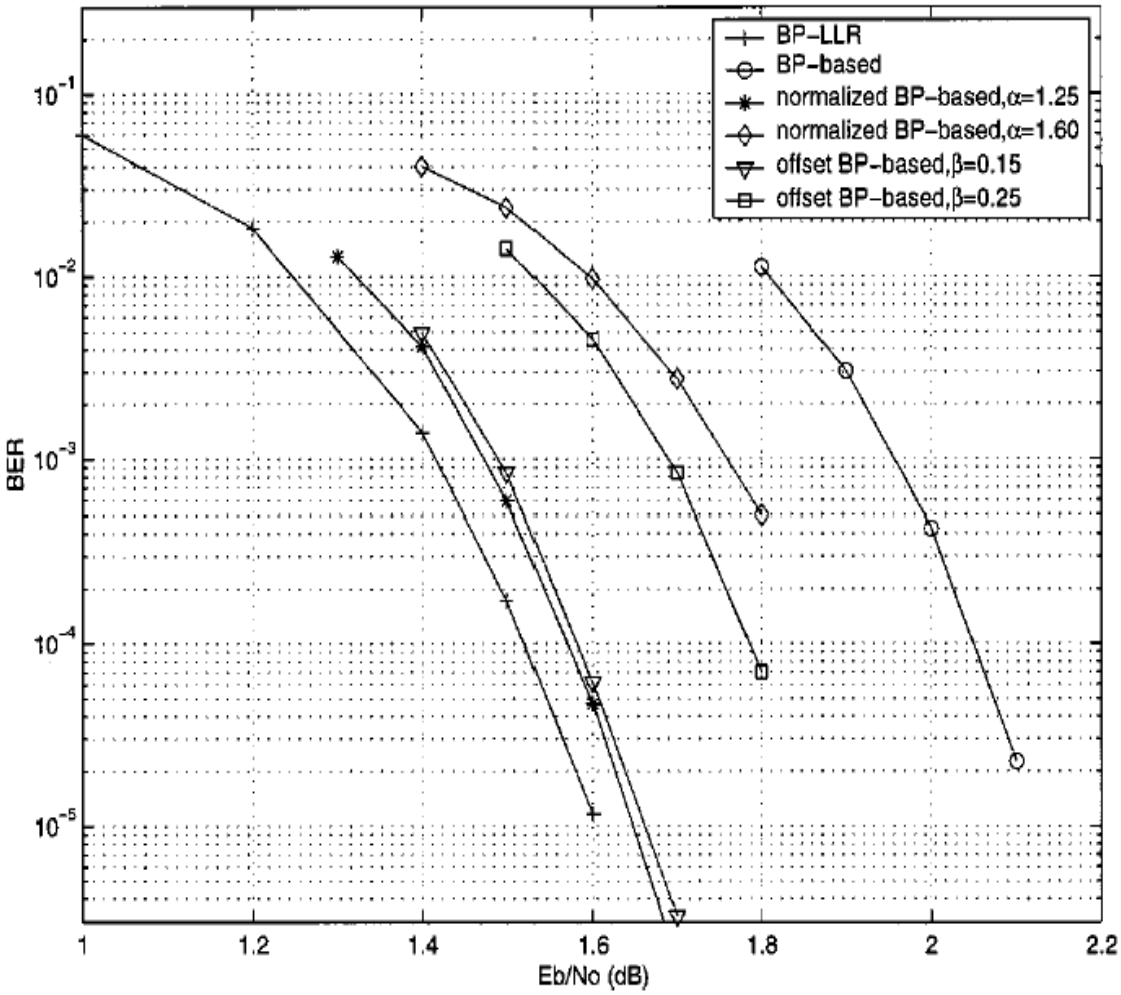

Fig. 6.6 Bit error rate of a rate-1/2 (8000, 4000) regular LDPC code with (d_v=3, d_c=6) and 100 decoding iterations under various decoding algorithms.

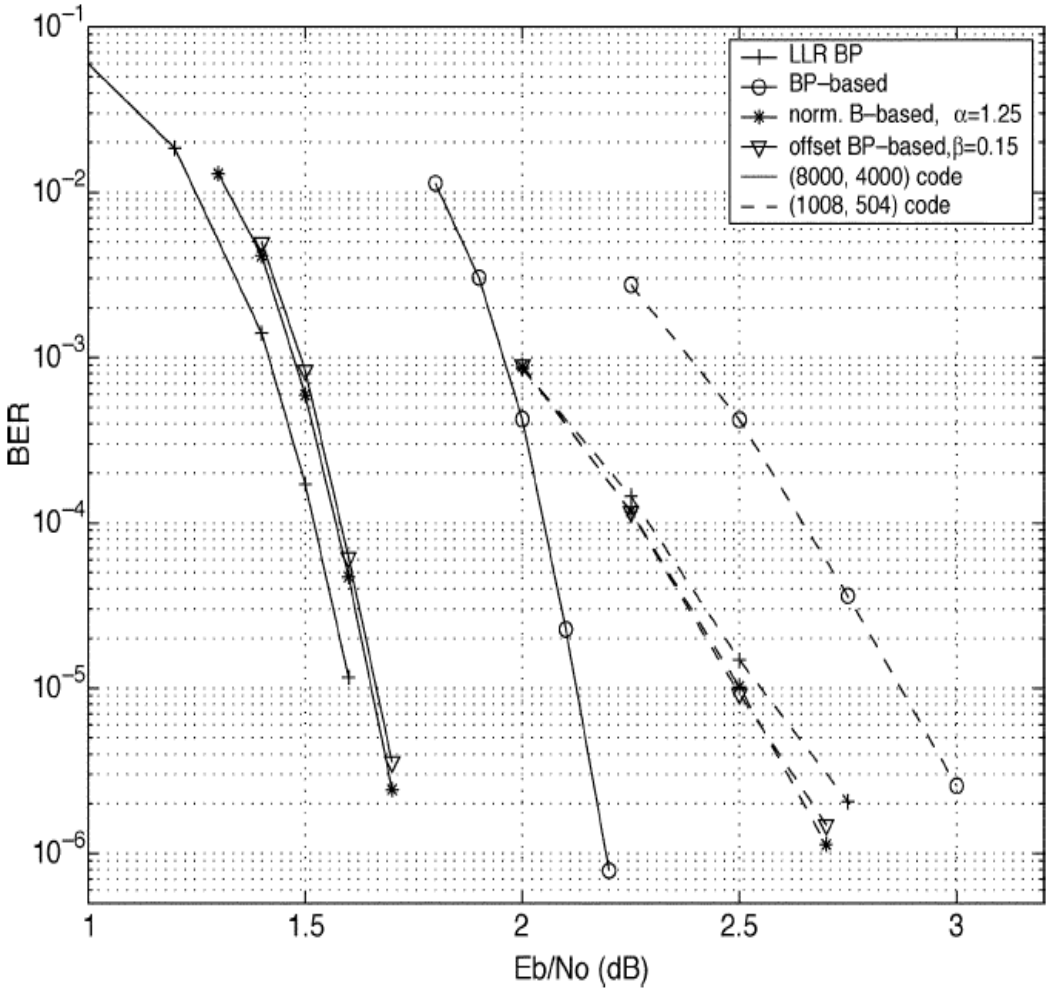

Fig. 6.7 Bit error rate of two rate-1/2 regular LDPC codes (8000, 4000) and (1008, 504) with (d_v =3, d_c =6) and 100 decoding iterations under various decoding algorithms.

6.3 Density Evolution Analysis and Performances for Quantized BP-Based Approximations

In this section we will investigate the density evolution and bit error rate (BER) performance of quantized BP-based approximations. In order to well understand the quantization scheme for the message-passing decoding for LDPC codes, we first present a very simple example [9] of time-invariant decoder on the (3, 6) regular LDPC code over a quantized BIAWGN channel.

Example 6.1: **Quantized BIAWGN channel with 3-bit messages for (3, 6) regular LDPC codes**: Consider a memoryless binary-input 3-bit quantized output additive Gaussian noise channel with the received alphabet set as

$$A_R = \{-4, -3, -2, -1, 1, 2, 3, 4\} \tag{6-16}$$

Let the message alphabet set A_M be equal to A_R. In the first iteration, the initial Z_i (i=1, 2, 3, 4, 5) for the decoder is the quantized *a posteriori* LLR of the ith channel soft output specified by (5-54), which is distributed by the Gaussian law $N(\mu_Z, \sigma_Z^2)$ ($\mu_Z = 2a^2/\sigma^2 > 0$, $\sigma_Z^2 = 4a^2/\sigma^2$) due to all $+a$'s symbol sent over the channel. Let the threshold set be given by

$$\tau = \{-\infty, -4.2, -2.8, -1.4, 0.0, 1.4, 2.8, 4.2, +\infty\} \tag{6-17}$$

and quantize the LLR value according to the specified thresholds, e.g., Z_i is 3 if the ith LLR falls into the interval [2.8, 4.2), and Z_i is 4 corresponding to the interval [4.2, $+\infty$).

The mapping function for a check-node update is

$$\Phi_c(Z_1, Z_2, Z_3, Z_4, Z_5) = sM \tag{6-18}$$

where the sign s is given by

$$s = \prod_{i=1}^{d_c - 1} sign(Z_i) \tag{6-19}$$

and where M is the reliability value specified by the following decision strategy. Let n_q ($q \in \{1, 2, 3, 4\}$) be the number of received messages out of all 5 messages under consideration with reliability value q.

1) If $n_1 > 0$ or $n_2 > 3$, then $M = 1$.

2) If n_2>1, then M =2.

3) If n_2=1 and n_3>1, then M =2.

4) If n_2=1 or n_3>2, then M =3.

5) Otherwise, M =4.

For a variable-node update, we first define a function ϕ_R that maps the quantized LLR message $F \in A_R$, received from the channel soft output, into another symbol set as

$$\phi_R: A_R \to \{-21, -15, -9, -3, 3, 9, 15, 21\} \tag{6-20}$$

where ϕ_R is formulated as

$$\phi_R(F)=6F-3\,sign(F) \tag{6-21}$$

Then let the function ϕ_M be defined to map the quantized LLR message from check-node update into a new symbol set as

$$\phi_M: A_M \to \{-12, -8, -6, -2, 2, 6, 8, 12\} \tag{6-22}$$

where ϕ_M is formulated as

$$\phi_M(L_i)=2\,sign(L_i)\left(|L_i|+\left\lfloor\frac{|L_i|}{2}\right\rfloor\right) \quad (i=1, 2) \tag{6-23}$$

Hence, for a variable-node update we have

$$\Phi_v(F, L_1, L_2)=\phi_R(F)+\phi_M(L_1)+\phi_M(L_2) \tag{6-24}$$

The following threshold is set so that the quantized output integer symbol falls into the set A_R.

$$\tau_M=\{-\infty, -18, -12, -6, 0, 6, 12, 18, +\infty\} \tag{6-25}$$

For example, if $F=2$, $L_1=-4$ and $L_2=3$ then

$$\phi_R(2)+\phi_M(-4)+\phi_M(3)=9-12+8=5 \tag{6-26}$$

Thus, the outgoing message from a variable-node update is +1 since $0<5<6$. □

Further discussions on the example: The obvious advantage of this decoder is very low computational complexity for the implementation in practice. The decoding algorithm on the (3, 6) regular LDPC codes has a threshold [9] $\sigma^*=0.847$, corresponding to a raw bit error rate (BER) of about $Q(1/\sigma^*)=0.119$ if the continuous output of the BIAWGN channel is quantized by 1-bit and decided by hard-decision with uncoded scheme. Recall that the LLR-BP decoding algorithm on the same code with continuous alphabet messages has a threshold of $\sigma_{BP}^*=0.88$ as listed in Table 6.2, which corresponds to a raw BER of approximately $Q(1/\sigma_{BP}^*)=0.13$. Therefore, the decoder with a very simple quantization scheme can offer a close performance to that of the LLR-BP algorithm with continuous message alphabets over the BIAWGN channel.

Simulation results of the example: Fig. 6.8 shows the simulated performance of the (3, 6) regular LDPC code by bit error probability versus σ in a BIAWGN channel, where the message-passing decoding algorithm with 3-bit quantization in this example and the LLR-BP algorithm with continuous message alphabets are employed. Normally, the performance asymptotically gets closer to the ultimate case for each algorithm with the increasing block length as indicated by its threshold. It is also wroth to note that the decoder with a very simple quantization scheme can result in a good trade-off between the computational complexity and bit error rate performance.

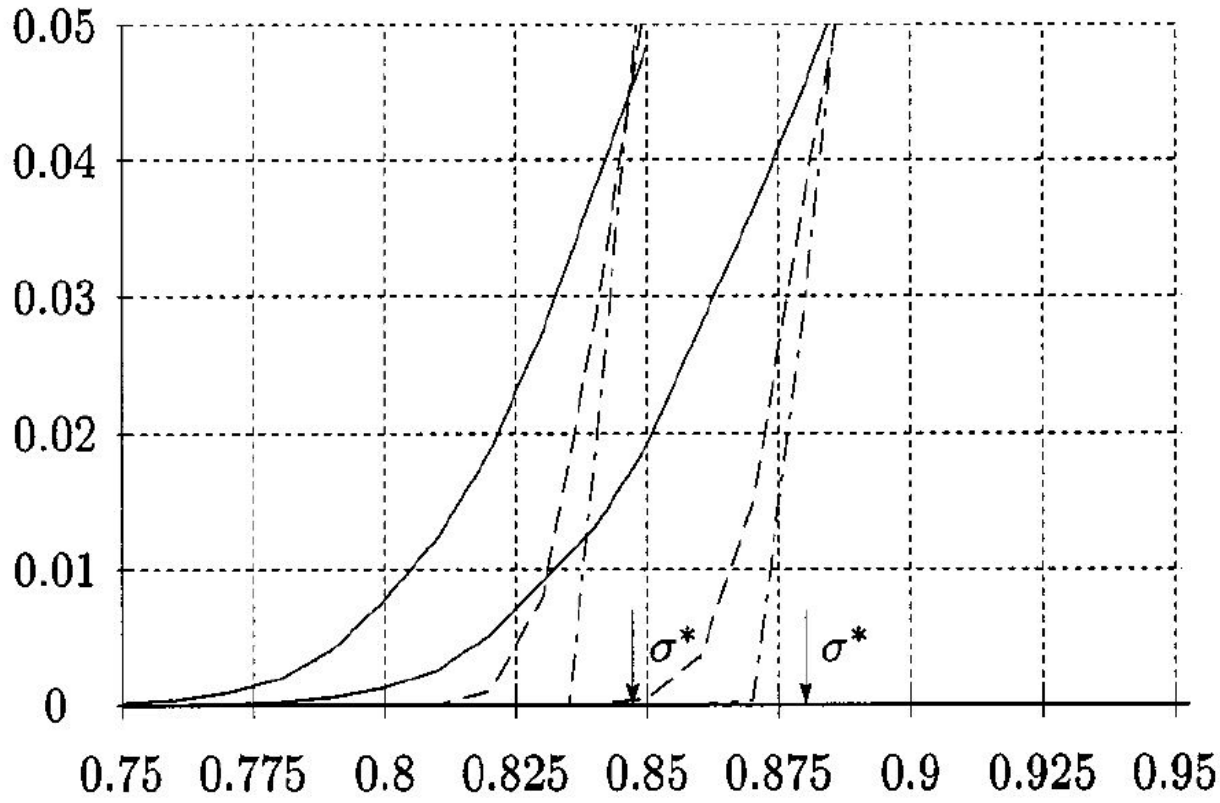

Fig. 6.8 The bit error probability versus σ for the ensemble of (3, 6) LDPC codes over a BIAWGN channel. The left and right curves correspond the message-passing algorithm with 3-bit quantization described in example 6.1 and LLR-BP algorithm with continuous message alphabets. The solid, dashed and dotted-dashed curves correspond to block lengths of 10^3, 10^4 and 10^5, respectively. The arrows indicate the related thresholds 0.847 and 0.88 for the respective algorithm.

6.3.1 Density Evolution for Quantized BP-Based Approximation

For logic circuit design with finite precision, normalization can be implemented with a look-up table, or as a register shift for normalization factors like two, four, or eight. However, in general, offset BP-based decoding is more convenient for hardware implementations than normalized BP-based decoding. This section focuses on the quantized effects for the LLR-BP algorithm, BP-based and the offset BP-based approximations. Only uniform quantizers have been considered for the above schemes.

Examining the use of nonuniform quantizers to cope with the nonlinearities of the function is an interesting research and can be referred to [54].

We first derive the density evolution for the quantized BP-based decoding in a similar way to the unquantized one. Let (q,Δ) denote a uniform quantization scheme with quantization step Δ and q quantization bits, one of which denotes the sign, while the remaining $q-1$ bits denote the magnitude. Suppose that the same quantization is taken for both received values and the extrinsic message passing in the decoder. Then the messages in the iterative decoding belong to an alphabet of size 2^q-1 represented by [$-T$, …., -1, 0, 1, …., T], where $T=2^{q-1}-1$. The quantization threshold is defined as $T_{th} \overset{\text{def}}{=} T\,\Delta$. A received value r_n is initially quantized to an integer $\bar{r}_n$, whose sign and magnitude are determined by

$$\text{sign}(r_n)=\text{sign}(\bar{r}_n) \tag{6-27a}$$

and

$$|\bar{r}_n|=\min\left(\left\lfloor(|r_n|/\Delta)+0.5\right\rfloor, T\right) \tag{6-27b}$$

where $\lfloor w\rfloor$ ($w\geq 0$) means the largest integer less than w. Here we need to redefine $\phi_+(x)$ and $\phi_-(x)$ (x=1, 2, ….., T) similar as (6-9a) and (6-9b) for unquantization scheme

$$\phi_+(x)=\sum_{i=x}^{T} P_Z(Z=i) \tag{6-28a}$$

and

$$\phi_{-}(x)=\sum_{i=-T}^{-x} P_Z(Z=i) \tag{6-28b}$$

where $P_Z(Z=i)$ is correspondingly the probability of an event Z takes on the value i, other than $P_Z(z)$ as pdf in (6-9a) and (6-9b) for unquantization scheme. Thus, for the quantized BP-based decoding, $\Pr(L\le l)$ is the probability of LLR extrinsic message L from a check node to an incident variable with $L\le l$, which can be easily derived for ($-T\le l\le -1$) similarly as (D-13) for unquantization scheme in the Appendix D. That is

$$\Pr(L\le l)=\frac{1}{2}\left\{\left[\phi_{+}(|l|)+\phi_{-}(|l|)\right]^{d_c-1}-\left[\phi_{+}(|l|)-\phi_{-}(|l|)\right]^{d_c-1}\right\} \tag{6-29a}$$

For $0\le l\le T-1$, $\Pr(L\le l)$ can be obtained by slightly modifying (D–5) in the Appendix D.

$$\begin{aligned}\Pr(L\le l)&=1-\Pr(L\ge l+1)\\ &=1-\frac{1}{2}\left\{\left[\phi_{+}(|l|+1)+\phi_{-}(|l|+1)\right]^{d_c-1}+\left[\phi_{+}(|l|+1)-\phi_{-}(|l|+1)\right]^{d_c-1}\right\}\end{aligned} \tag{6-29b}$$

Obviously, for $l=T$

$$\Pr(L\le l)=1 \tag{6-29c}$$

Therefore, the probability of an event that L takes on the value l is formulated as

$$P_L(L=l)=\begin{cases}\Pr(L=l) & l=-T\\ \Pr(L\le l)-\Pr(L\le l-1) & -T\le l\le T\end{cases} \tag{6-30}$$

Note that here $P_L(L=l)$ is the probability for $L=l$ that differs greatly from the pdf $P_L(l)$ in (6-10). In particular, for an AWGN channel the initial probability $P_Z(Z=i)$ in the first iteration is

$$P_Z(Z=i)=\begin{cases}\displaystyle\int_{-\infty}^{(i+1/2)\Delta}\frac{1}{\sqrt{2\pi}\sigma_0}\exp\left[-\frac{(r-\mu_0)^2}{2\sigma_0^2}\right]dr & i=-T\\ \displaystyle\int_{(i-1/2)\Delta}^{(i+1/2)\Delta}\frac{1}{\sqrt{2\pi}\sigma_0}\exp\left[-\frac{(r-\mu_0)^2}{2\sigma_0^2}\right]dr & -T+1\le i\le T-1\\ \displaystyle\int_{(i-1/2)\Delta}^{+\infty}\frac{1}{\sqrt{2\pi}\sigma_0}\exp\left[-\frac{(r-\mu_0)^2}{2\sigma_0^2}\right]dr & i=T\end{cases} \tag{6-31}$$

where μ_0 and σ_0^2 are the mean and variance of the a *posteriori* LLR that is expressed by (5-54).

Density evolution in variable node is almost the same as the unquantization scheme, where FFT/IFFT is adopted to compute the probability $P_Z(Z=i)$ effectively by using (6-11) with the discrete input probabilities $P_F(F=i)$ and $P_L(L=i)$ ($-T\le i\le T$). The obtained $P_Z(Z=i)$ will be adopted in (6-28a) and (6-28b) for the next round decoding iteration. Note that for a small number of quantization levels, the FFT can be replaced by direct computing the discrete convolution.

6.3.2 Density Evolution for Quantized BP-Based Approximations

For the normalized and offset BP-based approximations, the density evolutions for check-node updates can be easily derived as follows:

(1) Normalized BP-based approximation

Recall that in (6-13) where L and L_N are denoted as the outgoing LLR messages from a check node to an incident variable node for the BP-based and the normalized BP-based approximations, respectively, with the relation $l=\alpha l_N$. The probability of an event that L_N takes on the value l_N is

$$P_{L_N}\left(L_N = l_N\right) = P_L\left(\alpha L_N = l\right)$$

$$= P_L\left(L = l\right) \quad \left(l_N = -T/\alpha, \ldots, -1/\alpha, 0, 1/\alpha, \ldots, T/\alpha\right) \tag{6-32}$$

(2) Offset BP-based approximation

Recall that in (6-13) where L_{off} is denoted as the outgoing LLR message from a check node to an incident variable node for the offset BP-based approximation with the relation held in (6-14). The probability of an event that L_{off} takes on the value l_{off} is

$$P_{L_{off}}\left(L_{off} = l_{off}\right) = \begin{cases} P_L\left(L = l_{off} + \lfloor\beta\rfloor\right) & l_{off} = 1,2,\ldots,T - \lfloor\beta\rfloor \\ \sum_{i=-\lfloor\beta\rfloor}^{\lfloor\beta\rfloor} P_L\left(L = i\right) & l_{off} = 0 \\ P_L\left(L = l_{off} - \lfloor\beta\rfloor\right) & l_{off} = -1,-2,\ldots,-\left(T - \lfloor\beta\rfloor\right) \end{cases} \tag{6-33}$$

Note that for both the normalized and the offset BP-based approximations, the calculation of the probabilities $P_Z\left(Z = i\right)$ ($-T \le i \le T$) is the same as that of the BP-based approximation.

6.4 Performance Comparison between the LLR-BP Algorithm and Quantized offset BP-based Approximation $\sigma_* = 10\log\left(1/(2R(\sigma^*)^2)\right)$

(1) Thresholds for various quantized decoding algorithms

The thresholds for the quantized BP-based algorithm depend on both q and Δ. The numerical results [46] for the thresholds of regular LDPC codes with (d_v, d_c)=(3, 6) and (4, 8) using quantized BP-based algorithm and quantized offset BP-based algorithm are presented in Table 6.3. Note that the threshold in dB is similarly defined as that in Table 6.2 and without danger of confusion we simply use σ rather than σ_* in this table. The messages passing in the iterative decoding belong to an alphabet of size $2^q - 1$,

represented by $-T$, …., -1, 0, 1, …., T, where $T=2^{q-1}-1$. The quantization threshold is defined as $T_{th}=T\Delta$.

(d_s, d_c)	BP	BP-based	Quantized BP-based			Quantized Offset BP-based		
	σ (dB)	σ (dB)	(q, Δ)	T_{th}	σ (dB)	(q, Δ, β)	T_{th}	σ (dB)
(3, 6)	1.11	1.71	(4, 0.1)	0.7	1.93	(4, 0.15, 1)	1.05	2.05
			(5, 0.025)	0.375	2.93	(4, 0.2, 1)	1.4	1.47
			(5, 0.05)	0.75	1.86	(5, 0.075, 1)	1.125	1.47
			(5, 0.1)	1.5	**1.58**	(5, 0.075, 2)	1.125	1.60
			(6, 0.025)	0.775	1.82	(5, 0.15, 1)	2.25	1.24
			(6, 0.05)	1.55	**1.58**	(6, 0.05, 2)	1.55	1.30
			(6, 0.1)	3.1	1.71	(6, 0.05, 3)	1.55	1.29
			(7, 0.025)	1.575	**1.58**	(6, 0.05, 4)	1.55	1.36
			(7, 0.05)	3.15	1.71	(6, 0.075, 2)	2.325	**1.22**
			(7, 0.1)	6.3	1.72	(6, 0.15, 1)	4.65	1.24
			(8, 0.025)	3.175	1.71	(7, 0.05, 2)	3.15	1.26
			(8, 0.05)	6.35	1.71	(7, 0.05, 3)	3.15	**1.22**
						(7, 0.05, 4)	3.15	1.28
						(7, 0.075, 2)	4.725	**1.22**
						(7, 0.15, 1)	9.45	1.24
(4, 8)	1.62	2.50	(4, 0.1)	0.7	2.43	(5, 0.0875, 2)	1.3125	1.76
			(5, 0.025)	0.375	2.77	(5, 0.175, 1)	2.625	1.72
			(5, 0.05)	0.75	2.37	(6, 0.0875, 2)	2.7125	**1.69**
			(5, 0.1)	1.5	2.43	(6, 0.15, 1)	5.425	1.73
			(6, 0.025)	0.775	**2.35**	(7, 0.0875, 2)	5.5125	**1.69**
			(6, 0.05)	1.55	2.40	(7, 0.175, 1)	11.025	1.72
			(6, 0.1)	3.1	2.51			
			(7, 0.025)	1.575	2.40			
			(7, 0.05)	3.15	2.50			
			(7, 0.1)	6.3	2.51			
			(8, 0.025)	3.175	2.50			
			(8, 0.05)	6.35	2.50			

Table 6.3 The thresholds for quantized BP-based and quantized offset BP-based decoding algorithms.

The best threshold values obtained are shown in bold. It is very interesting to find that the thresholds for some quantization schemes are even better than those for the unquantization scheme. The reason can be interpreted by the fact that the outgoing LLR extrinsic messages in a check-node update, which are overestimated by the BP-based

approximation, are upper bounded in the quantized BP-based scheme. This clipping effect could be regarded as a trivial improvement to the BP-based algorithm.

Based on the density evolution for the quantized BP-based algorithm, the quantized density evolution of the offset BP-based algorithm can be derived readily using the method described in this section. For both ensembles of (3, 6) and (4, 8) codes, the various combinations of q, Δ and β with the resulting thresholds are shown in Table 6.3. For small q with large Δ means a large quantization range and results in better performance; for large q and fine quantization, i.e., a small Δ value, is of great importance. For example, for (3, 6) codes, with q=5, the best threshold result is achieved at 1.24 when Δ=0.15 and β=1; with q=6, the best result is obtained at 1.22 with Δ=0.075 and β=2. For both codes ensembles, with either q=5 or 6, the thresholds are very close to those of the continuous case using LLR-BP algorithm.

(2) Bit error rate performance for various quantized decoding algorithms

Fig. 6.9 compares the performance of the quantized offset BP-based algorithm for the same two regular LDPC codes as indicated in Fig. 6.7 with 50 decoding iterations to that of the unquantized LLR-BP algorithm. For both codes, q=5 quantization with one bit for the sign, can perform close to the LLR-BP algorithm. Thus, 4 bits operations are involved in the check node processing. For the (1008, 504) code, the quantized offset BP-based algorithm achieves a better performance than the LLR-BP algorithm, which may be due to the similar reason as the unquantized case as shown in Fig.6.7. For the (8000, 4000) LDPC code with q=6, the quantized offset BP-based can offer better performance than

that with q =5, and suffers a degradation of less than 0.1dB, compared with the unquantized LLR-BP algorithm.

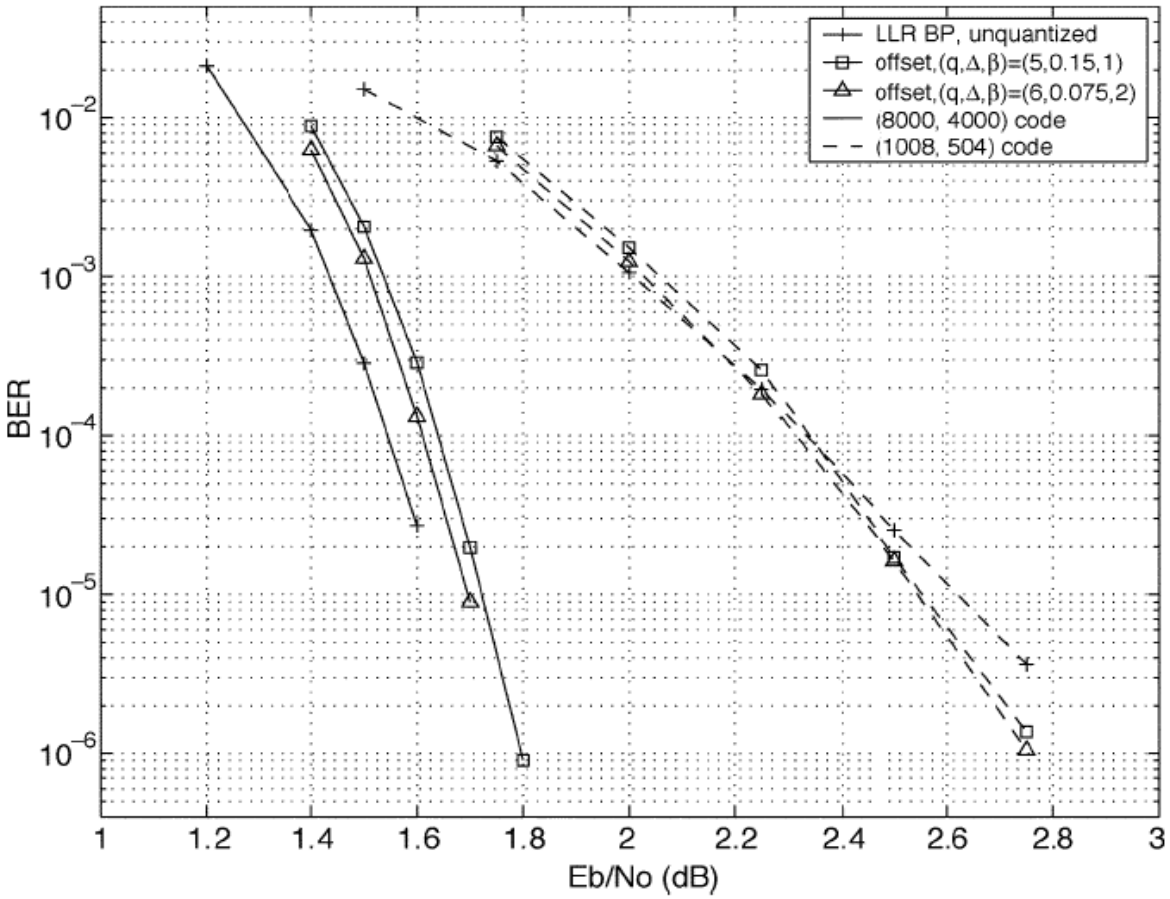

Fig. 6.9 Bit error rate of two rate-1/2 regular LDPC codes (8000, 4000) and (1008, 504) with (d_v =3, d_c =6) and 50 decoding iterations under unquantized LLR-BP algorithm and quantized offset BP-based algorithm.