CHAPTER 10

Reliability-Based Soft-Decision Decoding Algorithms for Linear Block Codes

Contributed by Marc P. C. Fossorier

All the decoding algorithms developed so far are based on hard-decision outputs of the matched filter in the receiver demodulator; that is, the output of the matched filter for each signaling interval is quantized in two levels, denoted as 0 and 1, which results in a hard-decision binary received sequence. Then, the decoder processes this hard-decision received sequence based on a specific decoding method. This type of decoding is referred to as *hard-decision decoding*. Hard-decision decodings using algebraic structures of the codes are called *algebraic decodings*. The metric used in hard-decision decodings is the Hamming distance. The objective is to decode the hard-decision received sequence to the closest codeword in the Hamming distance. A hard decision of a received signal results in a loss of information, which degrades performance.

If the outputs of the matched filter are unquantized or quantized in more than two levels, we say that the demodulator makes *soft decisions*. A sequence of soft-decision outputs of the matched filter is referred to as a *soft-decision received sequence*. Decoding by processing this soft-decision received sequence is called *soft-decision decoding*. Because the decoder uses the additional information contained in the unquantized (or multilevel quantized) received samples to recover the transmitted codeword, soft-decision decoding provides better error performance than hard-decision decoding. In general, soft-decision maximum likelihood decoding (MLD) of a code has about 3 dB of coding gain over algebraic decoding of the code; however, soft-decision decoding is much harder to implement than algebraic decoding and requires more computational complexity. This is the price to be paid for better error performance.

Many soft-decision decoding algorithms have been devised. These decoding algorithms can be classified into two major categories: *reliability-based* (or probabilistic) decoding algorithms and *code structure-based* decoding algorithms. Important reliability-based decoding algorithms are presented in this chapter. The other category of decoding algorithms will be covered in later chapters.

10.1 SOFT-DECISION DECODING

For soft-decision decoding, metrics other than the Hamming distance must be used. The most commonly used metrics are the likelihood function, Euclidean distance,

correlation, and correlation discrepancy. In this section we develop these metrics and discuss the error performance of the soft-decision MLD.

Let C be a binary (n, k) linear block code with minimum Hamming distance $d_{min}(C)$ that is used for error control. Suppose BPSK signaling is used for transmission over an AWGN channel with two-sided PSD $N_0/2$. Assume that each signal has unit energy. Let $\mathbf{v} = (v_0, v_1, \cdots, v_{n-1})$ be a codeword in C. For transmission, this codeword is mapped into a sequence of BPSK signals that, in vector form, represented by the bipolar sequence $\mathbf{c} = (c_0, c_1, \cdots, c_{n-1})$, where for $0 \leq l < n$,

$$c_l = 2v_l - 1 = \begin{cases} -1 & \text{for } v_l = 0, \\ +1 & \text{for } v_l = 1. \end{cases} \tag{10.1}$$

Let $\mathbf{c}_i = (c_{i0}, c_{i1}, \cdots, c_{i,n-1})$ and $\mathbf{c}_j = (c_{j0}, c_{j1}, \cdots, c_{j,n-1})$ be two signal sequences corresponding to codewords $\mathbf{v}_i = (v_{i0}, v_{i1}, \cdots, v_{i,n-1})$ and $\mathbf{v}_j = (v_{j0}, v_{j1}, \cdots, v_{j,n-1})$, respectively. The squared Euclidean distance between $\mathbf{c}_i$ and $\mathbf{c}_j$ is defined as

$$|\mathbf{c}_i - \mathbf{c}_j|^2 \triangleq \sum_{l=0}^{n-1} (c_{il} - c_{jl})^2. \tag{10.2}$$

For convenience, we denote this squared Euclidean distance by $d_E^2(\mathbf{v}_i, \mathbf{v}_j)$. It follows from (10.1) and (10.2) that

$$d_E^2(\mathbf{v}_i, \mathbf{v}_j) = 4 \sum_{l=0}^{n-1} (v_{il} - v_{jl})^2. \tag{10.3}$$

Let $d_H(\mathbf{v}_i, \mathbf{v}_j)$ denote the Hamming distance between $\mathbf{v}_i$ and $\mathbf{v}_j$. It follows from (10.3) that we have the following relationship between the squared Euclidean distance and the Hamming distance between $\mathbf{v}_i$ and $\mathbf{v}_j$:

$$d_E^2(\mathbf{v}_i, \mathbf{v}_j) = 4 d_H(\mathbf{v}_i, \mathbf{v}_j). \tag{10.4}$$

The minimum squared Euclidean distance of code C is then defined as follows:

$$d_{E,min}^2(C) \triangleq \min\{d_E^2(\mathbf{v}_i, \mathbf{v}_j) : \mathbf{v}_i, \mathbf{v}_j \in C \text{ and } \mathbf{v}_i \neq \mathbf{v}_j\}. \tag{10.5}$$

From (10.4) and (10.5), we readily see that

$$d_{E,min}^2(C) = 4 d_{min}(C). \tag{10.6}$$

Consider a soft-decision MLD with the log-likelihood function as the decoding metric. Let $\mathbf{r} = (r_0, r_1, \cdots, r_{n-1})$ be the soft-decision received sequence. The log-likelihood function of $\mathbf{r}$ given a codeword $\mathbf{v}$ is (defined in (1.10))

$$\log P(\mathbf{r}|\mathbf{v}) = \sum_{i=0}^{n-1} \log P(r_i|v_i). \tag{10.7}$$

With the log-likelihood function as the decoding metric, MLD is carried out as follows: the received sequence $\mathbf{r}$ is decoded into a codeword $\mathbf{v} = (v_0, v_1, \cdots, v_{n-1})$ for which the log-likelihood function $\log P(\mathbf{r}|\mathbf{v})$ given in (10.7) is maximized.

For an AWGN channel with two-sided PSD $N_0/2$, the conditional probability $P(\mathbb{r}|\mathbb{v})$ is given by

$$\begin{aligned} P(\mathbb{r}|\mathbb{v}) &= \frac{1}{(\pi N_0)^{-n/2}} \exp\left\{-\sum_{i=0}^{n-1}(r_i - (2v_i - 1))^2/N_0\right\} \\ &= \frac{1}{(\pi N_0)^{-n/2}} \exp\left\{-\sum_{i=0}^{n-1}(r_i - c_i)^2/N_0\right\}. \end{aligned} \tag{10.8}$$

The sum $\sum_{i=0}^{n-1}(r_i - c_i)^2$ is simply the squared Euclidean distance between the received sequence $\mathbb{r}$ and the code signal sequence $(c_0, c_1, \cdots, c_{n-1})$. We denote this distance with $d_E^2(\mathbb{r}, \mathbb{c})$. From (10.8) we see that maximizing $P(\mathbb{r}|\mathbb{v})$ (or $\log P(\mathbb{r}|\mathbb{v})$) is equivalent to minimizing the squared Euclidean distance $d_E^2(\mathbb{r}, \mathbb{c})$. Consequently, soft-decision MLD can be carried out with the squared Euclidean distance as the decoding metric as follows: the received sequence $\mathbb{r}$ is decoded into a codeword $\mathbb{v}$ for which the squared Euclidean distance $d_E^2(\mathbb{r}, \mathbb{c})$ is minimized.

Consider the squared Euclidean distance

$$d_E^2(\mathbb{r}, \mathbb{c}) = \sum_{i=0}^{n-1}(r_i - c_i)^2. \tag{10.9}$$

Expanding the right-hand side of (10.9), we have

$$d_E^2(\mathbb{r}, \mathbb{c}) = \sum_{i=0}^{n-1} r_i^2 + n - 2\sum_{i=0}^{n-1} r_i \cdot c_i. \tag{10.10}$$

In computing $d_E^2(\mathbb{r}, \mathbb{c})$ for all codewords in MLD, we see that $\sum_{i=0}^{n-1} r_i^2$ is a common term, and n is a constant. From (10.10) we readily see that minimizing $d_E^2(\mathbb{r}, \mathbb{c})$ is equivalent to maximizing

$$m(\mathbb{r}, \mathbb{v}) = m(\mathbb{r}, \mathbb{c}) \triangleq \sum_{i=0}^{n-1} r_i \cdot c_i. \tag{10.11}$$

The preceding sum is called the *correlation* between the received sequence $\mathbb{r}$ and the code sequence $\mathbb{c}$. Therefore soft-decision MLD can be carried out in terms of the correlation metric as follows: $\mathbb{r}$ is decoded into a codeword $\mathbb{v}$ for which $m(\mathbb{r}, \mathbb{c})$ is maximized.

Finally, we can rewrite (10.11) as

$$m(\mathbb{r}, \mathbb{c}) = \sum_{i=0}^{n-1} |r_i| - 2 \sum_{i: r_i \cdot c_i < 0} |r_i|. \tag{10.12}$$

Let

$$\lambda(\mathbb{r}, \mathbb{v}) = \lambda(\mathbb{r}, \mathbb{c}) \triangleq \sum_{i: r_i \cdot c_i < 0} |r_i|. \tag{10.13}$$

Then,

$$m(\mathbf{r}, \mathbf{c}) = \sum_{i=0}^{n-1} |r_i| - 2\lambda(\mathbf{r}, \mathbf{c}). \tag{10.14}$$

Therefore, maximizing $m(\mathbf{r}, \mathbf{c})$ is equivalent to minimizing $\lambda(\mathbf{r}, \mathbf{c})$. Because $\sum_{i=0}^{n-1} |r_i|$ is the largest correlation that can be attained for a given $\mathbf{r}$, we call $\lambda(\mathbf{r}, \mathbf{c})$ the *correlation discrepancy* between $\mathbf{r}$ and $\mathbf{c}$. Soft-decision MLD can be carried out in terms of the discrepancy metric as follows: $\mathbf{r}$ is decoded into a codeword $\mathbf{v}$ for which $\lambda(\mathbf{r}, \mathbf{c})$ is minimized.

Let R and $\{A_0, A_1, \cdots, A_n\}$ be the rate and weight distribution of code C, respectively. Then, with soft-decision MLD, the probability of decoding a received sequence incorrectly, called *block* (or *word*) *error probability,* is upper bounded as follows [1, p. 440]:

$$P_B \leq \sum_{i=1}^{n} A_i Q\left(\sqrt{2iRE_b/N_0}\right), \tag{10.15}$$

where E_b is the received energy per information bit, and

$$Q(\alpha) = \frac{1}{\sqrt{2\pi}} \int_{\alpha}^{\infty} e^{-x^2/2}\, dx \tag{10.16}$$

is called the *Q-function.* This upper bound is known as the *union bound.* If the weight distribution of C is known, we can evaluate this bound.

Because the minimum Hamming distance of C is d_{min}, $A_1 = A_2 = \cdots = A_{d_{min}-1} = 0$. Since $Q(\alpha)$ decreases exponentially with α, $Q(\sqrt{2d_{min}RE_b/N_0})$ is the dominant term in the summation of (10.15). Therefore, for large SNR, E_b/N_0, the first term in the summation of (10.15),

$$A_{d_{min}} Q\left(\sqrt{2d_{min}RE_b/N_0}\right), \tag{10.17}$$

gives a good approximation of the upper bound. $A_{d_{min}}$ is commonly called the *error coefficient.* For large E_b/N_0, the union bound given by (10.15) is quite tight.

For $1 \leq i \leq n$, let B_i denote the average number of nonzero information bits associated with a codeword of weight i. Then, the bit-error probability (the probability that an information bit is decoded incorrectly) is upper bounded by

$$P_b \leq \frac{1}{k} \sum_{i=1}^{n} A_i B_i Q\left(\sqrt{2iRE_b/N_0}\right). \tag{10.18}$$

Soft-decision MLD achieves optimum error performance; however, it is very hard to implement and requires large computational complexity, especially for long codes. A brute-force method requires 2^k computations of metrics for 2^k codewords, and $2^k - 1$ comparisons to find the most likely codeword. For large k, the number of computations and comparisons is simply too large for practical implementation. Several soft-decision MLD algorithms have been devised. They are very efficient for codes of short to moderate lengths. For long codes, they still require very large computational and storage complexities. To overcome this complexity problem, various nonoptimum or suboptimum soft-decision decoding algorithms have been

devised, and they provide a good trade-off between error performance and decoding complexity.

Soft-decision decoding algorithms for codes, block or convolutional, can be classified into two categories: code structure-based decoding algorithms and reliability-based (or probabilistic) decoding algorithms. The most well known structure-based decoding algorithm is the *Viterbi algorithm* (VA) [3, 4], which is devised based on the trellis representation of a code to achieve MLD with a significant reduction in computational complexity. It was first devised for decoding convolutional codes; however since block codes also have trellis structure (as shown in Chapter 9), the Viterbi decoding algorithm applies to these codes as well. Besides the VA, there are other trellis-based decoding algorithms for both block and convolutional codes.

Consider a MLD based on minimizing the squared Euclidean distance between the received sequence $\mathbf{r}$ and a codeword in C. It follows from (10.6) that if the received sequence $\mathbf{r}$ falls inside a hypersphere of radius $\sqrt{d_{min}}$ centered at the transmitted code sequence $\mathbf{c}$ in the n-dimensional Euclidean space, then decoding is correct. A soft-decision decoding algorithm that guarantees correct decoding for all received sequences located inside the 2^k hyperspheres of radius $\sqrt{d_{min}}$ centered at each of the 2^k code sequences is called a *bounded distance decoding algorithm*. Bounded distance decoding has long been regarded as a criterion for designing good suboptimum decoding algorithms because its asymptotic error performance is the same as that of MLD, as deduced from (10.17); however, recent results tend to indicate that although bounded distance decoding algorithms perform well for codes with a relatively small Hamming distance d_{min} (say $d_{min} \leq 12$), this is not always the case for more powerful codes, mostly due to the significant increase of the corresponding error coefficients.

For large E_b/N_0, P_b is well approximated based on (10.18) by

$$P_b \approx \frac{1}{k} A_{d_{min}} B_{d_{min}} Q\left(\sqrt{2d_{min} R E_b/N_0}\right). \tag{10.19}$$

If encoding is realized in systematic form, we can choose $B_i/k \approx i/n$, since information bits are determined independently based on the systematic form of the encoder. In this case, (10.19) becomes

$$P_b \approx \frac{d_{min}}{n} A_{d_{min}} Q\left(\sqrt{2d_{min} R E_b/N_0}\right); \tag{10.20}$$

however, this is no longer the case for other types of encodings, such as encoding based on the trellis-oriented generator matrix $\mathbb{G}_{TOGM}$ introduced in Chapter 9. In that case, $B_i/k > i/n$ in general [2]. Consequently, it follows from (10.20) that for large E_b/N_0, encoding in nonsystematic form becomes suboptimum with respect to encoding in systematic form for the bit-error probability associated with MLD. This problem can easily be overcome based on the fact that a maximum likelihood decoder provides the most likely (ML) codeword out of the 2^k candidate codewords of a codebook, independent of the mapping used between information bits and codewords. As a result, the decoder can be designed based on a particular generator matrix $\mathbb{G}$, although any generator matrix $\mathbb{G}'$ defining the same 2^k codewords as $\mathbb{G}$ can be used for encoding. In particular, $\mathbb{G}$ can be transformed by row additions

only into the matrix $\mathbf{G}' = \mathbf{G}_{REF}$, so that $\mathbf{G}_{REF}$ contains the k columns of an identity matrix, but not necessarily in the first k positions. This matrix $\mathbf{G}_{REF}$ is said to be in *reduced echelon form (REF)*. If $\mathbf{G}_{REF}$ is used for encoding, the information sequence is easily retrieved from the codeword delivered by the decoder based on the knowledge of the positions of the k columns of the identity matrix in $\mathbf{G}_{REF}$. An example that illustrates this simple operation will be given in Section 10.5.

10.2 RELIABILITY MEASURES AND GENERAL RELIABILITY-BASED DECODING SCHEMES

Let $\mathbf{r} = (r_0, r_1, \cdots, r_{n-1})$ be a soft-decision received sequence at the output of the matched filter of the receiver. For $0 \leq i \leq n-1$, suppose each received symbol r_i is decoded independently based on the following hard-decision rule:

$$z_i = \begin{cases} 0 & \text{for } r_i < 0, \\ 1 & \text{for } r_i \geq 0. \end{cases} \tag{10.21}$$

Then, the magnitude $|r_i|$ of r_i can be used as a reliability measure of the hard-decision decoded bit z_i, since the magnitude of the log-likelihood ratio

$$|\log(P(r_i|v_i = 1)/P(r_i|v_i = 0))|$$

associated with the foregoing hard decision is proportional to $|r_i|$, where v_i is the ith transmitted bit. The larger the $|r_i|$, the more reliable the hard decision z_i is.

Based on the given reliability measure, we can reorder the symbols of the received sequence $\mathbf{r}$ in decreasing order of reliability. As a result of this ordering, the received symbols and their corresponding hard decisions at the left-side positions of the reordered received sequence are more reliable than the received symbols and their corresponding hard decisions at the right-side positions of the reordered received sequence. With hard-decision, an error is more likely to be committed at a less reliable position than at a more reliable position. Figure 10.1 shows the number of errors occurring at each position after reordering 500,000 received sequences with 50 symbols for each sequence, in decreasing order of reliability for various values of SNR. From this figure we see that few hard-decision errors are recorded in the more reliable positions of the ordered sequences, whereas the number of hard-decision errors increases exponentially in the less reliable positions. Based on this error distribution phenomenon, soft-decision decoding algorithms using reliability measures of the received symbols have been devised. These decoding algorithms are thus known as *reliability-based decoding algorithms*.

Based on the reliability measures of the received symbols, the symbol positions of a received sequence can be divided into two groups according to a certain criterion: one group consists of the least reliable positions (LRPs), and the other group consists of the most reliable positions (MRPs). Then, decoding can be devised based on either processing of the LRPs or processing of the MRPs.

Let $\mathbf{z} = (z_0, z_1, \cdots, z_{n-1})$ be the hard-decision received sequence obtained from $\mathbf{r}$. Then, errors are more likely to occur in the LRPs, as shown in Figure 10.1, and the MRPs are likely to contain very few or no errors. The errors in the LRPs can be reduced or removed by modifying the hard-decision received vector in these positions. Let E be a set of error patterns with errors confined only in the LRPs.

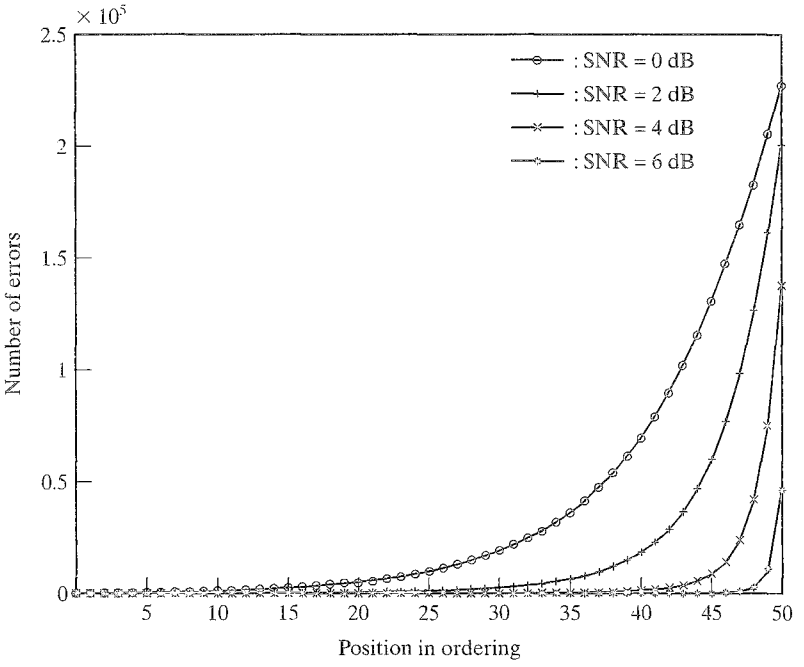

FIGURE 10.1: Number of errors at each ordered position for 500,000 BPSK sequences of length 50 with SNR values of 0 dB, 2 dB, 4 dB, and 6 dB.

For each error pattern $\mathbf{e}$ in E, we modify $\mathbf{z}$ by adding $\mathbf{e}$ to $\mathbf{z}$. Consequently, there are error patterns in E that reduce the number of errors at the LRPs of $\mathbf{z}$, and possibly, there is one error pattern in E that removes all the errors at the LRPs of $\mathbf{z}$. As a result, the modified vector $\mathbf{z} + \mathbf{e}$ very likely contains either no errors or a number of errors within the error-correcting capability t of the code. In that case, decoding $\mathbf{z} + \mathbf{e}$ with an algebraic decoder will give the ML codeword. Based on this concept, we can devise a general decoding scheme as follows:

1. Construct the error pattern set E based on the LRPs of $\mathbf{r}$.
2. For each error pattern $\mathbf{e}$ in E, form a modified received vector $\mathbf{z} + \mathbf{e}$.
3. Decode $\mathbf{z} + \mathbf{e}$ into a codeword in C with an algebraic decoder (decoding may fail if the number of errors in $\mathbf{z}+\mathbf{e}$ is greater then the error-correcting capability t of the code).
4. Steps 2 and 3 result in a list of candidate codewords. Compute the soft-decision decoding metrics of these candidates and find the one with the largest metric if correlation is used (or the smallest metric if squared Euclidean distance or correlation discrepancy is used), which is the decoded codeword.

The complexity of this general decoding scheme depends on the size of E and the complexity of the algebraic decoder for generating candidate codewords. The error performance of this scheme depends on how many LRPs are chosen to construct the error pattern set. Various decoding algorithms based on this general

scheme have been devised. These algorithms are referred to as *LRP-reprocessing algorithms*.

The second type of reliability-based decoding scheme processes the MRPs of $\mathbf{r}$. This decoding approach is based on the fact that a set of k independent positions in $\mathbf{z}$ uniquely determines a codeword in an (n, k) linear code. Based on the reliability measures of the received symbols, we determine a set of k most reliable independent positions (MRIPs) in $\mathbf{r}$. Then, the hard-decision received vector $\mathbf{z}$ contains only very few errors in these k MRIPs. Let $\mathbf{z}_k$ denote the k-tuple that consists of the symbols of $\mathbf{z}$ at the k MRIPs. Let E be a set of low-weight error patterns of length k. For each error pattern $\mathbf{e}$ in E, we form $\mathbf{z}_k + \mathbf{e}$ and encode it into a codeword in C. If $\mathbf{z}_k + \mathbf{e}$ contains no errors, the resultant codeword is the ML codeword. A general decoding scheme based on this concept follows:

1. Determine the error pattern set E based on the k MRIPs of $\mathbf{r}$.
2. For each error pattern $\mathbf{e}$ in E, encode $\mathbf{z}_k + \mathbf{e}$ into a codeword in C.
3. Step 2 generates a list of candidate codewords. Compute the soft-decision decoding metrics of these candidate codewords. Choose the one with the largest metric as the decoded codeword if correlation is used.

The error performance and complexity of this general decoding scheme depend on the choice of the error pattern set E. This general decoding scheme is referred to as the *MRIP-reprocessing decoding algorithm*. Various such decoding algorithms have been devised.

In the rest of this chapter, some important LRP-reprocessing and MRIP-reprocessing decoding algorithms are presented for binary linear block codes. In general, for nonbinary codes and multilevel modulation schemes, reliability measures are not as easily expressed as for BPSK transmission of binary codewords, and likelihood ratio metrics have to be considered. For a linear code over $GF(q)$, these metrics depend on q, the modulation used, and the channel model and may also be simplified in conjunction with the algorithm considered. Such considerations can be found in [5–7] for the AWGN channel with nonbinary error-correcting codes.

10.3 SUFFICIENT CONDITIONS ON THE OPTIMALITY OF A DECODED CODEWORD

Reliability-based decoding algorithms generally require generation of a list of candidate codewords of a predetermined size to restrict the search space for finding the ML codeword. These candidate codewords are usually generated serially one at a time. Each time a candidate codeword is generated, its metric is computed for comparison and final decoding. If a condition can be derived to test whether a generated candidate is the ML codeword, then decoding can be terminated whenever this condition is satisfied. This may result in an early termination of the decoding process without generation of the entire list of candidate codewords and hence reduces computations and delay. Such a condition is referred to as an *optimality condition*. We now desire a sufficient condition on optimality of a candidate codeword based on the reliability measures of the received symbols.

Let $\mathbf{v} = (v_0, v_1, \cdots, v_{n-1})$ be a codeword in C, and let $\mathbf{c} = (c_0, c_1, \cdots, c_{n-1})$ be its corresponding bipolar sequence using BPSK signaling, where $c_i = (2v_i - 1)$,

for $0 \leq i < n$. We define the following index sets for $\mathbb{v}$:

$$D_0(\mathbb{v}) \triangleq \{i : v_i = z_i \text{ with } 0 \leq i < n\}, \tag{10.22}$$

$$\begin{aligned} D_1(\mathbb{v}) &\triangleq \{i : v_i \neq z_i \text{ with } 0 \leq i < n\}, \\ &= \{0, 1, \cdots, n-1\} \backslash D_0(\mathbb{v}). \end{aligned} \tag{10.23}$$

Let

$$n(\mathbb{v}) = |D_1(\mathbb{v})|. \tag{10.24}$$

It follows from (10.1) and (10.21) that $r_i \cdot c_i < 0$ if and only if $z_i \neq v_i$. Consequently, we can express the correlation discrepancy $\lambda(\mathbb{r}, \mathbb{v})$ given by (10.13) as follows:

$$\lambda(\mathbb{r}, \mathbb{v}) = \sum_{i \in D_1(\mathbb{v})} |r_i|. \tag{10.25}$$

MLD based on correlation discrepancy is to find the codeword in C that has the smallest correlation discrepancy for a given received sequence $\mathbb{r}$. If there exists a codeword $\mathbb{v}^*$ for which

$$\lambda(\mathbb{r}, \mathbb{v}^*) \leq \alpha(\mathbb{r}, \mathbb{v}^*) \triangleq \min_{\mathbb{v} \in C, \mathbb{v} \neq \mathbb{v}^*} \{\lambda(\mathbb{r}, \mathbb{v})\}, \tag{10.26}$$

then $\mathbb{v}^*$ is the ML codeword for $\mathbb{r}$. It is not possible to determine $\alpha(\mathbb{r}, \mathbb{v}^*)$ without evaluating $\lambda(\mathbb{r}, \mathbb{v})$ for all $\mathbb{v}$ in C; however, if it is possible to determine a tight lower bound λ^* on $\alpha(\mathbb{r}, \mathbb{v}^*)$, then $\lambda(\mathbb{r}, \mathbb{v}) \leq \lambda^*$ represents a sufficient condition for the optimality of the candidate codeword $\mathbb{v}^*$ in the list of candidate codewords generated by a reliability-based decoding algorithm. We derive such a sufficient condition next.

If follows from (10.22) to (10.24) that the index set $D_0(\mathbb{v})$ consists of $n - n(\mathbb{v})$ indices. We order the indices in $D_0(\mathbb{v})$ based on the reliability measures of the received symbols as follows:

$$D_0(\mathbb{v}) = \{l_1, l_2, \cdots, l_{n-n(\mathbb{v})}\} \tag{10.27}$$

such that for $1 \leq i < j \leq n - n(\mathbb{v})$,

$$|r_{l_i}| < |r_{l_j}|. \tag{10.28}$$

Let $D_0^{(j)}(\mathbb{v})$ denote the set of first j indices in the ordered set $D_0(\mathbb{v})$; that is,

$$D_0^{(j)}(\mathbb{v}) = \{l_1, l_2, \cdots, l_j\}. \tag{10.29}$$

For $j \leq 0$, $D_0^{(j)}(\mathbb{v}) \triangleq \emptyset$, and for $j \geq n - n(\mathbb{v})$, $D_0^{(j)}(\mathbb{v}) \triangleq D_0(\mathbb{v})$.

For $1 \leq j \leq m$, let w_j be the jth nonzero weight in the weight profile $W = \{0, w_1, w_2, \cdots, w_m\}$ of code C, where $w_1 = d_{\min}(C)$, and $w_1 < w_2 < \cdots < w_m$. For a codeword $\mathbb{v}$ in C, we define

$$q_j \triangleq w_j - n(\mathbb{v}), \tag{10.30}$$

$$G(\mathbb{v}, w_j) \triangleq \sum_{i \in D_0^{(q_j)}(\mathbb{v})} |r_i|, \tag{10.31}$$

and

$$R(\mathbf{v}, w_j) \triangleq \{\mathbf{v}' \in C : d_H(\mathbf{v}', \mathbf{v}) < w_j\} \tag{10.32}$$

where $d_H(\mathbf{v}', \mathbf{v})$ denotes the Hamming distance between $\mathbf{v}'$ and $\mathbf{v}$. The set $R(\mathbf{v}, w_j)$ is the set of codewords in C that are at Hamming distances w_{j-1} or less from $\mathbf{v}$. For $j = 1$, $R(\mathbf{v}, w_1) = \{\mathbf{v}\}$, and for $j = 2$, $R(\mathbf{v}, w_2)$ contains $\mathbf{v}$ and all the codewords in C that are at a distance $d_{min}(C) = w_1$ from $\mathbf{v}$. $R(\mathbf{v}, w_2)$ is called the *minimum distance region* centered at $\mathbf{v}$. The following theorem gives a sufficient condition for $R(\mathbf{v}, w_j)$ to contain the ML codeword $\mathbf{v}_{ML}$ for a given received sequence $\mathbf{r}$.

Theorem 10.1 For a codeword $\mathbf{v} \in C$ and a nonzero weight $w_j \in W$, if the correlation discrepancy $\lambda(\mathbf{r}, \mathbf{v})$ of $\mathbf{v}$ with respect to a received sequence $\mathbf{r}$ satisfies the condition

$$\lambda(\mathbf{r}, \mathbf{v}) \leq G(\mathbf{v}, w_j), \tag{10.33}$$

then the ML codeword $\mathbf{v}_{ML}$ for $\mathbf{r}$ is contained in the region $R(\mathbf{v}, w_j)$.

Proof. Let $\mathbf{v}'$ be a codeword outside the region $R(\mathbf{v}, w_j)$; that is,

$$\mathbf{v}' \in C \backslash R(\mathbf{v}, w_j).$$

Then,

$$d_H(\mathbf{v}, \mathbf{v}') \geq w_j. \tag{10.34}$$

To prove the theorem, we need only to prove that $\lambda(\mathbf{r}, \mathbf{v}') \geq \lambda(\mathbf{r}, \mathbf{v})$. This would imply that no codeword outside the region $R(\mathbf{v}, w_j)$ is more likely than $\mathbf{v}$. In that case, $\mathbf{v}_{ML}$ must be in $R(\mathbf{v}, w_j)$.

Consider the sets $D_0(\mathbf{v}) \cap D_1(\mathbf{v}')$ and $D_1(\mathbf{v}) \cap D_0(\mathbf{v}')$. We define

$$n_{01} \triangleq |D_0(\mathbf{v}) \cap D_1(\mathbf{v}')|, \tag{10.35}$$

$$n_{10} \triangleq |D_1(\mathbf{v}) \cap D_0(\mathbf{v}')|. \tag{10.36}$$

Then, it follows from (10.22), (10.23), and (10.34) to (10.36) that

$$d_H(\mathbf{v}, \mathbf{v}') = n_{01} + n_{10} \geq w_j. \tag{10.37}$$

From (10.35) and (10.37), we readily find that

$$|D_1(\mathbf{v}')| \geq |D_0(\mathbf{v}) \cap D_1(\mathbf{v}')| \geq w_j - n_{10}. \tag{10.38}$$

Because $n_{10} = |D_1(\mathbf{v}) \cap D_0(\mathbf{v}')| \leq |D_1(\mathbf{v})|$, it follows from (10.38) and (10.24) that

$$\begin{aligned} |D_1(\mathbf{v}')| &\geq |D_0(\mathbf{v}) \cap D_1(\mathbf{v}')| \\ &\geq w_j - n_{10} \\ &\geq w_j - |D_1(\mathbf{v})| \\ &= w_j - n(\mathbf{v}). \end{aligned} \tag{10.39}$$

Note that $D_0(\mathbf{v}) \cap D_1(\mathbf{v}') \subseteq D_1(\mathbf{v}')$. From (10.25), (10.33), (10.39), and the definition of $D_0^{(j)}(\mathbf{v})$, we have

$$\begin{aligned}
\lambda(\mathbf{r}, \mathbf{v}') &= \sum_{i \in D_1(\mathbf{v}')} |r_i| \\
&\geq \sum_{i \in D_0(\mathbf{v}) \cap D_1(\mathbf{v}')} |r_i| \\
&\geq \sum_{i \in D_0^{(w_j - n(\mathbf{v}))}(\mathbf{v})} |r_i| \\
&= G(\mathbf{v}, w_j) \\
&\geq \lambda(\mathbf{r}, \mathbf{v}).
\end{aligned} \tag{10.40}$$

Therefore, $\lambda(\mathbf{r}, \mathbf{v}') \geq \lambda(\mathbf{r}, \mathbf{v})$, and no codeword $\mathbf{v}'$ outside the region $R(\mathbf{v}, w_j)$ is more likely than $\mathbf{v}$. Hence, $\mathbf{v}_{ML}$ must be in $R(\mathbf{v}, w_j)$. Q.E.D.

Given a codeword $\mathbf{v}$ in C, the condition (10.33) given in Theorem 10.1 simply defines a region in which the ML codeword $\mathbf{v}_{ML}$ can be found. It says that $\mathbf{v}_{ML}$ is among those codewords in C that are at a Hamming distance of w_{j-1} or less from the codeword $\mathbf{v}$. Two special cases are particularly important. For $j = 1$, $R(\mathbf{v}, w_1)$ contains only $\mathbf{v}$ itself. Therefore, the condition $\lambda(\mathbf{r}, \mathbf{v}) \leq G(\mathbf{v}, w_1)$ guarantees that $\mathbf{v}$ is the ML codeword $\mathbf{v}_{ML}$. For $j = 2$, $R(\mathbf{v}, w_2)$ contains $\mathbf{v}$ and its nearest neighbors (the codewords that are at minimum Hamming distance from $\mathbf{v}$). The following corollary summarizes these special results.

COROLLARY 10.1.1 Let $\mathbf{v}$ be a codeword in a binary linear (n, k) code C. Let $\mathbf{r}$ be a received sequence.

1. If $\lambda(\mathbf{r}, \mathbf{v}) \leq G(\mathbf{v}, w_1)$, then $\mathbf{v}$ is the ML codeword $\mathbf{v}_{ML}$ for $\mathbf{r}$.
2. If $\lambda(\mathbf{r}, \mathbf{v}) > G(\mathbf{v}, w_1)$ but $\lambda(\mathbf{r}, \mathbf{v}) \leq G(\mathbf{v}, w_2)$, then the ML codeword $\mathbf{v}_{ML}$ for $\mathbf{r}$ is at a Hamming distance not greater than the minimum Hamming distance $d_{min}(C) = w_1$ from $\mathbf{v}$. In this case, $\mathbf{v}_{ML}$ is either $\mathbf{v}$ or one of the nearest neighbors of $\mathbf{v}$.

The first part of Corollary 10.1.1 provides a sufficient condition for optimality of a codeword $\mathbf{v}$ and was first presented in [7]. This condition can be used as a stopping condition in any reliability-based decoding algorithm. The geometrical interpretation of this condition in the n-dimensional Euclidean space is depicted in Figure 10.2. In Figure 10.2, $s(\mathbf{x})$ represent the bipolar sequence associated with the binary n-tuple $\mathbf{x}$, which differs from $\mathbf{v}$ in all positions in $D_1(\mathbf{v})$ and in the $w_j - n(\mathbf{v})$ positions in $D_0^{(w_j - n(\mathbf{v}))}(\mathbf{v})$. This figure follows from (10.4), since no point contained in the n-dimensional hypersphere of squared Euclidean radius $4d_{min}(C)$ and centered at $\mathbf{c}$ is a valid code sequence, where $\mathbf{c}$ is the code sequence representing $\mathbf{v}$. The second part of the corollary allows us to search for the ML codeword $\mathbf{v}_{ML}$ in the minimum distance region centered at a candidate codeword. Both conditions when

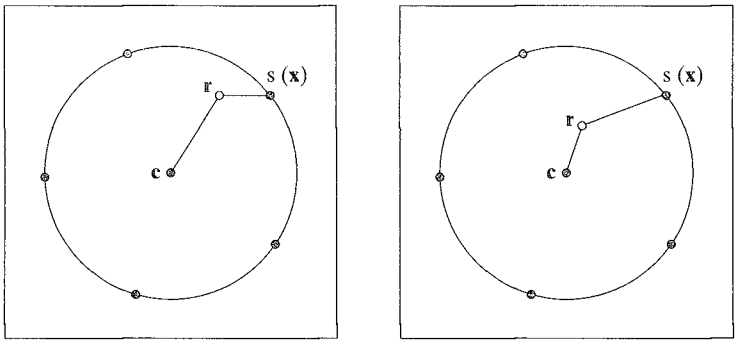

FIGURE 10.2: Sufficient condition for optimality of $\mathbf{c}$; (a) condition not satisfied: $d_E(\mathbf{r}, \mathbf{c}) > d_E(\mathbf{r}, s(\mathbf{x}))$; (b) condition satisfied: $d_E(\mathbf{r}, \mathbf{c}) \leq d_E(\mathbf{r}, s(\mathbf{x}))$.

used properly result in a significant reduction in average decoding computational complexity. This concept can straightforwardly be extended to the nonbinary case.

A sufficient condition for determining the optimality of a codeword based on two codewords can also be derived. Let $\mathbf{v}_1$ and $\mathbf{v}_2$ be two codewords in C. We define

$$\delta_1 \triangleq w_1 - n(\mathbf{v}_1), \tag{10.41}$$

$$\delta_2 \triangleq w_1 - n(\mathbf{v}_2), \tag{10.42}$$

$$D_{00} \triangleq D_0(\mathbf{v}_1) \cap D_0(\mathbf{v}_2), \tag{10.43}$$

$$D_{01} \triangleq D_0(\mathbf{v}_1) \cap D_1(\mathbf{v}_2). \tag{10.44}$$

Without loss of generality, we assume that $\delta_1 \geq \delta_2$. We order the indices in the preceding index sets according to the reliability values of the received symbols as defined by (10.27) and (10.28). We define

$$I(\mathbf{v}_1, \mathbf{v}_2) \triangleq \left(D_{00} \cup D_{01}^{(\lfloor(\delta_1 - \delta_2)/2\rfloor)}\right)^{(\delta_1)}, \tag{10.45}$$

where $X^{(q)}$ denotes the first q indices of an ordered index set X as defined in (10.29). We define

$$G(\mathbf{v}_1, w_1; \mathbf{v}_2, w_1) \triangleq \sum_{i \in I(\mathbf{v}_1, \mathbf{v}_2)} |r_i|. \tag{10.46}$$

Let $\mathbf{v}$ be either $\mathbf{v}_1$ or $\mathbf{v}_2$, whichever has the smaller correlation discrepancy. If

$$\lambda(\mathbf{r}, \mathbf{v}) \leq G(\mathbf{v}_1, w_1; \mathbf{v}_2, w_1), \tag{10.47}$$

then $\mathbf{v}$ is the ML codeword $\mathbf{v}_{ML}$ for $\mathbf{r}$. The derivation of (10.47) is left as a problem (see Problem 10.1).

The sufficient condition on optimality based on two codewords given by (10.47) is less stringent than the sufficient condition given in Corollary 10.1.1 for $j = 1$

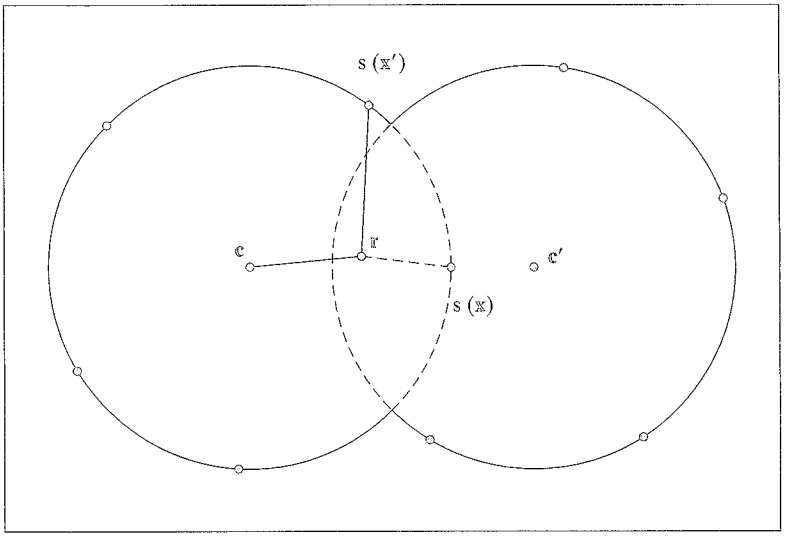

FIGURE 10.3: Sufficient condition for the optimality of $\mathbb{c}$ satisfied based on both $\mathbb{c}$ and $\mathbb{c}'$ but not satisfied based on $\mathbb{c}$ only; $d_E(\mathbb{r}, \mathbb{c}) \leq d_E(\mathbb{r}, s(\mathbb{x}'))$, but $d_E(\mathbb{r}, \mathbb{c}) > d_E(\mathbb{r}, s(\mathbb{x}))$.

based on a single codeword, which may result in a faster termination of the decoding process in a reliability-based decoding algorithm. The advantage of considering two codewords rather than one to derive a sufficient condition of optimality is illustrated in Figure 10.3. For any codeword $\mathbb{v}$ with corresponding signal sequence $\mathbb{c}$, no point contained in the n-dimensional hypersphere of squared Euclidean radius $4d_{min}(C)$ and centered at $\mathbb{c}$ is a valid candidate codeword. Therefore, if for two signal sequences $\mathbb{c}$ and $\mathbb{c}'$ representing two codewords $\mathbb{v}$ and $\mathbb{v}'$ processed by the algorithm, the corresponding hyperspheres intersect, then it is possible to eliminate binary n-tuples $\mathbb{x}$ based on the common knowledge of these two hyperspheres. Such a case is depicted in Figure 10.3, where the knowledge of the hypersphere associated with $\mathbb{c}'$ allows us to determine the optimality of the codeword $\mathbb{c}$.

10.4 GENERALIZED MINIMUM DISTANCE AND CHASE DECODING ALGORITHMS

In this section and the following ones, we present various decoding algorithms based on processing the LRPs of a received sequence. The first such algorithm is known as the *generalized minimum distance (GMD)* decoding algorithm devised by Forney [5] in 1966. This decoding approach was later generalized by Chase and others, mostly for binary linear block codes [9, 10].

10.4.1 The GMD Decoding Algorithm

The GMD decoding algorithm is a very simple and elegant method of using reliability information of the received symbols to improve algebraic decoding for both binary and nonbinary codes. For an (n, k) linear block code with minimum Hamming distance d_{min}, this decoding algorithm generates a list of at most $\lfloor (d_{min} + 1)/2 \rfloor$ candidate codewords based on an error-and-erasure algebraic decoding method

and then selects the most likely one from the generated codewords. As stated in Section 7.7, an error-and-erasure decoding method can correct all combinations of ν errors and e erasures provided that

$$2\nu + e \leq d_{min} - 1. \tag{10.48}$$

The GMD decoding algorithm simply considers all possible cases of erasures $e \leq d_{min} - 1$ in the $d_{min} - 1$ LRPs, which are the most likely positions to be in error. The decoding is carried out as follows:

1. Form the hard-decision received sequence $\mathbf{z}$ from $\mathbf{r}$ and assign a reliability value to each symbol of $\mathbf{z}$.
2. Generate a list of $\lfloor (d_{min} + 1)/2 \rfloor$ sequences by modifying the hard-decision received sequence $\mathbf{z}$. If d_{min} is even, modify $\mathbf{z}$ by erasing the least reliable symbol, the three least reliable symbols, $\cdots$, the $d_{min} - 1$ least reliable symbols. If d_{min} is odd, modify $\mathbf{z}$ by erasing no symbol, the two least reliable symbols, $\cdots$, the $d_{min} - 1$ least reliable symbols.
3. Decode each modified $\mathbf{z}$ into a codeword $\mathbf{v}$ using an error-and-erasure algebraic decoding algorithm.
4. Compute the soft-decision decoding metric for each generated candidate codeword. Select the candidate codeword with the best metric as the decoded solution.

Steps 2 and 3 can be executed simultaneously. In most cases, fewer than $\lfloor (d_{\min} + 1)/2 \rfloor$ candidate codewords are generated by this procedure, since some algebraic decodings at step 3 may fail and result in no candidate codewords. Also, a sufficient condition on optimality can be used for testing each candidate codeword when it is generated. Decoding is terminated whenever a generated candidate codeword satisfies the optimality condition. For decoding RS codes, the Euclidean error-and-erasure decoding algorithm presented in Section 7.7 can be used.

10.4.2 Chase Decoding Algorithms

In generalizing the GMD decoding approach, Chase devised three algorithms [9], namely, algorithm-1, algorithm-2, and algorithm-3. For decoding binary linear block codes, Chase algorithm-3 is very similar to the GMD decoding, except that the erasure operation is replaced by the complementation operation at the LRPs of the hard-decision received sequence $\mathbf{z}$ (i.e., changing a 1 into a 0, or a 0 into a 1). An error-correction-only algebraic decoder is used to generate a list of at most $\lfloor d_{min}/2 + 1 \rfloor$ candidate codewords. Chase algorithm-3 performs the following steps in decoding:

1. Form the hard-decision received sequence $\mathbf{z}$ from $\mathbf{r}$ and assign a reliability value to each symbol of $\mathbf{z}$.
2. Generate a list of at most $\lfloor d_{min}/2 + 1 \rfloor$ sequences by modifying the hard-decision received sequence $\mathbf{z}$. If d_{min} is even, modify $\mathbf{z}$ by complementing no position, the least reliable bit, the three least reliable bits, $\cdots$, the $d_{min} - 1$ least reliable bits. If d_{min} is odd, modify $\mathbf{z}$ by complementing no symbol, the two least reliable bits, the four least reliable bits, $\cdots$, the $d_{min} - 1$ least reliable bits.

3. Decode each modified $\mathbf{z}$ into a codeword $\mathbf{v}$ using an error-correction-only algebraic decoder.
4. Compute the soft-decision decoding metric for each generated candidate codeword. Select the candidate codeword with the best metric as the decoded solution.

From step 2 we see that for even d_{min}, error patterns of odd weights confined to the $d_{min} - 1$ LRPs are added to $\mathbf{z}$ for decoding, and for odd d_{min}, error patterns of even weights confined to the $d_{min} - 1$ LRPs are added to $\mathbf{z}$ for decoding. Because algebraic decoding of a modified $\mathbf{z}$ may fail, at most $\lfloor d_{min}/2 + 1 \rfloor$ candidate codewords can be generated. If a condition on optimality is used for testing each candidate codeword when it is generated, it may shorten the decoding process. For binary codes, Chase algorithm-3 achieves about the same error performance as the GMD decoding algorithm and requires about the same computational complexity.

Chase algorithm-2 is an improvement of algorithm-3; it generates a larger list of candidate codewords for decoding. In this algorithm, a set E of error patterns with all possible errors confined to the $\lfloor d_{min}/2 \rfloor$ LRPs of $\mathbf{z}$ is used to modify $\mathbf{z}$. This algorithm performs the following decoding steps:

1. Form the hard-decision received sequence $\mathbf{z}$ from $\mathbf{r}$ and assign a reliability value to each symbol of $\mathbf{z}$.
2. Generate the error patterns in E one at a time, possibly in likelihood order. For each error pattern $\mathbf{e}$, form the modified vector $\mathbf{z} + \mathbf{e}$.
3. Decode each $\mathbf{z} + \mathbf{e}$ into a codeword $\mathbf{v}$ using an error-correction-only algebraic decoder.
4. Compute the soft-decision decoding metric for each generated candidate codeword. Select the candidate codeword with the best metric as the decoded solution.

The set E is called the *test error pattern set*, and it consists of the $2^{\lfloor d_{min}/2 \rfloor}$ test error patterns obtained by considering all possible combinations of 0's and 1's in the $\lfloor d_{min}/2 \rfloor$ LRPs of $\mathbf{z}$. Therefore, the candidate codeword list for Chase algorithm-2 grows exponentially with d_{min} and hence is much larger than the candidate codeword list for algorithm-3 when d_{min} becomes large. The result is a greater computational complexity; however, algorithm-2 achieves a much better error performance than algorithm-3. A condition on optimality can be incorporated in Chase algorithm-2 to reduce the number of algebraic decodings.

Chase algorithm-1 generates a list of at most $\binom{n}{\lfloor d_{min}/2 \rfloor}$ candidate codewords by complementing all possible combinations of exactly $\lfloor d_{min}/2 \rfloor$ positions in the hard-decision received sequence $\mathbf{z}$. Owing to its very large computational complexity compared with other soft-decision decoding methods, Chase algorithm-1 has never received much attention. Among the three Chase algorithms, algorithm-2 gives the best trade-off between error performance and decoding complexity.

10.4.3 Generalized Chase and GMD Decoding Algorithms

Chase algorithms 2 and 3 can be generalized into a family of decoding algorithms that contains Chase algorithms 2 and 3 as the two extremes. For $a = 1, 2, \cdots, \lceil d_{min}/2 \rceil$,

algorithm $A(a)$ of this family generates at most $2^{a-1}(\lceil (d_{min}+1)/2 \rceil - a + 1)$ candidate codewords. For a chosen a, we form a set $E(a)$ of error patterns by modifying the hard-decision received sequence $\mathbf{z}$. For even d_{min}, $E(a)$ consists of the following error patterns:

1. The error patterns with errors confined to the $a-1$ LRPs. There are 2^{a-1} such error patterns.
2. For each of the preceding error patterns, the i next LRPs, with $i = 0, 1, 3, \cdots, d_{min} - 2a + 1$, are complemented.

For odd $d_{\min}$, $E(a)$ consists of the following error patterns:

1. The error patterns with errors confined to the $a-1$ LRPs.
2. For each of the preceding error patterns, the i next LRPs, with $i = 0, 2, 4, \cdots, d_{min} - 2a + 1$, are complemented.

Then, algorithm $A(a)$ executes the following steps in decoding:

1. Form the hard-decision received sequence $\mathbf{z}$ from $\mathbf{r}$ and assign a reliability value to each symbol of $\mathbf{z}$.
2. Generate the error patterns in $E(a)$ one at a time, possibly in likelihood order. For each error pattern $\mathbf{e}$, form the modified vector $\mathbf{z} + \mathbf{e}$.
3. Decode each $\mathbf{z} + \mathbf{e}$ into a codeword $\mathbf{v}$ using an error-correction-only algebraic decoder.
4. Compute the soft-decision decoding metric for each generated candidate codeword. Select the candidate codeword with the best metric as the decoded solution.

We readily see that $A(1)$ and $A(\lceil d_{min}/2 \rceil)$ are simply Chase algorithm-3 and Chase algorithm-2, respectively. It has been shown that for BPSK signaling, algorithm $A(a)$ with $a = 1, 2, \cdots, \lceil d_{min}/2 \rceil$ achieves bounded distance decoding [10].

Another family of $\lceil d_{min}/2 \rceil$ decoding algorithms, denoted by $\{A_e(a) : a = 1, \cdots, \lceil d_{min}/2 \rceil\}$, can be obtained by modifying the foregoing error pattern set $E(a)$. Let $E_e(a)$ denote the test error pattern set for algorithm $A_e(a)$. The set $E_e(a)$ is very similar to $E(a)$ after the complementation operation is replaced with the erasure operation. For even d_{min}, $E_e(a)$ consists of the following error patterns:

1. The error patterns with errors confined to the $a-1$ LRPs.
2. For each of the preceding error patterns, the i next LRPs, with $i = 1, 3, \cdots, d_{min} - 2a + 1$, are erased.

For odd d_{min}, $E_e(a)$ consists of the following error patterns:

1. The error patterns with errors confined to the $a-1$ LRPs.
2. For each of the preceding error patterns, the i next LRPs, with $i = 0, 2, 4, \cdots, d_{min} - 2a + 1$, are erased.

Decoding with algorithm $A_e(a)$ carries out the same steps as decoding with algorithm $A(a)$, except that an error-and-erasure algebraic decoder is used at step 3 to generate candidate codewords. For $a = 1, \cdots, \lceil d_{min}/2 \rceil$, the algorithm $A_e(a)$ generates at most $2^{a-1}(\lceil d_{min}/2 \rceil - a + 1)$ candidate codewords. For $a = 1$, $A_e(1)$ is simply the basic GMD decoding algorithm. Therefore, the family of algorithms $\{A_e(a)\}$ is a generalization of the GMD algorithm. The algorithms $A_e(\lceil d_{min}/2 \rceil)$ and $A(\lceil d_{min}/2 \rceil)$ can be shown to be equivalent, since for even weight codes, a given position among the $\lfloor d_{min}/2 \rfloor$ LRPs can simply be erased in Chase algorithm-2 (see Problem 10.4).

For $a = 1, \cdots, \lceil d_{min}/2 \rceil$, both families of decoding algorithms, $\{A(a)\}$ and $\{A_e(a)\}$, achieve bounded distance decoding [10]. The error performance and computational complexity of these two families of decoding algorithms depend on the choice of the parameter a: the larger the parameter a, the better the error performance, and the larger the decoding computational complexity. Therefore, these two families of bounded distance decoding algorithms provide a wide range of trade-offs between the error performance and the decoding computational complexity. For BPSK signaling and a given a, algorithm $A(a)$ slightly outperforms algorithm $A_e(a)$. Therefore, the choice of $A_e(a)$ or $A(a)$ for error control is mainly determined by implementation considerations.

Figure 10.4 shows the bit-error performances of the (64, 42, 8) RM code using decoding algorithms $A(a)$, for $a = 1, 2$, and 4. Algorithms $A(1)$, $A(2)$, and

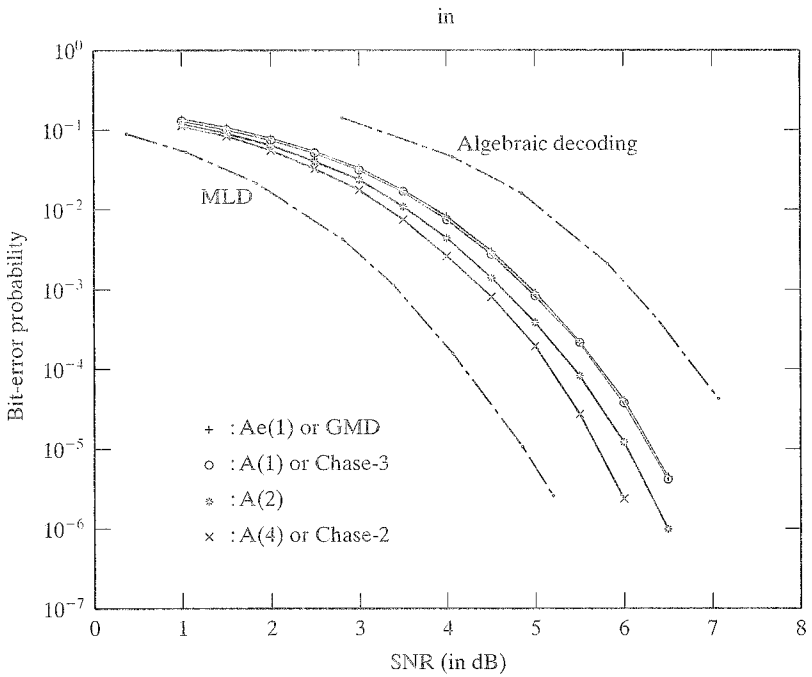

FIGURE 10.4: Bit-error performances of the (64, 42, 8) RM code with $A(1)$, $A(2)$, and $A(4)$ decoding algorithms.

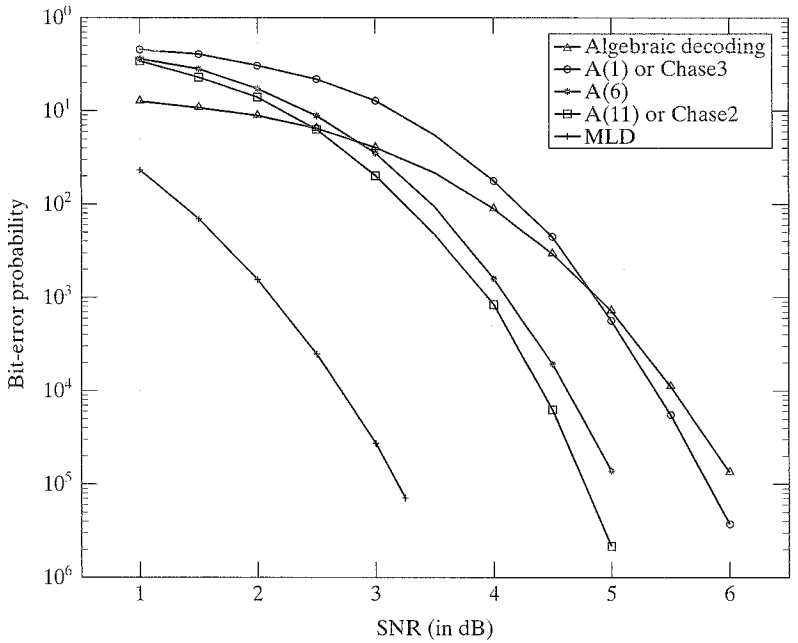

FIGURE 10.5: Bit-error performances of the (127, 64, 21) BCH code with $A(1)$, $A(6)$, and $A(11)$ decoding algorithms.

$A(4)$ require to generate and process at most 5, 8, and 16 candidate codewords, respectively. The Reed algorithm (see Section 4.3) is used to decode the (64, 42, 8) RM code. In this figure, the error performances of the algebraic, GMD (or $A_e(1)$), and MLD decodings of the code are also included for comparison. We see that GMD and $A(1)$ achieve the same error performance with the least computational complexity (processing only 4 candidate codewords). They achieve 1.1-dB coding gain over the algebraic Reed decoding algorithm at the BER 10^{-4}. Algorithms $A(2)$ and $A(4)$ give better error performance than the GMD and $A(1)$ decoding algorithms and require to process 2 and 4 more candidate codewords, respectively. Algorithm $A(4)$ achieves 1.6-dB coding gain over the algebraic decoding of the code at the BER 10^{-4} but has a 0.9-dB degradation in coding gain compared with the MLD of the code. Figure 10.5 shows the bit-error performances of the of the (127, 64, 21) BCH code with decoding algorithms $A(1)$, $A(6)$, and $A(11)$. The Berlekamp algorithm is used in decoding the (127, 64, 21) BCH code. Algorithms $A(1)$, $A(6)$, and $A(11)$ require to generate and process at most 11, 192, and 1024 candidate codewords, respectively. We see that $A(1)$ (also GMD) performs better than the algebraic decoding only for SNR values greater than 4.7 dB. Algorithm $A(11)$ (or Chase algorithm-2) achieves 1.1-dB coding gain over algebraic decoding at the BER 10^{-4} but has a 1.7-dB loss in coding gain compared with MLD. In general, for a given a, the performance degradation of $A(a)$ (or $A_e(a)$) compared with MLD increases with the minimum Hamming distance d_{min} of the code considered.

10.5 WEIGHTED ERASURE DECODING

The *weighted erasure decoding* (WED) algorithm is another variation of GMD for a binary input and Q-ary output channel. This algorithm was devised by Weldon for quantized channel outputs [11]. In this section, we present a simplified version of the WED algorithm.

Consider a binary linear (n, k) code with minimum distance $d_{\min}$ that is used for error control. Suppose the channel output is uniformly quantized into Q levels with $Q - 2$ granular regions of equal space and 2 overload regions at the two ends. For $Q = 2^m$, these Q regions are assigned Q w-weights $w_j = j/(Q-1)$, for $0 \le j \le Q-1$. For $0 \le i \le m-1$, we define

$$p_i = 2^{m-i-1}/(Q-1). \tag{10.49}$$

Then, $\sum_{i=0}^{m-1} p_i = 1$. It follows from the definition of w-weights and (10.49) that

$$w_j = \sum_{i=0}^{m-1} a_{j,i}\, p_i, \tag{10.50}$$

where the binary m-tuple $(a_{j,0}, a_{j,1}, \cdots, a_{j,m-1})$ is simply the binary representation of the integer j for $0 \le j \le Q-1$; that is,

$$j = \sum_{i=0}^{m-1} a_{j,i} 2^i. \tag{10.51}$$

Each received signal r_l, for $0 \le l \le n-1$, is quantized into one of the Q quantization levels and assigned the w-weight associated with that level. For $0 \le l \le n-1$, let w_l denote the w-weight of the received symbol r_l. We express w_l in the form of (10.50) as follows:

$$w_l = \sum_{i=0}^{m-1} a_{j,i}^{(l)} p_i. \tag{10.52}$$

We construct an $m \times n$ binary matrix $\mathbb{A}$ associated with the received sequence $\mathbb{r} = (r_0, r_1, \cdots, r_{n-1})$ for which the lth column is given by the binary m-tuple $(a_{j,0}^{(l)}, a_{j,1}^{(l)}, \cdots, a_{j,m-1}^{(l)})$ obtained from (10.52). Then, w_l is called the weight of the lth column of $\mathbb{A}$.

For $0 \le i \le m-1$, the ith row of $\mathbb{A}$ is decoded into a codeword in code C with an algebraic decoder. The m decoded codewords are then used in the same order to form another $m \times n$ binary array $\mathbb{A}'$. For $0 \le i \le m-1$, let f_i denote the number of places where the ith row of $\mathbb{A}$ and the ith row of $\mathbb{A}'$ differ. We define the reliability indicator of the ith row of $\mathbb{A}'$ as

$$R_i = \max\{0,\ d_{\min} - 2 f_i\}. \tag{10.53}$$

For the lth column of $\mathbb{A}'$, with $0 \le l \le n-1$, let S_1^l and S_0^l denote the index sets of the rows that contain 1 and 0 at bit position l, respectively. Then, the lth bit is decoded into 0 if

$$\sum_{i \in S_0^l} R_i\, p_i > \sum_{i \in S_1^l} R_i\, p_i \tag{10.54}$$

and decoded into 1 if

$$\sum_{i \in S_0'} R_i \, p_i \; < \; \sum_{i \in S_1'} R_i \, p_i. \tag{10.55}$$

In case of equality, the hard decision of r_l is taken to be the decoded bit.

Let N_j denotes the number of columns in $\mathbb{A}$ with weight w_j, for $0 \le j \le Q-1$. Then, the WED algorithm gives correct decoding if the following condition holds:

$$\sum_{j=0}^{Q-1} N_j w_j \; < \; \frac{d_{min}}{2}. \tag{10.56}$$

EXAMPLE 10.1

Consider the (8, 4, 4) RM code generated by

$$\mathbb{G} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

Suppose this code is decoded with the WED algorithm with 4-level output quantization and two granular regions of spacing $\Delta = 0.3$. The w-weights associated with the quantization levels are $w_0 = 0$, $w_1 = \frac{1}{3}$, $w_2 = \frac{2}{3}$, and $w_3 = 1$, respectively. We find that $p_0 = \frac{2}{3}$, and $p_1 = \frac{1}{3}$. It follows from (10.50) that we can represent the w-weights as follows:

$$\begin{aligned} w_0 &= 0 \cdot p_0 + 0 \cdot p_1, \\ w_1 &= 0 \cdot p_0 + 1 \cdot p_1, \\ w_2 &= 1 \cdot p_0 + 0 \cdot p_1, \\ w_3 &= 1 \cdot p_0 + 1 \cdot p_1. \end{aligned}$$

Therefore the four 2-tuples associated with the w-weights are (0, 0), (0, 1), (1, 0), and (1, 1).

The generator matrix $\mathbb{G}$ can be put in reduced echelon form (REF) by elementary row operations:

$$\mathbb{G}_{REF} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

From the last paragraph of Section 10.1, we know that encoding based on $\mathbb{G}_{REF}$ provides a smaller bit-error probability than encoding based on $\mathbb{G}$. Hence, suppose encoding is based on $\mathbb{G}_{REF}$. Then, the message $\mathbb{u} = (1, 0, 0, 0)$ is encoded into the codeword $\mathbb{v} = (1, 0, 0, 1, 0, 1, 1, 0)$. For BPSK transmission, $\mathbb{v}$ is mapped into the bipolar sequence $\mathbb{c} = (1, -1, -1, 1, -1, 1, 1, -1)$. Suppose this sequence is transmitted, and the received sequence is

$$\mathbb{r} = (\, 0.7, \; -2.1, \; -1.1, \; -0.4, \; -1.4, \; 1.7, \; 0.8, \; 0.1 \,).$$

If a hard decision is made for each received symbol, we obtain the following hard-decision received vector:

$$\mathbb{z} = (\,1,\ 0,\ 0,\ 0^*,\ 0,\ 1,\ 1,\ 1^*\,).$$

This hard-decision received sequence contains two errors (marked $*$) compared with the transmitted codeword $\mathbb{v}$. Because the error-correcting capability of the code is $t = 1$, correct decoding of $\mathbb{z}$ is not guaranteed with algebraic decoding. With WED, each symbol of $\mathbb{r}$ is assigned a specific weight in $\{w_0, w_1, w_2, w_3\}$ associated with the region that contains the symbol. Then, the w-weight sequence for $\mathbb{r}$ is

$$(\,w_3,\ w_0,\ w_0,\ w_0,\ w_0,\ w_3,\ w_3,\ w_2\,).$$

Based on the 2-tuple representation of each weight, we form the 2 x 8 binary array

$$\mathbb{A} = \begin{bmatrix} 1 & 0 & 0 & 0^* & 0 & 1 & 1 & 1^* \\ 1 & 0 & 0 & 0^* & 0 & 1 & 1 & 0 \end{bmatrix}.$$

The first row of $\mathbb{A}$ is equivalent to the hard decision of $\mathbb{r}$ with two positions in error. The second row of $\mathbb{A}$ contains only one error. The Reed decoding algorithm decodes the first row of $\mathbb{A}$ into information sequence $\mathbb{u}_1 = (0, 1, 0, 0)$, and the second row of $\mathbb{A}$ into information sequence $\mathbb{u}_2 = (1, 1, 1, 1)$. Because decoding assumes that encoding is based on the generator matrix $\mathbb{G}$ (Reed algorithm), the two decoded information sequences, $\mathbb{u}_1$ and $\mathbb{u}_2$, are re-encoded into two codewords based on $\mathbb{G}$. Using these two codewords as rows, we obtain the matrix

$$\mathbb{A}' = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}.$$

Comparing $\mathbb{A}$ and $\mathbb{A}'$, we obtain $f_0 = 2$ and $f_1 = 1$. It follows from (10.53) that the reliability indicators for the two rows of $\mathbb{A}'$ are $R_0 = 0$ and $R_1 = 2$. The index sets for the columns of $\mathbb{A}'$ are given in Table 10.1. To decode the code bit at position 0, we compute

$$\sum_{i \in S_0^0} R_i\, p_i = R_0\, p_0 = 0,$$

$$\sum_{i \in S_1^0} R_i\, p_i = R_1\, p_1 = 2 \cdot \frac{1}{3} = \frac{2}{3}.$$

TABLE 10.1: Index sets for the columns of $\mathbb{A}'$.

Column	0	1	2	3	4	5	6	7
S_0^l	{0}	{0, 1}	{0, 1}	{0}	{1}	$\emptyset$	$\emptyset$	{1}
S_1^l	{1}	$\emptyset$	$\emptyset$	{1}	{0}	{0, 1}	{0, 1}	{0}

Because $\sum_{i \in S_0^0} R_i\, p_i < \sum_{i \in S_1^0} R_i\, p_i$, the decoded bit at position 0 is $\hat{v}_0 = 1$. To decode the code bit at position 1, we compute

$$\sum_{i \in S_0^1} R_i\, p_i = R_0\, p_0 + R_1\, p_1 = 0 + 2 \cdot \tfrac{1}{3} = \tfrac{2}{3},$$

$$\sum_{i \in S_1^1} R_i\, p_i = 0.$$

Because $\sum_{i \in S_0^0} R_i\, p_i > \sum_{i \in S_1^0} R_i\, p_i$, the decoded bit at position 0 is $\hat{v}_1 = 0$. Repeating the foregoing decoding process, we obtain the following decoded sequence:

$$\hat{\mathbf{v}} = (1,\ 0,\ 0,\ 1,\ 0,\ 1,\ 1,\ 0),$$

which is identical to the transmitted codeword. Thus, the decoding is correct. The corresponding information sequence is easily retrieved based on $\mathbf{G}_{REF}$. This example illustrates that an error pattern not correctable by algebraic decoding can be corrected by the WED algorithm.

The error performance achieved by the WED algorithm strongly depends on the quantizer levels, which have to be chosen according to the operating SNR [12]. Nevertheless, the WED represents a very attractive solution for practical implementations of low-complexity reliability-based algorithms owing to its simplicity.

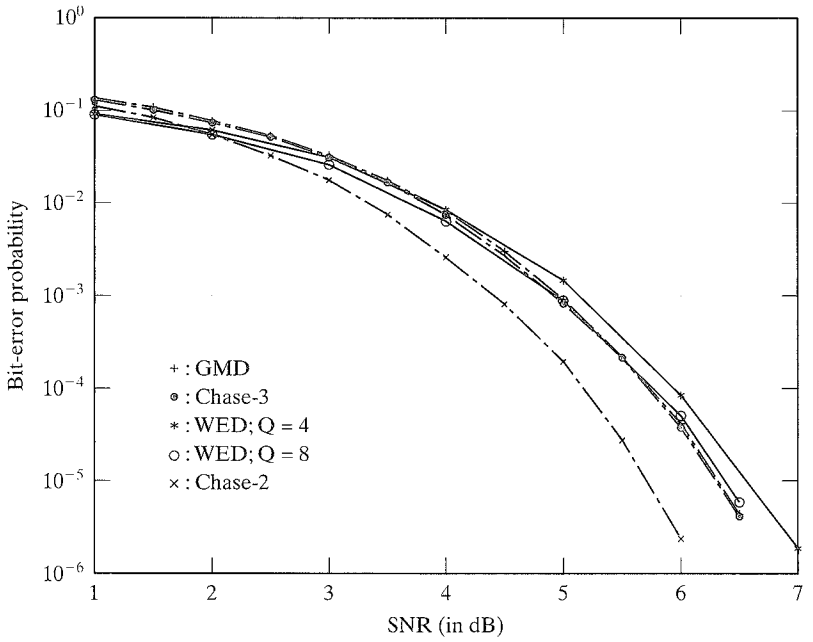

FIGURE 10.6: Comparisons between WED for $Q = 4$ and $Q = 8$ and GMD, Chase-3, and Chase-2 decodings for the (64, 42, 8) RM code.

Another interesting feature of the WED algorithm is that the number of LRPs modified varies with each received sequence, as opposed to GMD or Chase-type algorithms. This adaptive feature of Chase-type decoding algorithms has been investigated in [13]. Figure 10.6 compares WED for $Q = 4$ and $Q = 8$ with GMD, Chase-3, and Chase-2 decoding algorithms for the (64, 42, 8) RM code. We observe that WED for $Q = 8$ slightly outperforms GMD and Chase-3 decodings for all BER values larger than 10^{-3} owing to its implicit adaptive feature.

10.6 A MAXIMUM LIKELIHOOD DECODING ALGORITHM BASED ON ITERATIVE PROCESSING OF THE LEAST RELIABLE POSITIONS

In Chase decoding algorithm-2, the number of LRPs to be processed is limited to $\lfloor d_{\min}/2 \rfloor$ to reduce computational complexity while achieving bounded-distance decoding. As shown in Figures 10.4 and 10.5, this limitation results in a significant performance degradation compared with MLD, especially for long codes. Performance can be improved by increasing the number of LRPs to be processed. Unfortunately, this straightforward approach results in an exponentially increasing computational complexity with decreasing performance improvement. To reduce the computational complexity, an algorithm must be devised to reduce the search space dynamically as candidate codewords are generated. Such an algorithm has been devised by Kaneko, Nishijima, Inazumi, and Hirasawa [14]. For simplicity, we call this algorithm the KNIH algorithm. This algorithm processes the LRPs of the received sequence iteratively and uses a condition on the correlation discrepancy of the best candidate codeword generated either to guide the direction of further search or to terminate the decoding process when the most likely codeword is found. The KNIH algorithm achieves maximum likelihood decoding.

Let C be a binary linear (n, k) block code with minimum distance $d_{\min}$. Suppose a bounded-distance algebraic decoder is used to generate candidate codewords. The decoder is designed to correct error patterns of weights up to $t = \lfloor (d_{\min} - 1)/2 \rfloor$. Let $\mathbb{r} = (r_0, r_1, \cdots, r_{n-1})$ be the received sequence and $\mathbb{z}$ be its corresponding hard-decision sequence. Let $E(i)$ denote the set of error patterns with errors confined in the i LRPs of $\mathbb{z}$, and let $J(i)$ denote the set codewords obtained by decoding the sequence $\mathbb{z} + \mathbb{e}$ with $\mathbb{e} \in E(i)$. Then, it follows from the definition of $J(i)$ that

$$J(0) \subseteq J(1) \subseteq J(2) \cdots \subseteq J(n), \tag{10.57}$$

where $J(0)$ contains only one codeword obtained by decoding $\mathbb{z}$ if decoding is successful and is empty if decoding of $\mathbb{z}$ fails. For $0 \leq i \leq n-1$, consider a codeword $\mathbb{v}'$ in $J(i+1) \backslash J(i)$. First, $\mathbb{v}' \in J(i+1)$ implies that $\mathbb{v}'$ must have at most t positions j outside the $(i+1)$ LRPs such that $v_j' \neq z_j$. Next, $\mathbb{v}' \notin J(i)$ implies that $\mathbb{v}'$ contains at least $(t+1)$ positions j outside the i LRPs such that $v_j' \neq z_j$. Combining these two facts, it follows that $\mathbb{v}'$ has exactly $(i+1)$ positions j outside the i LRPs such that $v_j' \neq z_j$, and one such position is the $(i+1)$th LRP. Without loss of generality, assume that the received symbols r_j are ordered according to increasing reliability values; that is, r_0 is the least reliable symbol, and r_{n-1} is the most reliable symbol. It follows from the definition of correlation discrepancy given by (10.13) that the

correlation discrepancy of $\mathbf{v}'$ with respect to the received sequence $\mathbf{r}$ satisfies

$$\lambda(\mathbf{r}, \mathbf{v}') \geq \sum_{j=1}^{t+1} |r_{i+j}|. \tag{10.58}$$

Let $\mathbf{v}$ be the codeword in $J(i)$ that has the least correlation discrepancy with $\mathbf{r}$ (i.e., $\mathbf{v}$ is the most likely codeword in $J(i)$). Then, $d_H(\mathbf{v}, \mathbf{v}') \geq d_{\min}$. Let

$$\delta_t \triangleq \max\{0, d_{\min} - (t+1) - |D_1(\mathbf{v})|\}. \tag{10.59}$$

We define

$$G_i(\mathbf{v}) \triangleq \sum_{j=1}^{t+1} |r_{i+j}| + \sum_{j \in D_0^{(\delta_t)}(\mathbf{v})} |r_j|. \tag{10.60}$$

Based on the same approach as for Theorem 10.1, it can be shown that [14]

$$\lambda(\mathbf{r}, \mathbf{v}') \geq G_i(\mathbf{v}). \tag{10.61}$$

The preceding inequality simply says that any codeword outside of $J(i)$ cannot have a correlation discrepancy smaller than $G_i(\mathbf{v})$. Therefore, if

$$\lambda(\mathbf{r}, \mathbf{v}) < G_i(\mathbf{v}), \tag{10.62}$$

then the most likely codeword with respect to $\mathbf{r}$ must be in $J(i)$. From (10.60) we see that the threshold $G_i(\mathbf{v})$ monotonically increases with i. Therefore, as i increases, it becomes more and more likely that the condition of (10.62) will be satisfied, and the search for the ML codeword will be terminated.

Based on the foregoing development, we can formulate the KNIH algorithm as follows:

1. Decode $\mathbf{z}$ into a codeword with the algebraic decoder. Set $i = 0$. If decoding is successful, record the decoded codeword $\mathbf{v}$ and its correlation discrepancy $\lambda(\mathbf{r}, \mathbf{v})$ and go to step 2. Otherwise go to step 4.
2. Based on $\mathbf{v}$, determine the smallest integer $i_{\min}$ for which $\lambda(\mathbf{r}, \mathbf{v}) < G_{i_{\min}}(\mathbf{v})$. Set $i = i + 1$ and go to step 3.
3. If $i \leq i_{\min}$, find the codeword $\mathbf{v}'$ in $J(i) \backslash J(i-1)$ that has the least correlation discrepancy with $\mathbf{r}$. If $\lambda(\mathbf{r}, \mathbf{v}') < \lambda(\mathbf{r}, \mathbf{v})$, set $\mathbf{v} = \mathbf{v}'$. Go to step 2.
   If $i > i_{\min}$, then $\mathbf{v}$ is the ML codeword $\mathbf{v}_{ML}$. The decoding stops.
4. Find the codeword $\mathbf{v}$ in $J(i+1) \backslash J(i)$ that has the least correlation discrepancy with $\mathbf{r}$, set $i = i + 1$ and go to step 2. If $J(i+1) \backslash J(i)$ is empty (i.e., all the decodings of $\mathbf{z} + \mathbf{e}$ with $\mathbf{e} \in E(i+1) \backslash E(i)$ fail), set $i = i + 1$ and repeat step 4.

The KNIH algorithm always finds the most likely codeword $\mathbf{v}_{ML}$, but the number of candidate codewords to be generated depends on the SNR. Even though the average number of computations may be relatively small, the worst-case computational complexity is bounded by 2^k, the total number of codewords in C.

The definition of $G_i(\mathbf{v})$ given in (10.60) can be viewed as an improved version of $G(\mathbf{v}, w_1)$ given in (10.31), since in (10.60) the fact that at least $(t+1)$ positions

outside the i LRPs differ from the corresponding positions in $\mathbb{z}$ is used. In other words, $G(\mathbb{v}, w_1)$ given in (10.31) has been improved by exploiting the reprocessing strategy of the KNIH algorithm. The average number of computations obtained by the KNIH algorithm presented here can be further reduced if in condition (10.60) a further improvement of $G_i(\mathbb{v})$ is used. If the decoding of $\mathbb{z}$ is successful, then the algebraic decoding codeword $\mathbb{v}_0$ can be used in conjunction with the best recorded candidate codeword $\mathbb{v}$ to determine the optimality of $\mathbb{v}$ based on an improved version of $G(\mathbb{v}, w_1; \mathbb{v}_0, w_1)$ given in (10.46). If the decoding of $\mathbb{z}$ fails, then we simply use $G_i(\mathbb{v})$ in the condition of (10.60). Such a condition is used in [14].

This algorithm is effective for decoding codes of short to medium lengths. For long codes, computational complexity becomes very large, especially for low SNR. For example, consider the decoding of the (128, 64) extended BCH code. At SNR = 5 dB, the KNIH algorithm generates more than 2 million candidate codewords to decode one particular received sequence $\mathbb{r}$ [14]. To overcome the computational complexity problem we may set $i_{min} \leq i_{max} < n$ at step 2 of the algorithm. This reduces the worst-case computational complexity; however, the algorithm becomes suboptimum. A proper choice of i_{max} will provide a good trade-off between error performance and computational complexity. Also, solutions for reducing the average number of candidate codewords processed by the KNIH algorithm have been proposed in [15].

10.7 REDUCED LIST SYNDROME DECODING ALGORITHM

The *reduced list syndrome decoding* (RLSD) *algorithm* proposed by Snyders [16] provides a different approach to MLD. This algorithm is devised based on the parity-check matrix $\mathbb{H}$ of a code, the syndrome $\mathbb{s}$ of the hard-decision received vector $\mathbb{z}$, and the reliability measures of the symbols in the soft-decision received vector $\mathbb{r}$. The algorithm requires identification of at most the $n - k - 1$ least reliable independent positions (LRIPs) in $\mathbb{r}$ and is mostly suitable for decoding high rate codes with $n - k \ll k$. With respect to the LRP-reprocessing decoding algorithms presented in the previous four sections, this decoding algorithm requires additional effort to select independent least reliable positions in $\mathbb{r}$ for reprocessing. The LRIPs can be identified by eliminating dependent LRPs. A systematic procedure for eliminating dependent positions in $\mathbb{r}$ will be presented in Section 10.8.1.

The main idea of the RLSD algorithm is to construct a fixed reduced list of error patterns for the syndrome $\mathbb{s}$ computed from the hard-decision received vector $\mathbb{z}$ and the parity-check matrix $\mathbb{H}$. From this list of error patterns we generate a list of candidate codewords among which one is the ML codeword with respect to the soft-decision received vector $\mathbb{r}$. There are 2^{n-k} possible syndromes. Therefore, 2^{n-k} such lists of error patterns must be constructed. The construction can be tedious; however, code structure can be used to reduce the effort. For a given code, the construction of the reduced lists of error patterns for all possible syndromes need be done only once.

Reduced list syndrome decoding is carried out based on the formulation of MLD proposed in [17]. We express the parity-check matrix $\mathbb{H}$ in terms of its columns as follows:

$$\mathbb{H} = [\mathbb{h}_0, \mathbb{h}_1 \cdots \mathbb{h}_{n-1}], \tag{10.63}$$

where for $0 \leq i < n$, $\mathbb{h}_i$ denotes the ith column of $\mathbb{H}$. Let $\mathbb{v}$ be a codeword. We define an error pattern $\mathbb{e} = (e_0, e_1, \cdots, e_{n-1})$ as follows:

$$e_l = \begin{cases} 1 & \text{for } l \in D_1(\mathbb{v}), \\ 0 & \text{for } l \notin D_1(\mathbb{v}), \end{cases} \tag{10.64}$$

for $0 \leq l < n$, where $D_1(\mathbb{v})$ is the index set defined in (10.23). Then, $\mathbb{e} = \mathbb{z} + \mathbb{v}$, and $\mathbb{v} = \mathbb{e} + \mathbb{z}$. Consequently, the syndrome of $\mathbb{z}$ is

$$\begin{aligned} \mathbb{s} = \mathbb{z}\mathbb{H}^T &= \mathbb{e}\mathbb{H}^T \\ &= \sum_{l \in D_1(\mathbb{v})} \mathbb{h}_l. \end{aligned} \tag{10.65}$$

For a given codeword $\mathbb{v}$ the error pattern $\mathbb{e}$ defined in (10.64) is called an $n(\mathbb{v})$-pattern, where $n(\mathbb{v}) = |D_1(\mathbb{v})|$ as defined in (10.24). From (10.65) we see that an $n(\mathbb{v})$-pattern $\mathbb{e}$ is an error pattern of weight $n(\mathbb{v})$ such that the $n(\mathbb{v})$ columns of $\mathbb{H}$ corresponding to the nonzero positions of $\mathbb{e}$ (or the positions in $D_1(\mathbb{v})$) sum to the syndrome $\mathbb{s}$. We also can represent this $n(\mathbb{v})$-pattern with its corresponding set of columns of $\mathbb{H}$ summing to $\mathbb{s}$:

$$\{\mathbb{h}_l : l \in D_1(\mathbb{v})\}.$$

It follows from the definition of an $n(\mathbb{v})$-pattern, (10.13), and (10.65) that we can formulate MLD as follows: among all $n(\mathbb{v})$-patterns $\mathbb{e}$ for $n(\mathbb{v}) \leq n$, find the pattern $\mathbb{e}^*$ that minimizes the correlation discrepancy $\lambda(\mathbb{r}, \mathbb{z}+\mathbb{e})$ between $\mathbb{r}$ and the codeword $\mathbb{v} = \mathbb{e} + \mathbb{z}$. Then, the codeword $\mathbb{v}^* = \mathbb{e}^* + \mathbb{z}$ is the ML codeword with respect to $\mathbb{r}$.

For a given syndrome $\mathbb{s}$ there are 2^k $n(\mathbb{v})$-patterns; however, we can eliminate some of them from consideration. This follows from the fact that the rank of $\mathbb{H}$ is $n - k$. Using this fact, we can prove that for any $n(\mathbb{v})$-pattern $\mathbb{e}$ with $n(\mathbb{v}) > n - k$, there exists an $n(\mathbb{v}')$-pattern $\mathbb{e}'$ with $n(\mathbb{v}') \leq n - k$ such that

$$\mathbb{s} = \mathbb{e}\mathbb{H}^T = \mathbb{e}'\mathbb{H}^T. \tag{10.66}$$

The preceding equality implies that $\mathbb{e} + \mathbb{e}' = \mathbb{v}_0$ is a codeword. Furthermore, this codeword $\mathbb{v}_0$ has Hamming weight $n(\mathbb{v}) - n(\mathbb{v}')$. Consider the codewords $\mathbb{v} = \mathbb{e} + \mathbb{z}$ and $\mathbb{v}' = \mathbb{e}' + \mathbb{z}$. Then, $\mathbb{v}_0 = \mathbb{v} + \mathbb{v}'$. We can readily prove that

$$\begin{aligned} \lambda(\mathbb{r}, \mathbb{v}) &= \lambda(\mathbb{r}, \mathbb{v}') + \lambda(\mathbb{r}, \mathbb{v}_0). \\ &\geq \lambda(\mathbb{r}, \mathbb{v}'), \end{aligned} \tag{10.67}$$

which implies that the $n(\mathbb{v})$-pattern $\mathbb{e}$ cannot produce the ML codeword $\mathbb{v}_{ML}$. Consequently, all $n(\mathbb{v})$-patterns of weight $n(\mathbb{v}) > n - k$ for a given syndrome $\mathbb{s}$ can be eliminated from the search of the ML codeword $\mathbb{v}_{ML}$. In fact, more $n(\mathbb{v})$-patterns can be eliminated based on the partial knowledge of LRIPs. Based on the definition of an $n(\mathbb{v})$-pattern, the only $n(\mathbb{v})$-pattern $\mathbb{e}$ of weight 1 that needs to be considered is the one that corresponds to the column in $\mathbb{H}$ that is equal to the syndrome $\mathbb{s}$ (if such a column exists).

Let $\mathbb{q}_l$ denote the column of $\mathbb{H}$ that corresponds to the lth LRIP of $\mathbb{r}$, and let Ω_r denote the set

$$\Omega_r \triangleq \{\mathbb{q}_1, \mathbb{q}_2, \cdots \mathbb{q}_r\}. \tag{10.68}$$

An $n(\mathbf{v}')$-pattern $\mathbf{e}'$ can be eliminated from the decoding process if Ω_r is sufficiently known to determine an $n(\mathbf{v})$-pattern $\mathbf{e}$ such that $\lambda(\mathbf{r}, \mathbf{v}) \leq \lambda(\mathbf{r}, \mathbf{v}')$, where $\mathbf{v} = \mathbf{e} + \mathbf{z}$ and $\mathbf{v}' = \mathbf{e}' + \mathbf{z}$, respectively. Such an $n(\mathbf{v}')$-pattern $\mathbf{e}'$ is said to be Ω_r-*eliminated*. For a given code, a reduced list of $n(\mathbf{v})$-patterns can be constructed by discarding all Ω_r-eliminated patterns for $r \leq n - k$. For example, consider the $n(\mathbf{v})$-pattern $\{\mathbb{q}_1, \mathbb{h}_{i_2}, \mathbb{h}_{i_3}, \cdots \mathbb{h}_{i_{n(\mathbf{v})}}\}$, with $i_2 < i_3 < \cdots < i_{n(\mathbf{v})}$, where for $2 \leq j \leq n(\mathbf{v})$, $\mathbb{h}_{i_j} \neq \mathbb{q}_1$. Then, any $n(\mathbf{v}')$-pattern $\mathbf{e}'$ with $n(\mathbf{v}') \geq n(\mathbf{v})$ that contains $\{\mathbb{h}_{i_2}, \mathbb{h}_{i_3}, \cdots, \mathbb{h}_{i_{n(\mathbf{v})}}\}$ as a subset is Ω_1-eliminated.

Determining all Ω_r-eliminated error patterns for $1 \leq r \leq n - k$ is a code-dedicated, often tedious procedure; however, it is possible to derive some general guidelines for removing error patterns from the decoding process for particular values of r. For example, the next theorem provides one such guideline, which follows directly from the fact that the rank of $\mathbb{H}$ is $n - k$.

> THEOREM 10.2 If $\mathbf{s} \neq \sum_{i=1}^{n-k} \mathbb{q}_i$, then all $n(\mathbf{v})$-patterns with $n(\mathbf{v}) = n - k$ are Ω_{n-k}-eliminated. Otherwise, $\{\mathbb{q}_1, \mathbb{q}_2, \cdots \mathbb{q}_{n-k}\}$ is the only $n(\mathbf{v})$-pattern of weight $n - k$ that is not Ω_{n-k}-eliminated.

In [16], several theorems that allow elimination of error patterns from the reduced lists are derived. The sizes of the lists can be further reduced by exploiting the structures and parameters of the code considered, as shown in the following example code.

EXAMPLE 10.2

Consider the (7, 4, 3) Hamming code with parity-check matrix

$$\mathbb{H} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}. \tag{10.69}$$

For a Hamming code of length $2^m - 1$, since $\mathbb{H}$ consists of all the nonzero m-tuples as columns, a nonzero syndrome $\mathbf{s}$ must be identical to one of the columns in $\mathbb{H}$. Therefore, there is an $n(\mathbf{v})$-pattern $\{\mathbf{s}\}$ of weight 1. The reduced list for the (7, 4, 3) Hamming code consists of one $n(\mathbf{v})$-pattern $\{\mathbf{s}\}$ of weight 1, three $n(\mathbf{v})$-patterns $\{\mathbb{q}_1, \mathbb{q}_1 + \mathbf{s}\}$, $\{\mathbb{q}_2, \mathbb{q}_2 + \mathbf{s}\}$, and $\{\mathbb{q}_1 + \mathbb{q}_2, \mathbb{q}_1 + \mathbb{q}_2 + \mathbf{s}\}$, of weight 2, and one $n(\mathbf{v})$-pattern $\{\mathbb{q}_1, \mathbb{q}_2, \mathbb{q}_1 + \mathbb{q}_2 + \mathbf{s}\}$ of weight 3. All other $n(\mathbf{v})$-patterns of weight 2 are Ω_1-eliminated or Ω_2-eliminated. For example, $\{\mathbb{q}_3, \mathbb{q}_2 + \mathbb{q}_3\}$ is Ω_2-eliminated as $\mathbf{s} = \mathbb{q}_2$. In general, MLD is achieved by first identifying those $n(\mathbf{v})$-patterns on the list that are valid for the computed syndrome $\mathbf{s}$ and then finding the ML codeword $\mathbf{v}_{ML}$ among the candidate codewords generated by the $n(\mathbf{v})$-patterns for the computed $\mathbf{s}$. For Hamming codes, all patterns on the list have to be kept, since $\mathbb{H}$ consists of all the nonzero m-tuples as columns. Also, depending on the syndrome $\mathbf{s}$ computed, the complete list can be further reduced. For example, if $\mathbf{s} = \mathbb{q}_1$, the list reduces to $\{\mathbf{s}\}$, and for $\mathbf{s} = \mathbb{q}_1 + \mathbb{q}_2$, the list reduces to $\{\mathbf{s}\}$ and $\{\mathbb{q}_1, \mathbb{q}_2\}$; however, if $\mathbf{s} = \mathbb{q}_1 + \mathbb{q}_2 + \mathbb{q}_3$, then the entire list needs to be considered.

Let $\mathbf{r} = (-0.7; -0.8; -1.2; 0.5; 0.4; -1.1; -0.6)$ be the received sequence. The hard-decision received vector is $\mathbf{z} = (0\ 0\ 0\ 1\ 1\ 0\ 0)$, and the syndrome of $\mathbf{z}$ is $\mathbf{s} = \mathbf{z}\mathbb{H}^T = (101) = \mathbb{h}_5^T$. We find that $\mathbb{q}_1 = \mathbb{h}_4$, $\mathbb{q}_2 = \mathbb{h}_3$, and $\mathbf{s} = \mathbb{q}_1 + \mathbb{q}_2$. As a result,

only the $n(\mathbf{v})$-patterns $\{\mathbb{h}_5\}$ and $\{\mathbb{h}_3, \mathbb{h}_4\}$ have to be considered. The error pattern corresponding to $\{\mathbb{h}_5\}$ is $\mathbb{e}_1 = (0\ 0\ 0\ 0\ 0\ 1\ 0)$, and the codeword generated by this error pattern is $\mathbb{v}_1 = \mathbb{e}_1 + \mathbb{z} = (0\ 0\ 0\ 0\ 0\ 1\ 0) + (0\ 0\ 0\ 1\ 1\ 0\ 0) = (0\ 0\ 0\ 1\ 1\ 1\ 0)$. The correlation discrepancy of $\mathbb{v}_1$ is $\lambda(\mathbb{r}, \mathbb{v}_1) = 1.1$. The error pattern corresponding to $\{\mathbb{h}_3, \mathbb{h}_4\}$ is $\mathbb{e}_2 = (0\ 0\ 0\ 1\ 1\ 0\ 0)$, and the codeword generated by this error pattern is $\mathbb{v}_2 = \mathbb{e}_2 + \mathbb{z} = (0\ 0\ 0\ 1\ 1\ 0\ 0) + (0\ 0\ 0\ 1\ 1\ 0\ 0) = (0\ 0\ 0\ 0\ 0\ 0\ 0)$. The correlation discrepancy of $\mathbb{v}_2$ is $\lambda(\mathbb{r}, \mathbb{v}_2) = 0.9$. Hence, $\mathbb{v}_2 = (0\ 0\ 0\ 0\ 0\ 0\ 0)$ is the ML codeword.

In [16], the reduced lists of $n(\mathbf{v})$-patterns for the (15, 11, 3) and (31, 26, 3) Hamming codes are given. The complete reduced list for the (15, 11, 3) code is composed of one $n(\mathbf{v})$-pattern of weight 1, seven $n(\mathbf{v})$-patterns of weight 2, seven $n(\mathbf{v})$-patterns of weight 3, and one $n(\mathbf{v})$-pattern of weight 4. Also in [16], a method is presented for efficiently constructing the reduced list of $n(\mathbf{v})$-patterns for an extended code based on the reduced list of the original code. Schemes for constructing reduced lists of $n(\mathbf{v})$-patterns for codes based on binary trees and graphs are also presented in [18]. MLD of high-rate codes based on the RLSD algorithm can be achieved with very few real operations and therefore is quite suitable for high-speed decoding of these codes.

10.8 MOST RELIABLE INDEPENDENT POSITION REPROCESSING DECODING ALGORITHMS

In the previous four sections we presented various soft-decision decoding algorithms for linear block codes that are based on processing the LRPs of a received sequence. These decoding algorithms are referred to as LRP-reprocessing decoding algorithms. In this section we present two soft-decision decoding algorithms that are based on the processing of MRIPs of a received sequence. These decoding algorithms are referred to as *MRIP-reprocessing decoding algorithms*.

10.8.1 Most Reliable and Least Reliable Bases

For the GMD and Chase-type decoding algorithms, only a partial ordering of the reliability values of the received symbols is needed to identify the LRPs of a received sequence for decoding; however, decoding based on MRIPs, requires not only a complete ordering of the received sequence based on their reliability values but also identification of k MRIPs. These k MRIPs can be determined by permuting the columns of the generator matrix of a code based on the ordering of the received symbols and then finding k independent columns by elementary row operations of the permuted generator matrix.

Consider a binary (n, k) linear block code C with generator matrix $\mathbb{G}$. Suppose a code sequence is transmitted. Let $\mathbb{r} = (r_0, r_1, \cdots, r_{n-1})$ be the received sequence. We order the received symbols based on their reliability values in decreasing order (if there is a tie between two received symbols, then we order them arbitrarily). The resultant sequence is denoted by

$$\mathbb{r}' = (r_0', r_1', \cdots, r_{n-1}'), \tag{10.70}$$

with $|r_0'| > |r_1'| > \cdots > |r_{n-1}'|$. This reordering of the received symbols defines a permutation π_1 for which $\mathbb{r}' = \pi_1[\mathbb{r}]$. We permute the columns of $\mathbb{G}$ based on π_1 and

obtain the following matrix:

$$\mathbb{G}' = \pi_1[\mathbb{G}] = [\mathbb{g}_0', \mathbb{g}_1' \cdots \mathbb{g}_{n-1}'], \tag{10.71}$$

where for $0 \leq i < n$, $\mathbb{g}_i'$ denotes the ith column of $\mathbb{G}'$. The code C' generated by $\mathbb{G}'$ is equivalent to C generated by $\mathbb{G}$; that is,

$$C' = \pi_1[C] = \{\pi_1(\mathbb{v}) : \mathbb{v} \in C\}.$$

Because the ith column $\mathbb{g}_i'$ of $\mathbb{G}'$ corresponds to the ith component r_i' of $\mathbb{r}'$ with reliability value $| r_i' |$, we call $| r_i' |$ the reliability value associated with column $\mathbb{g}_i'$.

Although the first k positions of $\mathbb{r}'$ (or the first k columns of $\mathbb{G}'$) are the k most reliable positions, they are not necessarily independent, and therefore they do not always represent an information set. To determine the k MRIPs, we perform elementary row operations (or Gaussian eliminations) to put $\mathbb{G}'$ in the reduced echelon form. There are k columns in $\mathbb{G}'$ in reduced echelon form that contain only one 1. These k columns are linearly independent. Consequently, the positions in $\mathbb{r}'$ that correspond to these k linearly independent columns are the k MRIPs. Note that the matrix in reduced echelon form generates the same code C' as $\mathbb{G}'$, except for the mapping between an information sequence and a codeword. We use these k linearly independent columns as the first k columns of a new generator matrix $\mathbb{G}''$, maintaining the decreasing order of their associated reliability values. The remaining $n - k$ columns of $\mathbb{G}'$ in reduced echelon form give the next $n - k$ columns of $\mathbb{G}''$ arranged in order of decreasing associated reliability values. This process defines a second permutation π_2. It is clear that the code generated by $\mathbb{G}''$ is

$$C'' = \pi_2[C'] = \pi_2[\pi_1[C]].$$

Rearranging the components of $\mathbb{r}'$ according to the permutation π_2, we obtain the sequence

$$\mathbb{y} = (y_0, y_1, \cdots, y_{k-1}, y_k, \cdots, y_{n-1}), \tag{10.72}$$

with $| y_0 | > | y_1 | > \cdots > | y_{k-1} |$, and $| y_k | > \cdots > | y_{n-1} |$. It is clear that $\mathbb{y} = \pi_2[\mathbb{r}'] = \pi_2[\pi_1[\mathbb{r}]]$. The first k components of $\mathbb{y}$ are the k MRIPs. These k MRIPs are said to form the *most reliable basis* (MRB) for code C.

We can permute the rows of $\mathbb{G}''$ to obtain a generator matrix $\mathbb{G}_1$ in systematic form,

$$\mathbb{G}_1 = [\mathbb{I}_k \mathbb{P}] = \begin{bmatrix} 1 & 0 & \cdots & 0 & p_{0,0} & \cdots & p_{0,n-k-1} \\ 0 & 1 & \cdots & 0 & p_{1,0} & \cdots & p_{1,n-k-1} \\ \vdots & & \ddots & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 & p_{k-1,0} & \cdots & p_{k-1,n-k-1} \end{bmatrix}, \tag{10.73}$$

where $\mathbb{I}_k$ represents the $k \times k$ identity matrix, and $\mathbb{P}$ is the $k \times (n - k)$ parity-check matrix. The code C_1 defined by $\mathbb{G}_1$ consists of the same 2^k codewords as code C'' and is equivalent to both C' and C. Suppose that $\mathbb{G}_1$ is used for decoding the permuted received sequence $\mathbb{y}$. Assume that $\hat{\mathbb{v}}$ is the decoded codeword in C_1. Then, $\mathbb{v} = \pi_1^{-1}\pi_2^{-1}[\hat{\mathbb{v}}]$ is the decoded codeword in C, where π_1^{-1} and π_2^{-1} are the inverse permutations of π_1 and π_2, respectively (see Problem 10.11).

By analogy with the MRB, we define the least reliable basis (LRB) as the set of the $n - k$ least reliable independent positions (LRIP) in the dual code $C^{\perp}$ of C. Then, it is shown in [19] that the positions not included in the MRB form the LRB. Because the complement of an information set is an information set of the dual code, it follows from (10.73) that $\mathbb{H}_1 = [\mathbb{P}^T \mathbb{I}_{n-k}]$ is the parity-check matrix of the code C_1. Consequently, if $n - k < k$, we can determine $\mathbb{G}_1$ after constructing $\mathbb{H}_1$ in the dual code $C^{\perp}$. In this case, the ordering is from right to left in increasing order of reliability values. This process also defines two permutations μ_1 and μ_2 corresponding to the ordering of the received sequence in increasing order of reliability values and the determination of the $n - k$ first independent positions of this ordering, respectively. It follows that at most $n \cdot \min\{k, n-k\}^2$ binary additions are performed to construct $\mathbb{G}_1$. Also, although the received sequence can be ordered with $O(n \log_2(n))$ comparisons if unquantized values are considered, the ordering corresponding to π_1 (or μ_1) becomes trivial if quantized values are used, since only the numbers of quantized values corresponding to each quantization level suffice to determine the complete ordering of the quantized received sequence.

10.8.2 Priority-First Search Decoding Algorithm

As shown in Chapter 9, every linear block code can be represented graphically by a trellis. Another useful graphical representation of a linear block code is a tree. Every binary linear systematic (n, k) block code C can be represented by a binary tree T, as shown in Figure 10.7. This binary tree has the following structures:

1. T consists of $n + 1$ levels.
2. For $0 \leq i < k$, there are 2^i nodes at the ith level of the tree. For $k \leq i \leq n$, the number of nodes at the ith level of the tree is 2^k. There is only one node s_0 at

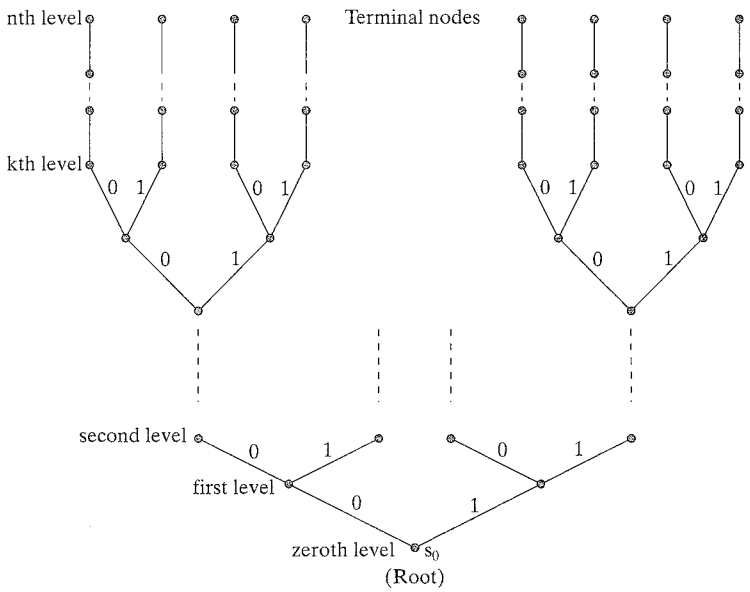

FIGURE 10.7: A tree representation of a binary linear systematic block code.

the zeroth level of the tree called the *initial node* (or the *root*) of the tree, and there are 2^k nodes at the nth level of the tree, which are called the *terminal nodes* of the tree.

3. For $0 \leq i < k$, there are two branches leaving every node s_i at level-i and connecting to two different nodes at level-$(i+1)$. One branch is labeled with an information symbol 0, and the other branch is labeled with an information symbol 1. For $k \leq i \leq n$, there is only one branch leaving every node s_i at level-i and connecting to one node at level-$(i+1)$. This branch is labeled with a parity-check symbol, either 0 or 1.
4. The label sequence of a path connecting the initial node s_0 to a node s_k at the kth level corresponds to an information sequence $\mathbf{u}$ of k bits. The label sequence of the path connecting the initial node s_0 through a node s_k at the kth level to a terminal node s_n at the nth level is a codeword in C. The label sequence of the tail connecting node s_k to node s_n corresponds to the $n-k$ parity-check symbols of the codeword.

From the preceding definition of a code tree, it is clear there is a one-to-one correspondence between a codeword in C and a path connecting the initial node s_0 to a terminal node s_n. The set of nodes at the ith level of the tree is called the *node* (or *state*) *space*, denoted by $\Sigma_i(C)$.

EXAMPLE 10.3

Figure 10.8 depicts the tree representation of the systematic (6, 3) linear code generated by

$$\mathbb{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}.$$

The tree representation of a linear block code can be used to facilitate decoding. One such tree-based decoding algorithm is the *priority-first search* (PFS) decoding algorithm devised by Han, Hartmann, and Chen [20]. This decoding algorithm processes the tree T_1 of the permuted code C_1 generated by the generator matrix $\mathbb{G}_1$ given in (10.73), where the first k levels of the tree correspond to the k MRIPs of $\mathbf{r}$. The basic concept of this algorithm is to penetrate the tree from the initial node s_0, level by level, to search for the ML path. At each level, a list of likely paths is constructed, and only the most likely path on the list is extended. Because the starting part of the tree corresponds to the most reliable positions and contains only a smaller number of nodes at each level, it is possible to direct the search of the ML path in a particular highly reliable direction and hence keep the size of the list and the computation effort small. Each time the kth level of the tree is reached, a candidate codeword is generated and tested. The candidate codeword with the least discrepancy is kept. For each test, all the partial paths on the list with correlation discrepancies larger than this least correlation discrepancy are discarded from the list. This process is repeated until the list becomes empty. Then, the stored candidate codeword $\mathbf{v}^*$ with the least correlation discrepancy is the decoded codeword and is

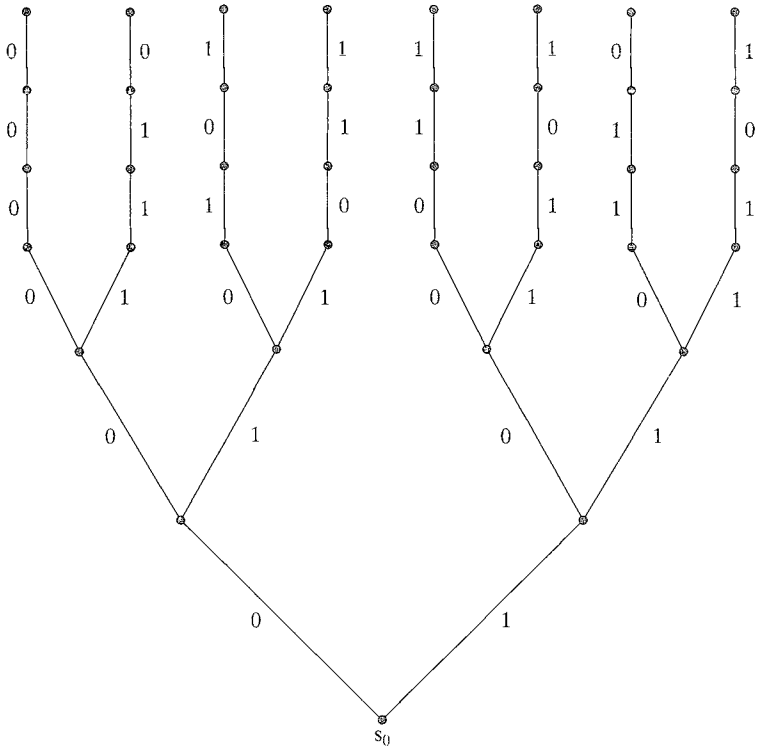

FIGURE 10.8: A tree representation for a systematic (6, 3) linear code.

the ML codeword in C_1 with respect to the permuted received sequence $\mathbf{y}$. Then,

$$\mathbf{v} = \pi_1^{-1}[\pi_2^{-1}[\mathbf{v}^*]]$$

is the ML codeword in C with respect to $\mathbf{r}$.

The PFS decoding is carried out based on a *cost function* $f(s_i)$ associated with each state $s_i \in \Sigma_i(C)$. This cost function is a measure of the correlation discrepancy of any path that starts from the initial node s_0, passes through the node s_i, and ends at a terminal node of the tree. It is used to guide the search of the most likely path in the code tree along the most likely direction. Decoding is performed on the ordered received sequence

$$\mathbf{y} = \pi_2[\pi_1[\mathbf{r}]] = (y_0, y_1, \cdots, y_{n-1}).$$

Let $\mathbf{z} = (z_0, z_1, \cdots, z_{n-1})$ denote the hard-decision vector obtained from $\mathbf{y}$. For $1 \leq i \leq n$, let $\mathbf{p}(s_i) \triangleq (v_0, v_1, \cdots, v_{i-1})$ denote the path in the code tree that connects the initial node s_0 to node s_i at the ith level. We define the binary sequence $\mathbf{v}(s_i) = (v_0(s_i), v_1(s_i), \cdots, v_{n-1}(s_i))$, where

$$v_l(s_i) = \begin{cases} v_l & \text{for } 0 \leq l < i, \\ z_l & \text{for } i \leq l < n. \end{cases} \tag{10.74}$$

Therefore, $\mathbf{p}(s_i)$ is a prefix of $\mathbf{v}(s_i)$. Note that $\mathbf{v}(s_i)$ and $\mathbf{z}$ differ only within the positions in $\{0, 1, \cdots, i-1\}$. Let $D_1(\mathbf{v}(s_i))$ denote the set of positions where $\mathbf{v}(s_i)$

and z differ. Then, $D_1(\mathbb{v}(s_i)) \subseteq \{0, 1, \cdots, i-1\}$. It follows from the definition of correlation discrepancy given by (10.13) that

$$\lambda(\mathbb{y}, \mathbb{v}(s_i)) = \sum_{l \in D_1(\mathbb{v}(s_i))} |y_l|. \tag{10.75}$$

For $1 \leq i \leq k$, let $\mathbb{v}^i$ represent any of the 2^{k-i} paths of length n ending at a terminal node of the code tree T_1 that have $(v_0, v_1, \cdots, v_{i-1})$ as the common prefix and diverge from node s_i at level-i. These paths correspond to 2^{k-i} codewords in C_1. The correlation discrepancy between $\mathbb{v}^i$ and y is

$$\lambda(\mathbb{y}, \mathbb{v}^i) = \sum_{l \in D_1(\mathbb{v}^i)} |y_l|. \tag{10.76}$$

where $D_1(\mathbb{v}^i)$ denotes the set of positions where $\mathbb{v}^i$ and z differ. It follows from the definitions of $\mathbb{v}(s_i)$ and $\mathbb{v}^i$ that

$$D_1(\mathbb{v}(s_i)) \subseteq D_1(\mathbb{v}^i). \tag{10.77}$$

From (10.75), (10.76), and (10.77), we obtain the following lower bound on $\lambda(\mathbb{y}, \mathbb{v}^i)$: for $1 \leq i \leq k$,

$$\lambda(\mathbb{y}, \mathbb{v}(s_i)) \leq \lambda(\mathbb{y}, \mathbb{v}^i). \tag{10.78}$$

The preceding lower bound on $\lambda(\mathbb{y}, \mathbb{v}^i)$ can be further improved based on ideas similar to those presented in Section 10.3. Let $\tilde{\mathbb{u}}$ be the information sequence defined by $\tilde{u}_l = z_l$ for $0 \leq l \leq k-1$; that is, $\tilde{\mathbb{u}}$ is simply formed by the first k-bits of the hard-decision received vector z. From $\tilde{\mathbb{u}}$, we form the codeword $\tilde{\mathbb{v}} = \tilde{\mathbb{u}}\mathbb{G}_1$, which is called the *initial seed* codeword. Because $\mathbb{G}_1$ is in systematic form, the information part of $\tilde{\mathbb{v}}$ is $(z_0, z_1, \cdots, z_{k-1})$. Therefore, $\tilde{\mathbb{v}}$ and z differ only within the positions in $\{k, k+1, \cdots, n-1\}$. Based on the weight profile $\{w_0 = 0, w_1, w_2, \cdots, w_m\}$ of C, we define

$$\delta(s_i) = \min_{l \in \{0, m\}} \{|w_l - n(\tilde{\mathbb{v}}) - n(\mathbb{v}(s_i))|\}, \tag{10.79}$$

where $n(\tilde{\mathbb{v}}) = |D_1(\tilde{\mathbb{v}})|$, and $n(\mathbb{v}(s_i)) = |D_1(\mathbb{v}(s_i))|$. Since $\tilde{\mathbb{v}}$ differs from z in $n(\tilde{\mathbb{v}})$ LRPs, and $\mathbb{v}(s_i)$ differs from z in $n(\mathbb{v}(s_i))$ MRPs, $\delta(s_i)$ represents the minimum number of additional positions in which z and $\mathbb{v}(s_i)$ must differ for $\mathbb{v}(s_i)$ to be a valid codeword. Consequently, we obtain the following lower bound on $\lambda(\mathbb{y}, \mathbb{v}^i)$:

$$\lambda(\mathbb{y}, \mathbb{v}^i) \geq \lambda(\mathbb{y}, \mathbb{v}(s_i)) + \sum_{l=1}^{\delta(s_i)} |y_{n-l}|. \tag{10.80}$$

We define

$$f(s_i) \triangleq \lambda(\mathbb{y}, \mathbb{v}(s_i)) + \sum_{l=1}^{\delta(s_i)} |y_{n-l}|, \tag{10.81}$$

which is called the cost function of node s_i. This cost function consists of two parts. The first part of $f(s_i)$, $\lambda(\mathbb{y}, \mathbb{v}(s_i))$, represents the contribution from the

common prefix $(v_0, v_1, \cdots, v_{i-1})$ of all the 2^{k-i} codewords (or paths) diverging from node s_i. The second part of $f(s_i)$, $\sum_{l=1}^{\delta(s_i)} |y_{n-l}|$, represents a lower bound on the contributions from a tail $(v_i, v_{i+1}, \cdots, v_{n-1})$ of a codeword diverging from node s_i. The cost function $f(s_i)$ simply says that each codeword passing through node s_i has a correlation discrepancy of at least $f(s_i)$. For two different nodes, s_i and s_j, in the tree, if $f(s_i) < f(s_j)$, then it is reasonable to assume that a path passing through node s_i is more likely to produce the received sequence $\mathbf{y}$ than a path passing through node s_j. Therefore, a higher priority should be given to node s_i to penetrate the tree in search of the ML codeword. This assumption forms the basis of the PFS decoding algorithm.

An important property of the cost function $f(s_i)$ is that along a path from node s_0 to node s_i, it is a *nondecreasing function* of the tree level i. For $0 \leq i < k$, consider the two nodes $s_{i+1}^{(1)}$ and $s_{i+1}^{(2)}$ in $\Sigma_{i+1}(C)$ that are adjacent to node $s_i \in \Sigma_i(C)$ with a cost function $f(s_i)$. Without loss of generality, we assume that the label bit for the branch $(s_i, s_{i+1}^{(1)})$ is identical to z_i, and the label bit for the branch $(s_i, s_{i+1}^{(2)})$ is $z_i + 1$, the complement of z_i. Then, it follows from the definition of $\mathbf{v}(s_j)$ that $\mathbf{v}(s_i) = \mathbf{v}(s_{i+1}^{(1)})$, and $\mathbf{v}(s_{i+1}^{(2)})$ differs from $\mathbf{v}(s_{i+1}^{(1)})$ only at the ith position. We readily see that

$$f(s_{i+1}^{(1)}) = f(s_i). \tag{10.82}$$

From (10.81) we can compute $f(s_{i+1}^{(2)})$. It follows from (10.79) and the fact that $\mathbf{v}(s_{i+1}^{(2)})$ and $\mathbf{v}(s_{i+1}^{(1)})$ differ only at the ith position that

$$f(s_{i+1}^{(2)}) - f(s_i) \geq |y_i| - |y_{n-\delta(s_i)}| \geq 0. \tag{10.83}$$

The equality of (10.82) and the inequality of (10.83) imply that for a path from node s_0 to node s_i, $1 \leq i \leq k$, the cost function $f(s_i)$ is a nondecreasing function of the tree level i. For $k \leq i < n$, there is only one branch leaving a node $s_i \in \Sigma_i(C)$ and connecting a node $s_{i+1} \in \Sigma_{i+1}(C)$. Therefore, $\mathbf{v}^i = \mathbf{v}^k$ for $i \geq k$. In this case, the cost function $f(s_i)$ is simply $f(s_k) = \lambda(\mathbf{y}, \mathbf{v}^k)$; that is,

$$f(s_i) = f(s_k) = \lambda(\mathbf{y}, \mathbf{v}^k). \tag{10.84}$$

The PFS decoding algorithm is devised based on the cost function $f(s_i)$ to penetrate a code tree using a depth-first processing procedure. This algorithm continuously updates a list of nodes in the tree that have been visited (or reached), and their cost functions. This list is called the *cost function list* (CFL). The nodes are ordered in increasing values of their cost functions, such that the first (or top) node of the list has the smallest cost function, and the last (or bottom) node of the list has the largest cost function. The first node on this ordered list has the highest priority to be chosen as the location (or the point) for penetrating the tree in the search of the most likely path. The algorithm executes the following steps:

1. Construct the initial seed codeword $\tilde{\mathbf{v}}$. Compute the cost function $f(s_0)$ of the initial node s_0 and form the initial cost function list $CFL_0 = \{s_0, f(s_0)\}$. Go to step 2.
2. Check the first node s_i on the list and remove its cost function $f(s_i)$ from the list. If $i < k$, go to step 3; otherwise, go to step 4.

3. Extend the node s_i into its two adjacent nodes, $s_{i+1}^{(1)}$ and $s_{i+1}^{(2)}$, at level-$(i+1)$ of the tree. Compute the cost functions $f(s_{i+1}^{(1)})$ and $f(s_{i+1}^{(2)})$ of $s_{i+1}^{(1)}$ and $s_{i+1}^{(2)}$, and reorder the list. Go to step 2.
4. If $i = k$, evaluate the cost function $f(s_k) = \lambda(\mathrm{y}, \mathrm{v}^k)$ and record the codeword v_{best} that has the smallest cost function among all codewords tested so far. Delete all the nodes on the list whose cost functions are larger than $\lambda(\mathrm{y}, \mathrm{v}_{best})$. If the list is not empty, go to step 2; otherwise, go to step 5.
5. Output v_{best} as the decoded codeword, and stop the decoding process.

Note that in step 4, nodes on the list with cost functions larger than $\lambda(\mathrm{y}, \mathrm{v}_{best})$ can be deleted owing to the monotonicity of the cost function. The codewords passing through these nodes do not contain the ML codeword and hence should be removed from further consideration. Also, (10.82) implies that for $i < k$, $s_{i+1}^{(1)}$ is always put at the top of the list. Because the algorithm is to minimize the cost function (correlation discrepancy) of a codeword at level-k, the output v_{best} is the MLD codeword.

EXAMPLE 10.4

Consider the (8, 4, 4) RM code with weight profile $W = \{0, 4, 8\}$ and generator matrix

$$\mathbb{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}. \tag{10.85}$$

Suppose the all-zero codeword $\mathrm{v} = (0\,0\,0\,0\,0\,0\,0\,0)$ is transmitted, and

$$\begin{aligned} \mathrm{r} &= (r_0, r_1, r_2, r_3, r_4, r_5, r_6, r_7) \\ &= (-1.2,\ -1.0,\ -0.9,\ -0.4,\ 0.7,\ -0.2,\ -0.3,\ -0.8) \end{aligned}$$

is received. In decoding r, the PFS algorithm first orders the symbols of r in decreasing reliability values, so we obtain

$$\begin{aligned} \mathrm{r}' &= (r_0', r_1', r_2', r_3', r_4', r_5', r_6', r_7') \\ &= \pi_1(\mathrm{r}) = (r_0, r_1, r_2, r_7, r_4, r_3, r_6, r_5) \\ &= (-1.2,\ -1.0,\ -0.9,\ -0.8,\ 0.7,\ -0.4,\ -0.3,\ -0.2). \end{aligned} \tag{10.86}$$

Permuting the columns of $\mathbb{G}$ based on the permutation π_1, we obtain the matrix

$$\mathbb{G}' = \pi_1[\mathbb{G}] = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}. \tag{10.87}$$

First, we notice that the first three columns of $\mathbb{G}'$, $\mathbb{g}_0'$, $\mathbb{g}_1'$, and $\mathbb{g}_2'$, are linearly independent; however, the fourth column, $\mathbb{g}_3'$, depends on the first three columns. The fifth column, $\mathbb{g}_4'$, is linearly independent of the first three columns. Therefore, $\mathbb{g}_0'$, $\mathbb{g}_1'$, $\mathbb{g}_2'$, and $\mathbb{g}_4'$ form the most reliable basis. In r', the MRIPs are 0, 1, 2, and 4.

With inverse permutation π_1^{-1}, we find that in $\mathbf{r}$ the MRIPs are also 0, 1, 2, and 4. Adding the fourth row of $\mathbf{G}'$ to the second and third rows, and permuting the fourth and fifth columns of the resultant matrix, we obtain the following matrix:

$$\mathbf{G}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}. \tag{10.88}$$

The permutation π_2 results in the following received vector,

$$\begin{aligned} \mathbf{y} &= (y_0, y_1, y_2, y_3, y_4, y_5, y_6, y_7) \\ &= \pi_2(\mathbf{r}') = (r_0', r_1', r_2', r_4', r_3', r_5', r_6', r_7') \\ &= (-1.2, -1.0, -0.9, 0.7, -0.8, -0.4, -0.3, -0.2). \end{aligned} \tag{10.89}$$

The hard-decision received vector $\mathbf{z}$ obtained from $\mathbf{y}$ is

$$\mathbf{z} = (0, 0, 0, 1, 0, 0, 0, 0).$$

We set $\tilde{\mathbf{u}} = (0001)$. The initial seed codeword $\tilde{\mathbf{v}}$ is given by

$$\tilde{\mathbf{v}} = \tilde{\mathbf{u}}\mathbf{G}_1 = (0, 0, 0, 1, 0, 1, 1, 1).$$

Comparing $\tilde{\mathbf{v}}$ with $\mathbf{z}$, we find that $D_1(\tilde{\mathbf{v}}) = \{5, 6, 7\}$, and $n(\tilde{\mathbf{v}}) = |D_1(\tilde{\mathbf{v}})| = 3$. Because $\mathbf{v}(s_0) = \mathbf{z}$, $D_1(\mathbf{v}(s_0)) = \emptyset$, and $n(\mathbf{v}(s_0)) = |D_1(\mathbf{v}(s_0))| = 0$. From (10.79) and (10.81) we find that $\delta(s_0) = 1$, and $f(s_0) = 0.2$, respectively. Then, the PFS algorithm for decoding $\mathbf{y}$ is initialized with the cost function list $CFL_0 = \{s_0, f(s_0) = 0.2\}$ (or simply, $\{f(s_0) = 0.2\}$).

The decoding process is depicted by the tree shown in Figure 10.9. At the first step, s_0 is extended to nodes $s_1^{(1)}$ and $s_1^{(2)}$ with connecting branches $(s_0, s_1^{(1)}) = 0$ and $(s_0, s_1^{(2)}) = 1$. Because $D_1(\mathbf{v}(s_1^{(2)}) = \{0\}$, and $n(\mathbf{v}(s_1^{(2)})) = 1$, we find from (10.79) that $\delta(s_1^{(2)}) = 0$. From (10.81) we find that $f(s_1^{(2)}) = 1.2$. Updating CFL_0, we have a new $CFL = \{s_1^{(1)}, f(s_1^{(1)}) = 0.2; s_1^{(2)}, f(s_1^{(2)}) = 1.2\}$ (or simply, $\{f(s_1^{(1)}) = 0.2; f(s_1^{(2)}) = 1.2\}$).

The rest of the decoding steps are summarized in Table 10.2. From this table we find that MLD is achieved in six steps. The ML codeword is $\mathbf{v}_{best} = (0\,0\,0\,0\,0\,0\,0\,0)$ with correlation discrepancy $\lambda(\mathbf{y}, \mathbf{v}_{best}) = 0.7$.

The PFS algorithm performs MLD efficiently for binary linear block codes of lengths up to 128 at medium to high SNR; however, at low SNR, the number of computations required by the algorithm to achieve MLD may become prohibitively large, especially for long codes, owing to the very large number of nodes in the tree to be visited and extended, which results in a very large cost function list. The worst-case scenario is that all the 2^k nodes at the kth level of the tree must be visited and their cost functions computed. To overcome this problem, a suboptimum version of the PFS algorithm is presented in [21], in which a limit is put on the

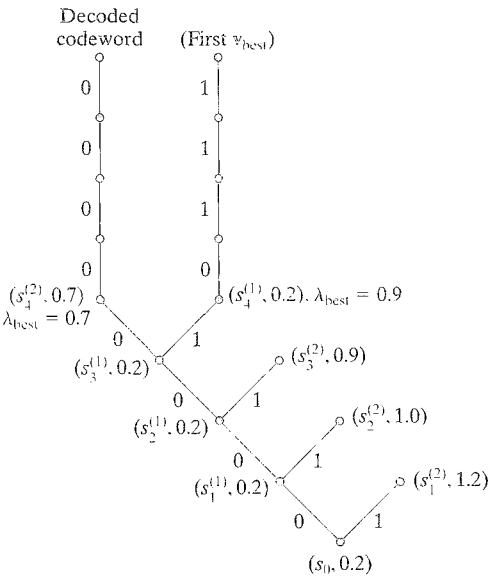

FIGURE 10.9: Decoding tree for Example 10.4.

TABLE 10.2: Decoding steps for PFS algorithm.

Step	s_i	$s_{i+1}^{(l)}$ s_{i+1}^{2}	$\mathrm{p}(s_{i+1}^{(l)})$ $\mathrm{p}(s_{i+1}^{2})$	$\lvert D_l(\mathrm{v}(s_{i+1}^{(l)}))\rvert$ $\lvert D_l(\mathrm{v}(s_{i+1}^{(2)}))\rvert$	$\delta(s_{i+1}^{(l)})$ $\delta(s_{i+1}^{2})$	v_{best} $\lambda(y, v_{best})$	CFL
1	s_0	$s_1^{(1)}$	(0)	0	1	—	$\{f(s_1^{(1)}); f(s_1^{(2)})\}$
		$s_1^{(2)}$	(1)	1	0	—	$= \{0.2; 1.2\}$
2	$s_1^{(1)}$	$s_2^{(1)}$	(00)	0	1	—	$\{f(s_2^{(1)}); f(s_2^{(2)}); f(s_1^{(2)})\}$
		$s_2^{(2)}$	(01)	1	0	—	$= \{0.2; 1.0; 1.2\}$
3	$s_2^{(1)}$	$s_3^{(1)}$	(000)	0	1	—	$\{f(s_3^{(1)}); f(s_3^{(2)}); f(s_2^{(2)}); f(s_1^{(2)})\}$
		$s_3^{(2)}$	(001)	1	0	—	$= \{0.2; 0.9; 1.0; 1.2\}$
4	$s_3^{(1)}$	$s_4^{(1)}$	(0001)	0	1	—	$\{f(s_4^{(1)}); f(s_4^{(2)}); f(s_3^{(2)}); f(s_2^{(2)}); f(s_1^{(2)})\}$
		$s_4^{(2)}$	(0000)	1	0	—	$= \{0.2; 0.7; 0.9; 1.0; 1.2\}$
5	$s_4^{(1)}$	— —	— —	— —	— —	(00010111) 0.9	$\{f(s_4^{(2)}) = 0.7\}$
6	$s_4^{(2)}$	— —	— —	— —	— —	(00000000) 0.7	$\emptyset$

size of the cost function list. This suboptimum version provides a good trade-off between error performance and decoding complexity with a proper choice of the maximum size of the cost function list. A very large number of computations can be saved at the expense of a very small degradation in performance, say a few tenths of a decibel.

The ancestor of the PFS algorithm was initially proposed by Dorsch in [22]. A clever implementation of this algorithm based on k lists was presented by Battail and Fang in [23]. These algorithms can be viewed as using the simplified cost function $f(s_i) \triangleq \lambda(\mathbf{y}, \mathbf{v}(s_i))$ in (10.81) and, as initially proposed, are therefore less efficient than the PFS algorithm. Recently, Valembois and Fossorier proposed a computational improvement of the Battail–Fang algorithm in [24]. In this method, a second ordered list of the visited nodes is updated, so that every time a node is visited, only one node is extended. As a result, this method requires less memory than the PFS algorithm and often succeeds in decoding blocks that result in memory overflow with the PFS algorithm. Its average computational complexity is slightly greater than that of the PFS algorithm but its variance is much lower in general.

One drawback of the PFS algorithm and similar algorithms is that the order in which the error patterns are processed in the list depends on each received sequence. In the next section, an algorithm with a fixed order of error-pattern processing is presented. This may represent an interesting feature for VLSI realizations of MRIP-reprocessing algorithms. Furthermore, it was shown in [25] that this structured reprocessing results in only a very small increase of average computational complexity with respect to that of the PFS algorithm.

10.8.3 Ordered Statistic Decoding Algorithm

MLD is, in general, achieved at the expense of great computational complexity, especially for long codes; however, if we do not insist on achieving optimum MLD performance, efficient soft-decision decoding algorithms can be devised to achieve near (or practically) optimum error performance with a significant reduction in decoding complexity.

Consider a suboptimum soft-decision decoding algorithm A. Let $P(A)$ be the probability that algorithm A commits a decoding error, whereas MLD is correct. Let P_B denote the block error probability of MLD as defined in (10.15). Then, the block error probability $P_B(A)$ of algorithm A is upper bounded by

$$P_B(A) \leq P_B + P(A). \tag{10.90}$$

If $P(A) << P_B$, then $P_B(A) \approx P_B$. In this case, algorithm A practically achieves the same error performance as MLD. Clearly, it is desirable to devise such a practically optimum decoding algorithm with a significant reduction in decoding complexity. One such decoding algorithm has been proposed in [26]. This algorithm is a MRIP-reprocessing algorithm that is devised to process the MRIPs progressively in stages based on the joint statistics of the noise after symbol reordering. These statistics are also used to evaluate the error performance achieved after each stage and therefore allow us to determine when practically optimum error performance or a desired level of error performance is achieved. This algorithm is referred to as an *ordered statistic decoding* (OSD) algorithm.

The OSD algorithm reprocesses the ordered received sequence $\mathbf{y}$ given by (10.72) based on the permuted code C_1 generated by the generator matrix $\mathbf{G}_1$ given by (10.73). Because the first k symbols of $\mathbf{y}$ are the k most reliable independent symbols, their hard decisions should contain very few errors. Based on this concept, the algorithm generates a sequence of candidate codewords for testing by processing

the k most reliable independent symbols of $\mathbf{y}$. The candidate codeword $\mathbf{v}^*$ with the least correlation discrepancy with $\mathbf{y}$ is the decoded codeword. Then, $\pi_1^{-1}[\pi_2^{-1}[\mathbf{v}^*]]$ gives the decoded codeword in C. For $0 \leq i \leq k$, the OSD algorithm of order-i executes the following steps:

1. Perform hard-decision decoding of the k most reliable independent symbols of $\mathbf{y}$ (the first k symbols of $\mathbf{y}$). These k hard decisions give k binary digits, which form an information sequence $\mathbf{u}_0$.
2. Construct the codeword $\mathbf{v}_0 = \mathbf{u}_0\mathbf{G}_1$ for the information sequence $\mathbf{u}_0$, and compute the correlation discrepancy $\lambda(\mathbf{y}, \mathbf{v}_0)$ of $\mathbf{v}_0$ with respect to $\mathbf{y}$.
3. For $1 \leq l \leq i$, make all possible changes of l of the k most reliable bits in $\mathbf{u}_0$. For each change, form a new information sequence $\mathbf{u}$. Generate its corresponding codeword $\mathbf{v} = \mathbf{u}\mathbf{G}_1$. Compute the correlation discrepancy $\lambda(\mathbf{y}, \mathbf{v})$ for each generated codeword. Record the codeword $\mathbf{v}_{best}$ that has the least correlation discrepancy. This step is referred to as the phase-l reprocessing of $\mathbf{u}_0$. It requires generating $\binom{k}{l}$ candidate codewords.
4. Start the next reprocessing phase and continue to update $\mathbf{v}_{best}$ until the ith reprocessing phase is completed. The recorded codeword $\mathbf{v}_{best}$ is the decoded codeword.

The OSD algorithm of order-i consists of $(i + 1)$ reprocessing phases and requires processing of a total of

$$1 + \binom{k}{1} + \cdots + \binom{k}{i}$$

candidate codewords to make a decoding decision. The OSD algorithm of order-k is MLD, which requires processing of 2^k codewords. This is not what we want. As pointed out earlier, the k first symbols of $\mathbf{y}$ are the k most reliable independent symbols, and their hard decisions most likely contain very few errors. That is to say that the information sequence $\mathbf{u}_0$ contains very few errors. Consequently, making all possible changes of a small number of positions of $\mathbf{u}_0$ most likely will produce the ML codeword. Therefore, an OSD algorithm with a small order-i should practically achieve the MLD error performance. It is shown in [26] that for most block codes of lengths up to 128 and rates $k/n \geq \frac{1}{2}$, an order of $i_0 \triangleq \lfloor d_{min}/4 \rfloor$ is enough practically to achieve the same error performance as MLD for bit-error rates larger than 10^{-6}. In this case, the two error performance curves basically fall on top of each other. If $i_0 = \lfloor d_{min}/4 \rfloor$ is smaller than k, then $1 + \binom{k}{1} + \cdots + \binom{k}{i_0}$ is much smaller than 2^k. Consequently, an OSD algorithm of order-i_0 results in a practically optimum error performance with a significant reduction in computational complexity. The OSD algorithm requires only an encoder to generate the candidate codewords for testing and is therefore very easy to implement.

Figures 10.10 and 10.11 depict the bit-error performances of the $(64, 42, 8)$ RM code and the $(128, 64, 22)$ extended BCH code with OSD algorithm of various orders, respectively. For the $(64, 42, 8)$ RM code, since $d_{min} = 8$, the order of reprocessing to achieve practically optimum error performance is $i_0 = \lfloor 8/4 \rfloor = 2$. Indeed, as shown in Figure 10.10, the OSD algorithm of order-2 practically achieves the MLD error

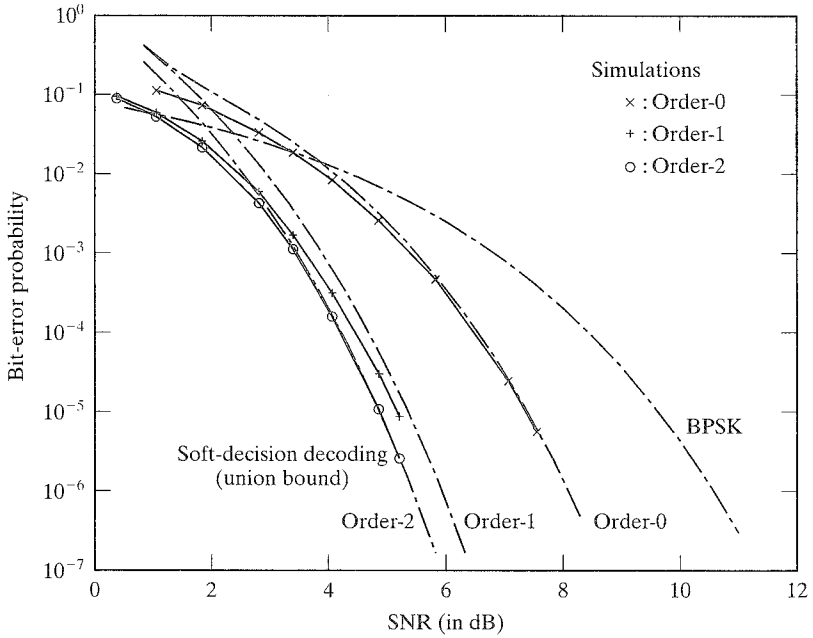

FIGURE 10.10: Order-i reprocessing for the (64, 42, 8) RM code.

performance. It requires generating $1 + \binom{42}{1} + \binom{42}{2} = 904$ candidate codewords and computing their correlation discrepancies. Figure 10.10 also shows that the OSD algorithm of order-1 gives an error performance only 0.3 dB away from MLD at the BER 10^{-5}. It requires generating only 43 candidate codewords. For the (128, 64, 22) extended BCH code, to achieve practically optimum error performance, the order of reprocessing required by the OSD algorithm is $i_0 = \lfloor 22/4 \rfloor = 5$; however, as shown in Figure 10.11, the OSD algorithm of order-4 is already within the union bound of the code. The OSD algorithm of order-4 requires generating a total of 679,121 candidate codewords, which is much smaller than 2^{64}. In these figures, tight bounds on the error performance of each reprocessing order are also included. These bounds correspond to (10.90), in which $P(A)$ is evaluated based on order statistics following the methods of [26–28].

The number of candidate codewords to be processed by the OSD algorithm of order-i can be drastically reduced by using a sufficient condition on optimality derived in Section 10.3. At the end of each reprocessing phase before the ith (last) one, the best candidate codeword $\mathbf{v}_{best}$ recorded is tested for optimality. If the sufficient condition on optimality is satisfied, then $\mathbf{v}_{best}$ is the ML codeword, and decoding can be terminated. The sufficient condition based on one candidate codeword given by (10.31) can be improved by taking into account the structure of the reprocessing method. Assume that for $l < i$, phase-l of order-i reprocessing has been completed. At this stage, all candidate codewords that differ from the hard-decision received vector $\mathbf{z}$ in at most l MRIPs have been processed. The remaining candidate codewords to be processed must differ from $\mathbf{z}$ in at least $(l + 1)$ MRIPs.

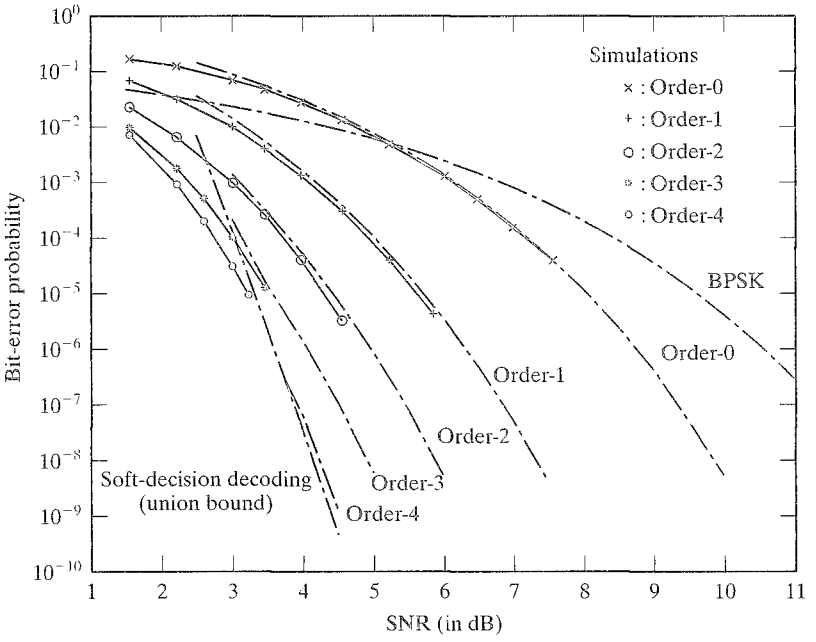

FIGURE 10.11: Order-i reprocessing for the (128, 64, 22) extended BCH code.

Based on this fact, the bound on optimality given by (10.31) can be improved as

$$G(\mathbb{v}_{best}, w_1) = \sum_{j=1}^{l+1} |y_{k-j}| + \sum_{j \in D_0^{(\delta')}(\mathbb{v}_{best})} |y_j|, \tag{10.91}$$

where

$$\delta' = \max\{0, d_{min} - |D_1(\mathbb{v}_{best})| - (l+1)\}. \tag{10.92}$$

The first term of (10.91), $\sum_{j=1}^{l+1} |y_{k-j}|$, represents the minimum contribution associated with inverting at least $(l+1)$ MRIPs in $\mathbb{z}$, and the second term is derived in the same way as for (10.31). Comparing (10.91) with (10.31), we see that LRPs in (10.31) are replaced with MRPs in (10.91). Consequently, from Theorem 10.1 (or (10.33)), we see that (10.91) provides a stronger (or less stringent) sufficient condition for optimality than (10.31).

Further computation savings can be achieved by improving the optimality threshold $G(\mathbb{v}_{best}, w_1)$ of (10.91) in a number of ways. First, in evaluating (10.91), we can use a processed codeword $\mathbb{v}$ other than $\mathbb{v}_{best}$ if $G(\mathbb{v}, w_1)$ is larger than $G(\mathbb{v}_{best}, w_1)$. Second, if $\mathbb{v}_{best} \neq \mathbb{v}_0$, we can consider the MRIPs of $\mathbb{z}$ inverted in generating $\mathbb{v}_{best}$ in evaluating $G(\mathbb{v}_{best}, w_1)$. Third, we may use a sufficient condition on optimality based on two candidate codewords, as in (10.46), in conjunction with the reprocessing of the MRIPs in $\mathbb{z}$ [29]. Finally, we can use the monotonicity of the reliability values in the MRIPs to discard candidate codewords within each reprocessing phase [26, 30].

The effectiveness of using a sufficient condition for optimality test to terminate the decoding process of the OSD algorithm is best demonstrated by an example.

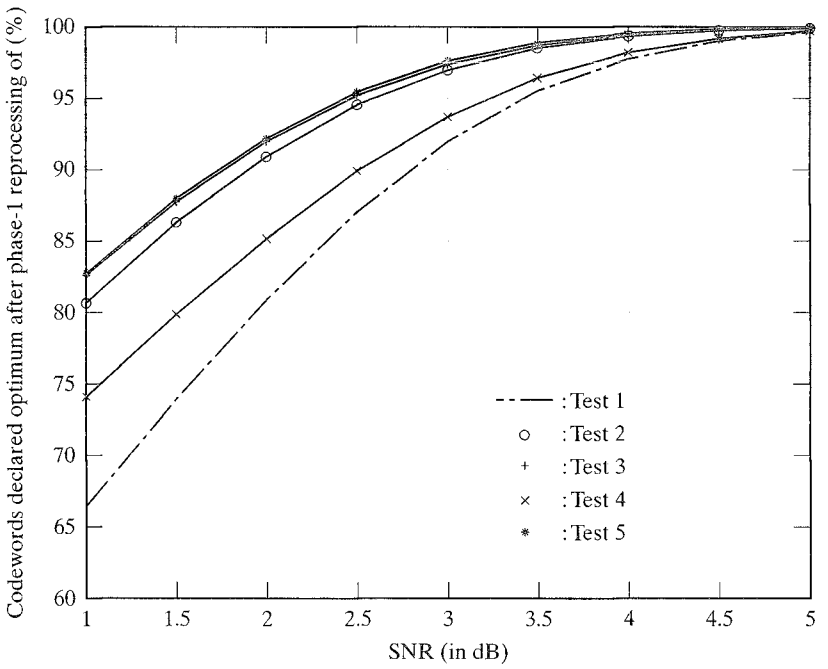

FIGURE 10.12: Percentage of codewords declared optimum after phase-1 of order-2 reprocessing for the (24, 12, 8) Golay code.

Consider the (24, 12, 8) Golay code (presented in Chapter 4). For this code, an order-2 OSD algorithm achieves practically optimum error performance. Figure 10.12 depicts the percentage of codewords that are declared optimum after phase-1 reprocessing for five different cases of applications of sufficient conditions on optimality. For the first three cases, optimality is determined based on one candidate codeword, and for the remaining two cases, optimality is determined based on two candidate codewords. In the first case, the optimality threshold $G(\mathbf{v}, w_1)$ is evaluated based on (10.31) by considering the recorded codeword $\mathbf{v}_{best}$ at the end of phase-1 reprocessing. In the second case, $G(\mathbf{v}, w_1)$ is computed from (10.91) for $l = 1$ based on $\mathbf{v}_{best}$. The third case is similar to the second case, except that $G(\mathbf{v}, w_1)$ is evaluated based on the codeword that minimizes $G(\mathbf{v}, w_1)$ of (10.91) among the 13 candidate codewords generated at the end of phase-1 reprocessing. For the fourth case, the optimality threshold $G(\mathbf{v}_1, w_1; \mathbf{v}_2, w_1)$ is evaluated based on (10.46) by considering the two codewords that have the smallest correlation discrepancies among the 13 candidate codewords generated at the end of phase-1 reprocessing. The fifth case is similar to the fourth case, except that the optimality threshold $G(\mathbf{v}_1, w_1; \mathbf{v}_2, w_1)$ is computed from a generalization of (10.91) for two codewords [29]. Figure 10.12 shows that (10.91) provides a more efficient optimality condition for terminating the decoding process than (10.31). In terms of cost effectiveness, case-3 is the most effective one, as it is easier to implement than case-5. For case-3, 83% of the time, the decoding process is terminated at the end of phase-1 reprocessing at the SNR of 1 dB. This result means that even at a low SNR of 1 dB, only 22

candidate codewords need to be processed on average for decoding a received sequence.

EXAMPLE 10.5

Consider order-1 reprocessing of the received sequence given in Example 10.4 for the (8, 4, 4) RM code. At phase-0, we find $\mathbb{v}_0 = \mathbb{v}_{best} = (0, 0, 0, 1, 0, 1, 1, 1)$, and $\lambda(\mathbb{y}, \mathbb{v}_{best}) = 0.9$. From (10.91) and (10.92), we find that for $l = 0$, $\delta' = 0$, and $G(\mathbb{v}_{best}, w_1) = 0.7$. Because $\lambda(\mathbb{y}, \mathbb{v}_{best}) > G(\mathbb{v}_{best}, w_1)$, decoding must be carried into phase-1 reprocessing. At the end of phase-1 reprocessing, we find the codeword $\mathbb{v} = (0, 0, 0, 0, 0, 0, 0, 0)$, with $\lambda(\mathbb{y}, \mathbb{v}) = 0.7$. Since $\lambda(\mathbb{y}, \mathbb{v}) < \lambda(\mathbb{y}, \mathbb{v}_{best})$, we set $\mathbb{v}_{best} = (0, 0, 0, 0, 0, 0, 0, 0)$. From (10.91) and (10.92), we find that for $l = 1$, $\delta' = 1$, and $G(\mathbb{v}_{best}, w_1) = 0.9 + 0.7 + 0.2 = 1.8$. Because $\lambda(\mathbb{y}, \mathbb{v}_{best}) < 1.8$, $\mathbb{v}_{best}$ is the MLD codeword.

In [31], order-i reprocessing is further improved by considering only a reduced probabilistic list of the most probable candidate codewords instead of the entire set composed of the $\sum_{l=0}^{i} \binom{k}{l}$ candidates associated with order-i. For many codes, this method greatly reduces the number of candidate codewords reprocessed by phase-i of order-i reprocessing and still achieves practically optimum error performance at BER values larger than 10^{-6}. For codes of rate $k/n \leq \frac{1}{2}$, many positions outside the MRB still have reasonably good reliability measures. Consequently, for such codes, order-j reprocessing of few information sets becomes more efficient than order-i reprocessing of the MRB, with $j < i$. A general presentation of order-j reprocessing of several information sets is given in [32]. This method was further improved in [33], where an iterative information set reduction is used to closely achieve the error performance of order-$(i + 1)$ reprocessing with a decoding cost closer to that of order-i reprocessing. Finally, Valembois and Fossorier proposed a new version of OSD called "Box-and-Match" (BMA) decoding [34]. The BMA algorithm roughly achieves the error performance of order-$(2i)$ reprocessing with a decoding cost closer to that of order-i reprocessing but with a memory requirement of the same order (whereas OSD has no memory requirement). In contrast with the OSD algorithm, which processes error patterns independently of each other, the BMA stores some partial information about the error patterns processed at phase-j to be used at phase-j', with $j' > j$. As a result, many error patterns processed at phase-j' by the OSD algorithm are discarded by the BMA algorithm, and the BMA algorithm can achieve near-MLD of larger codes than the OSD algorithm.

The OSD algorithm and the Chase-type decoding algorithm are complementary in nature: the former is based on processing certain MRPs of a received sequence, and the latter is based on processing certain LRPs. This complementary feature allows us to combine the two algorithms to form a hybrid decoding algorithm that has the advantages of both algorithms. Consider the OSD algorithm with order-i reprocessing. If there are at least $i + 1$ MRIPs of the ordered received sequence in error, but fewer than $t + 1$ errors have occurred outside the $\lfloor d_{\min}/2 \rfloor$ LRPs, then the OSD algorithm with order-i reprocessing will result in an incorrect decoding; however, Chase algorithm-2 will result in a correct decoding. On the other hand, if the number of errors in the LRPs is large, but the number of errors in the

MRPs remains less than $i + 1$, Chase algorithm-2 will fail to decode the received sequence correctly, but the OSD algorithm with order-i reprocessing will result in a correct decoding. Based on these results, we can combine these two algorithms to process the received sequence in parallel. At the end of the processing, there are two candidate codewords, one produced by the Chase decoder and one produced by the OS decoder. These two final candidate codewords are then compared, and the one with the larger correlation (or smaller correlation discrepancy) is chosen as the final decoded codeword. It has been shown in [35] that for $i < \lfloor d_{min}/4 \rfloor$, this hybrid decoding improves the error performance of the OSD algorithm with order-i reprocessing at medium to high SNR.

The OSD algorithm also can be combined with the PFS decoding algorithm presented in Section 10.8.2 to limit the size of the cost function list. In the PFS decoding algorithm, only the cost functions $f(s_i)$ that correspond to inverting at most i MRIPs can be added to the cost function list, either from the beginning of the reprocessing or at a later stage to first exploit the dynamic nature of the PFS algorithm. In general, a probabilistic decoding method is associated with two related problems: (1) converging as quickly as possible to the optimum solution and (2) recognizing the optimum solution that has been found and stopping the decoding. For MLD, the PFS algorithm converges faster than the OSD algorithm to the optimum solution owing to its dynamical behavior; however, the structured reprocessing strategy of the OSD algorithm allows us to derive a less stringent sufficient condition for optimality of the decoded codewords. These issues are elaborated in [25].

In this section we described a probabilistic decoding method that achieves either a desired error performance or practically the optimal error performance of MLD for a given range of SNR values. Also using a probabilistic approach, Dumer proposed in [36] a general approach for soft-decision decoding that, at most, doubles the block error probability P_B for MLD *at all SNR values*. This method has also been extended to MLD [37]. The decoding complexity of this method is shown to be less than that of trellis-based MLD, suggesting that probabilistic soft-decision decoding algorithms seem to represent a promising approach for optimum or near-optimum decoding of block codes with lengths of hundreds of bits.

10.8.4 Syndrome-Based Ordered Statistic Decoding Algorithm

The OSD algorithm can be formulated in terms of syndrome like the syndrome-based MLD decoding algorithm presented in Section 10.7. Consider the parity-check matrix $\mathbb{H}_1 = [\mathbb{P}^T \mathbb{I}_{n-k}]$ for the code C_1 generated by the permuted generator matrix $\mathbb{G}_1$ in the MRB given by (10.73). Again, let $\mathbf{z}$ be the hard-decision received vector obtained from the permuted received sequence $\mathbf{y}$. Let $\mathbf{s} = \mathbf{z}\mathbb{H}_1^T$ be the syndrome of $\mathbf{z}$. Consider a set Q_l of l columns of $\mathbb{P}^T$. Let

$$\mathbf{s}_1 = (s_{1,0}, s_{1,1}, \cdots, s_{1,n-k-1})$$

be an $(n - k)$-tuple obtained by taking the transpose of the sum of the l vectors in Q_l. Let $\mathbf{s}_2 = \mathbf{s} + \mathbf{s}_1$. We form the following error vector $\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2)$, where $\mathbf{e}_1 = (e_{1,0}, e_{1,1}, \cdots, e_{1,k-1})$ with $e_{1,j} = 1$ if $j \in Q_l$, and $e_{1,j} = 0$ otherwise, and $\mathbf{e}_2 = \mathbf{s}_2$. Then, $\mathbf{v} = \mathbf{z} + \mathbf{e}$ is a codeword in C. MLD is achieved by finding the error vector $\mathbf{e}^*$ such that $\mathbf{v}^* = \mathbf{z} + \mathbf{e}^*$ has the smallest correlation discrepancy with $\mathbf{y}$.

The syndrome-based OSD algorithm of order i consists of the following steps:

1. Compute the syndrome $\mathbf{s} = \mathbf{z}\mathbf{H}_1^T$.
2. For $1 \leq l \leq i$, form all possible sets Q_l of l columns of $\mathbb{P}^T$. For each set Q_l, form the vector sum of the l columns. Take the transpose of this vector to obtain $\mathbf{s}_1$. From $\mathbf{s}$ and $\mathbf{s}_1$, form $\mathbf{s}_2 = \mathbf{s} + \mathbf{s}_1$ and the error vector $\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2)$. For each error vector $\mathbf{e}$, form the codeword $\mathbf{v} = \mathbf{z} + \mathbf{e}$ and evaluate its correlation discrepancy $\lambda(\mathbf{y}, \mathbf{v})$. Find and record the codeword $\mathbf{v}_{best}$ that has the smallest correlation discrepancy. This completes phase-l reprocessing.
3. Start the next reprocessing phase and continue to update $\mathbf{v}_{best}$ until the ith reprocessing phase is completed. The recorded codeword $\mathbf{v}_{best}$ is the decoded codeword.

This formulation of order-i reprocessing in terms of syndrome provides a structured search for the syndrome-based MLD presented in Section 10.7.

10.9 WEIGHTED MAJORITY-LOGIC DECODING

In all the algorithms presented previously, the reliability information of the received symbols is used to reduce the search space of MLD. This reliability information can also be used to improve the error performance of algebraic decoding algorithms while maintaining their simplicity. As a result, a good trade-off between error performance and decoding complexity is achieved, and generally, such decoding methods are suitable for high-speed decoding. Now, we present the use of reliability information to decode majority-logic decodable codes. For simplicity, we consider RM codes.

10.9.1 Majority-Logic Decoding of RM Codes over the Binary Symmetric Channel (BSC)

In Section 4.2 we showed that for any nonnegative integers r and m, with $r < m$, there exists a binary rth-order RM code, denoted by RM(r, m), of length $n = 2^m$, minimum Hamming distance $d_{min} = 2^{m-r}$, and dimension $k(r, m) = \sum_{i=0}^{r} \binom{m}{i}$. Suppose this RM$(r, m)$ code is used for error control over the AWGN channel. For BPSK transmission, a codeword $\mathbf{v} = (v_0, v_1, \cdots, v_{n-1})$ is mapped into a bipolar sequence $\mathbf{c} = (c_0, c_1, \cdots, c_{n-1})$ with $c_i = 2v_i - 1$. After transmission, the received sequence at the output of the correlation detector is $\mathbf{r} = (r_0, r_1, \cdots, r_{n-1})$. For hard-decision decoding, $\mathbf{r}$ is converted into a binary sequence $\mathbf{z} = (z_0, z_1, \cdots, z_{n-1})$ with $z_i = 0$ for $r_i < 0$, and $z_i = 1$ otherwise. For majority-logic decoding, a set S_j of check-sums for each information bit u_j, $0 \leq j \leq k-1$, is formed. Each check-sum is a modulo-2 sum of the hard-decision decoded bits z_i of certain received symbols r_i (see Chapter 4). For RM codes, the sets of bits in any two distinct check-sums are disjoint. The check-sums in S_j form a set of independent estimates of the information bit u_j. Next, we briefly review the main steps of majority-logic decoding of RM codes as detailed in Section 4.2, but now we closely follow the approach of [38]. This approach allows us to extend the hard-decision majority-logic decoding to soft-decision (or weighted) majority-logic decoding.

For equiprobable signaling, the transmitted information bit u_j is decoded into its estimate $\hat{u}_j \in \{0, 1\}$, which maximizes the conditional probability $P(S_j \mid u_j)$. Let $S_j = \{A_{j,l} : 1 \le l \le |S_j|\}$, where $A_{j,l}$ denotes the lth check-sum in S_j. Because the check-sums in S_j are disjoint, we obtain

$$\log\left(P(S_j \mid u_j)\right) = \sum_{l=1}^{|S_j|} \log\left(P(A_{j,l} \mid u_j)\right). \tag{10.93}$$

An algorithm that maximizes (10.93) is referred to as a *maximum a posteriori probability* (MAP) decoding algorithm (or simply *a posteriori probability* (APP) decoding algorithm [38]). Let γ_l be the probability that the number of independent errors in the check-sum $A_{j,l}$ is odd. Then, it is shown in [38] that

$$\begin{aligned} \gamma_l &= P(A_{j,l} = 0 \mid u_j = 1) = P(A_{j,l} = 1 \mid u_j = 0) \\ &= \frac{1}{2}\left[1 - \prod_{i \in B_l(j)} (1 - 2\,p_i)\right], \end{aligned} \tag{10.94}$$

where $B_l(j)$ represents the set of positions associated with the digits constituting the check-sum $A_{j,l}$, and p_i is the probability that the ith digit is in error. It is clear that γ_l is simply the probability that the check-sum $A_{j,l}$ is different from the transmitted information bit u_j.

For the BSC with crossover probability $p_i = p = Q(\sqrt{2E_b/N_0})$, γ_l is the same for all l and independent of j. It follows from (10.93) that [38]

$$\sum_{l=1}^{|S_j|} \log\left(\frac{P(A_{j,l} \mid u_j = 1)}{P(A_{j,l} \mid u_j = 0)}\right) = \sum_{l=1}^{|S_j|} (2A_{j,l} - 1) \log\left(\frac{1 - \gamma_l}{\gamma_l}\right). \tag{10.95}$$

Because $\log((1 - \gamma_l)/\gamma_l)$ is a positive constant, the decision rule is then

$$\sum_{l=1}^{|S_j|} (2A_{j,l} - 1) \underset{>}{\overset{<}{}} 0, \tag{10.96}$$

which simply compares $\sum_{l=1}^{|S_j|} A_{j,l}$ with $|S_j|/2$. If $\sum_{l=1}^{|S_j|} A_{j,l} > |S_j|/2$, we set the estimate $\hat{u}_j$ of u_j to 1, and if $\sum_{l=1}^{|S_j|} A_{j,l} < |S_j|/2$, we set $\hat{u}_j = 0$. Whenever $\sum_{l=1}^{|S_j|} A_{j,l} = |S_j|/2$, we flip a fair coin to determine $\hat{u}_j$, which results in an erroneous decision in half of the cases on average. This type of error dominates the error performance. The foregoing decoding is the conventional majority-logic decoding. We readily observe that this decoding method is exactly the same as the Reed algorithm presented in Section 4.2.

10.9.2 Majority-Logic Decoding Based on Reliability Information

For the AWGN channel model, p_i is defined as [38]

$$p_i = \frac{e^{-|L_i|}}{1 + e^{-|L_i|}}, \tag{10.97}$$

where $L_i = 4r_i/N_0$ represents the log-likelihood ratio associated with the received symbol r_i. The optimum decision rule based on (10.97) becomes computationally expensive and depends on the operating SNR [38]. Using the approximation

$$\prod_{i \in B_l(j)} (1 - 2\ p_i) \approx 1 - 2 \max_{i \in B_l(j)} p_i, \tag{10.98}$$

we obtain

$$\gamma_l \approx \frac{e^{-4|r_l|_{min}/N_0}}{1 + e^{-4|r_l|_{min}/N_0}}, \tag{10.99}$$

where $|r_l|_{min} = \min_{i \in B_l(j)} \{|r_i|\}$. It follows from (10.95) that for the AWGN channel, the decision rule based on the approximation (10.99) becomes

$$\sum_{l=1}^{|S_j|} (2A_{j,l} - 1)\ |r_l|_{min} \underset{>}{\overset{<}{}} 0. \tag{10.100}$$

In (10.100), each term of (10.96) is weighted with the reliability value of the least reliable digit in its corresponding check-sum. If $\sum_{l=1}^{|S_j|}(2A_{j,l} - 1)\ |r_l|_{min} > 0$, we set the estimate $\hat{u}_j$ of u_j to 1, and if $\sum_{l=1}^{|S_j|}(2A_{j,l} - 1)\ |r_l|_{min} < 0$, we set $\hat{u}_j = 0$. Whenever $\sum_{l=1}^{|S_j|}(2A_{j,l} - 1)\ |r_l|_{min} = 0$, we flip a fair coin to determine $\hat{u}_j$; however, the occurrence of such events becomes less and less probable as the number of quantization levels used at the receiver increases. This decoding algorithm, referred to as *weighted majority-logic decoding*, was first proposed in [39].

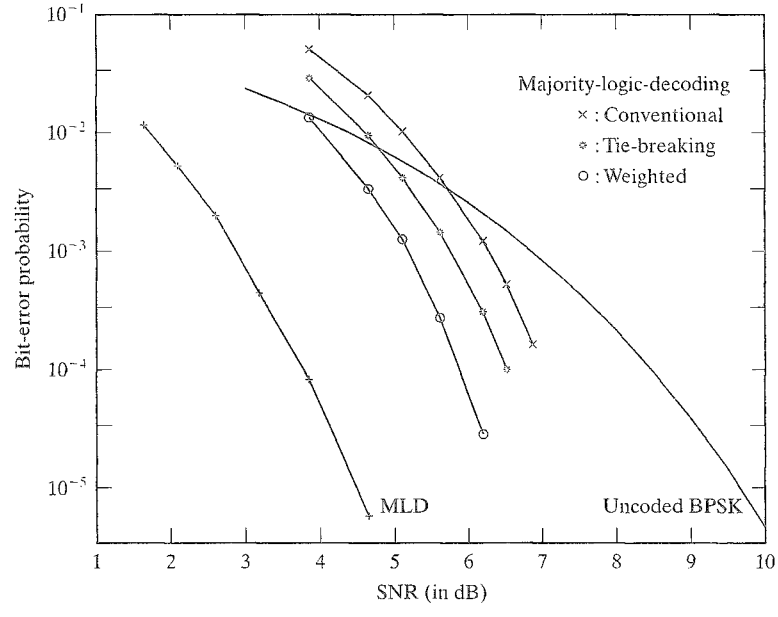

FIGURE 10.13: Conventional and weighted majority-logic decoding for the (64, 22, 16) RM code.

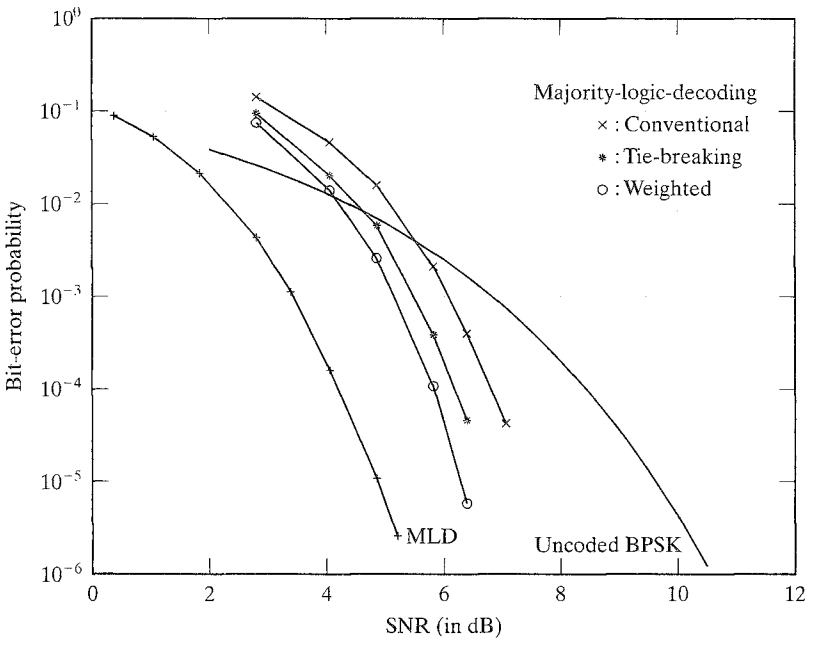

FIGURE 10.14: Conventional and weighted majority-logic decoding for the (64, 42, 8) RM code.

As mentioned in Section10.9.1, the error performance of the conventional majority-logic decoding is dominated by the event $\sum_{l=1}^{|S_j|} A_{j,l} = |S_j|/2$. In such a case, we can break the tie by using (10.100) instead of flipping a fair coin. A simplified version of this tie-breaking rule is obtained by simply discarding the check-sum with the smallest reliability value whenever $\sum_{l=1}^{|S_j|} A_{j,l} = |S_j|/2$ in the Reed algorithm. Such a test requires very few additional computations.

Figures 10.13 and 10.14 depict the simulation results for conventional and weighted majority-logic decoding of the (64, 22, 16) and (64, 42, 8) RM codes, respectively. Conventional majority-logic decoding in conjunction with (10.100) to break the ties is also presented. We observe that both soft-decision approaches provide significant coding gains over conventional majority-logic decoding while maintaining its simplicity and decoding speed.

Another version of weighted majority-logic decoding based on the parity-check sums computed from a parity-check matrix of a one-step majority-logic decodable code will be presented in Chapter 17.

10.10 ITERATIVE RELIABILITY-BASED DECODING OF ONE-STEP MAJORITY-LOGIC DECODABLE CODES

10.10.1 Iterative MAP-Based Decoding

Suppose a one-step majority-logic decodable code (see Chapter 8), or a low-density parity-check (LDPC) code (an LDPC code is defined as the null space of a

parity-check matrix H with low density of nonzero entries (or 1's), that is, a sparse matrix [45]) (see Chapter 17) is used for error control over an AWGN channel. Then, this code can be decoded based on the approach presented in Section 10.9.2. In general, the algorithm computes the a posteriori probability $P(S_i \mid v_i)$ associated with each bit v_i, $0 \leq i \leq n-1$, from the corresponding a priori probabilities p_i based on (10.93) and (10.94). Because for the classes of codes considered, this decoding is completed in one step, a natural extension is to use the set of a posteriori probabilities $P(S_i \mid v_i)$ as new a priori probabilities p_i and to repeat the entire process. As a result, we obtain a simple iterative decoding method based on the following general procedure. Let J_{max} denote the maximum number of iterations to be performed:

1. Construct the hard-decision received sequence $\mathbf{z}$ from $\mathbf{r}$.
2. Initialize the decoding process by setting $p_i^{(0)} = p_i$, for $i = 0, 1, \ldots, n-1$, and $\mathbf{z}^{(0)} = \mathbf{z}$.
3. For $j = 1, 2, \ldots, J_{max}$,
   a. compute $\mathbb{p}^{(j)} = (p_0^{(j)}, p_1^{(j)}, \cdots, p_{n-1}^{(j)})$ as the vector of a posteriori error probabilities using $\mathbb{p}^{(j-1)}$ as the vector of a priori error probabilities;
   b. if $p_i^{(j)} > 0.5$, then set $z_i^{(j)} = z_i^{(j-1)} \oplus 1$ and $p_i^{(j)} = 1 - p_i^{(j)}$, for $i = 0, 1, \ldots, n-1$.
4. Set the estimated word $\mathbf{v}^* = \mathbf{z}^{(J_{max})}$.

In this iterative algorithm the average number of iterations can be reduced by checking at the end of each step j whether the syndrome computed from $\mathbf{z}^{(j)}$ is zero. In this case, we can directly set $\mathbf{v}^* = \mathbf{z}^{(j)}$.

For the algorithm presented in Section 10.9.2 based on (10.100), we obtain the following iterative procedure:

1. Construct the hard-decision received sequence $\mathbf{z}$ from $\mathbf{r}$.
2. Set $|r_i^{(0)}| = |r_i|$, $i = 0, 1, \ldots, n-1$, and $\mathbf{z}^{(0)} = \mathbf{z}$.
3. For $j = 1, 2, \ldots, J_{max}$, for $0 \leq i \leq n-1$, and for $1 \leq l \leq |S_i|$, compute the check-sums $A_{i,l}$ based on the values $z_i^{(j-1)}$'s, and identify
$$|r_{i,l}^{(j-1)}|_{min} = \min_{m \in B_l(i) \setminus i} \{|r_m^{(j-1)}|\}.$$
Then, compute
$$r_i^{(j)} = |r_i^{(0)}| + \sum_{l=1}^{|S_i|} (\overline{A}_{i,l} - A_{i,l})\, |r_{i,l}^{(j-1)}|_{min},$$
where $\overline{A}_{i,l}$ represents the complementary value of the check-sum $A_{i,l}$.
If $r_i^{(j)} < 0$, then set $z_i^{(j)} = z_i^{(j-1)} \oplus 1$.
4. Set the estimated word $\mathbf{v}^* = \mathbf{z}^{(J_{max})}$.

This algorithm represents the simplest implementation of iterative soft-decision decoding of either one-step majority-logic decodable codes or LDPC codes. Because

this algorithm is independent of the operating SNR, it is referred to as the *uniformly most powerful* (UMP) MAP-based decoding algorithm. Other iterative decoding algorithms based on the same general approach have been proposed in [40–42]. In general, such algorithms converge with relatively few iterations (say, between 10 and 20) but have a nonnegligible error performance degradation with respect to optimum MLD or MAP decoding owing to the increasing correlations between the decisions made at each step.

To reduce the effects of correlated decisions, it was proposed in [43] to combine this approach with variable threshold decoding. In the algorithm of [43], only the most likely errors (i.e., those whose associated a posteriori probabilities are larger than a certain threshold) are corrected, and the variable threshold is reduced during the decoding iteration process. As a result, the error performance improves at the expense of a larger number of iterations. A second approach, which almost eliminates correlations between decisions made at successive steps, is presented next.

10.10.2 Iterative Belief-Propagation-Based Decoding

We first use a simple example to illustrate the correlations between decisions made at successive iteration steps. Consider three check-sums defined by three codewords of Hamming weight 2, $\mathbf{v}_1 = (v_{1,0}, v_{1,1}, \cdots, v_{1,n-1})$, $\mathbf{v}_2 = (v_{2,0}, v_{2,1}, \cdots, v_{2,n-1})$, and $\mathbf{v}_3 = (v_{3,0}, v_{3,1}, \cdots, v_{3,n-1})$, in the dual code, with $v_{1,i} = v_{1,l} = 1$, $v_{2,i} = v_{2,x} = 1$, and $v_{3,l} = v_{3,y} = 1$. At the jth iteration, the general algorithm presented in Section 10.10.1 computes

$$p_i^{(j)} = \tilde{f}(p_i^{(0)}) + \left(f(p_l^{(j-1)}) + f(p_x^{(j-1)})\right), \tag{10.101}$$

$$p_l^{(j)} = \tilde{f}(p_l^{(0)}) + \left(f(p_i^{(j-1)}) + f(p_y^{(j-1)})\right), \tag{10.102}$$

where $\tilde{f}(\cdot)$ and $f(\cdot)$ are two functions defined by the specific algorithm considered. Similarly, at the $(j+1)$th iteration, we have

$$\begin{aligned} p_i^{(j+1)} &= \tilde{f}(p_i^{(0)}) + \left(f(p_l^{(j)}) + f(p_x^{(j)})\right) \\ &= \tilde{f}(p_i^{(0)}) + \left(f\left(\tilde{f}(p_l^{(0)}) + f(p_i^{(j-1)}) + f(p_y^{(j-1)})\right) + f(p_x^{(j)})\right). \end{aligned} \tag{10.103}$$

Because the a posteriori probability $p_l^{(j)}$ for bit v_l at the jth iteration is evaluated based on some information from bit v_i, as shown in (10.102), this value becomes correlated with $p_i^{(0)}$. As a result, $p_i^{(0)}$ and $p_l^{(j)}$ can no longer be assumed to be independent when used to evaluate $p_i^{(j+1)}$, as shown in (10.103).

To overcome this problem, for each position-i, $0 \le i \le n-1$, $|S_i|$ a posteriori probabilities $p_{i,l}^{(j)}$ rather than one are computed at the jth iteration, where for $A_{i,l} \in S_i$, $p_{i,l}^{(j)}$ represents the a posteriori probability obtained by considering the $|S_i| - 1$ check-sums other than $A_{i,l}$ in S_i. For simplicity, in this particular example we represent instead with $p_{i,l}^{(j)}$ the a posteriori probability obtained by discarding

the check-sum of S_i containing bit v_l. As a result, (10.101) and (10.102) become

$$\begin{aligned} p_{i,l}^{(j)} &= \tilde{f}(p_i^{(0)}) + f(p_{x,i}^{(j-1)}), \\ p_{i,x}^{(j)} &= \tilde{f}(p_i^{(0)}) + f(p_{l,i}^{(j-1)}), \\ p_{l,i}^{(j)} &= \tilde{f}(p_l^{(0)}) + f(p_{y,l}^{(j-1)}), \\ p_{l,y}^{(j)} &= \tilde{f}(p_l^{(0)}) + f(p_{i,l}^{(j-1)}), \end{aligned}$$

and (10.103) is replaced with

$$\begin{aligned} p_{i,l}^{(j+1)} &= \tilde{f}(p_i^{(0)}) + f(p_{x,i}^{(j)}), \\ p_{i,x}^{(j+1)} &= \tilde{f}(p_i^{(0)}) + f(p_{l,i}^{(j)}) \\ &= \tilde{f}(p_i^{(0)}) + f\left(\tilde{f}(p_l^{(0)}) + f(p_{y,l}^{(j-1)})\right). \end{aligned} \tag{10.104}$$

Hence, no correlated values appear in (10.104). If a decision about bit v_i has to be made at the $(j+1)$th iteration, then the a posteriori probability

$$p_i^{(j+1)} = \tilde{f}(p_i^{(0)}) + f(p_{l,i}^{(j)}) + f(p_{x,i}^{(j)})$$

is considered, as it contains all available information evaluated separately about bit v_i. The general form of this algorithm is known as the *belief propagation* (BP) *algorithm* and was introduced in [44]; however, the application of this algorithm to decoding LDPC codes can be found in [45] and was recently described as an instance of the BP algorithm in [46–48]. A good description of this algorithm can be found in [46]. The BP approach also can be applied to the decoding of one-step majority-logic decodable codes [50].

For LDPC codes, a simplified version of the BP algorithm based on the approximation (10.99) has been proposed in [49]. This algorithm can be summarized as follows:

1. For each i, $0 \le i \le n-1$, and each l, $1 \le l \le |S_i|$, the hard decisions $z_{l,i}^{(0)}$ and $z_i^{(0)}$ are initialized with the hard-decision decoding z_i of the received symbols r_i; also $|r_i^{(0)}| = |r_i|$, and for each l, $1 \le l \le |S_i|$, $|r_{i,l}^{(0)}| = |r_i^{(0)}|$.
2. The jth iteration with $1 \le j \le J_{max}$ consists of the following steps:
   a. For each i, $0 \le i \le n-1$ and each l, $1 \le l \le |S_i|$, evaluate the check-sums

$$A_{i,l} = z_i^{(j-1)} \oplus \left(\sum_{i' \in B_l(i) \setminus i} z_{i',l}^{(j-1)} \right), \tag{10.105}$$

   and identify

$$|r_{i,l}^{(j)}|_{min} = \min_{i' \in B_l(i) \setminus i} \{|r_{i',l}^{(j-1)}|\}. \tag{10.106}$$

b. For each i, $0 \leq i \leq n-1$, and each l, $1 \leq l \leq |S_i|$, compute

$$r_{i,l}^{(j)} = |r_i^{(0)}| + \sum_{A_{i,l'} \in S_i \setminus A_{i,l}} (\overline{A}_{i,l'} - A_{i,l'}) |r_{i,l'}^{(j)}|_{min}. \tag{10.107}$$

c. For each i, compute

$$r_i^{(j)} = |r_i^{(0)}| + \sum_{l=1}^{|S_i|} (\overline{A}_{i,l} - A_{i,l}) |r_{i,l}^{(j)}|_{min}. \tag{10.108}$$

d. Form $\mathbf{z}^{(j)} = (z_0^{(j)}, z_1^{(j)}, \cdots, z_{n-1}^{(j)})$ such that $z_i^{(j)} = z_i^{(0)}$ if $r_i^{(j)} > 0$, and $z_i^{(j)} = z_i^{(0)} \oplus 1$ if $r_i^{(j)} < 0$.

e. For each i, $0 \leq i \leq n-1$, and each l, $1 \leq l \leq |S_i|$, evaluate $z_{i,l}^{(j)}$ such that $z_{i,l}^{(j)} = z_i^{(0)}$ if $r_{i,l}^{(j)} > 0$, and $z_{i,l}^{(j)} = z_i^{(0)} \oplus 1$ if $r_{i,l}^{(j)} \leq 0$.

3. Set the estimated word to $\mathbf{v}^* = \mathbf{z}^{(J_{max})}$.

This algorithm is referred to as the UMP BP-based decoding algorithm, as it is independent of the operating SNR. In general, it improves the error performance of the algorithm presented in Section 10.10.1 at the expense of a larger number of iterations. Figures 10.15 and 10.16 show the bit-error performance for iterative

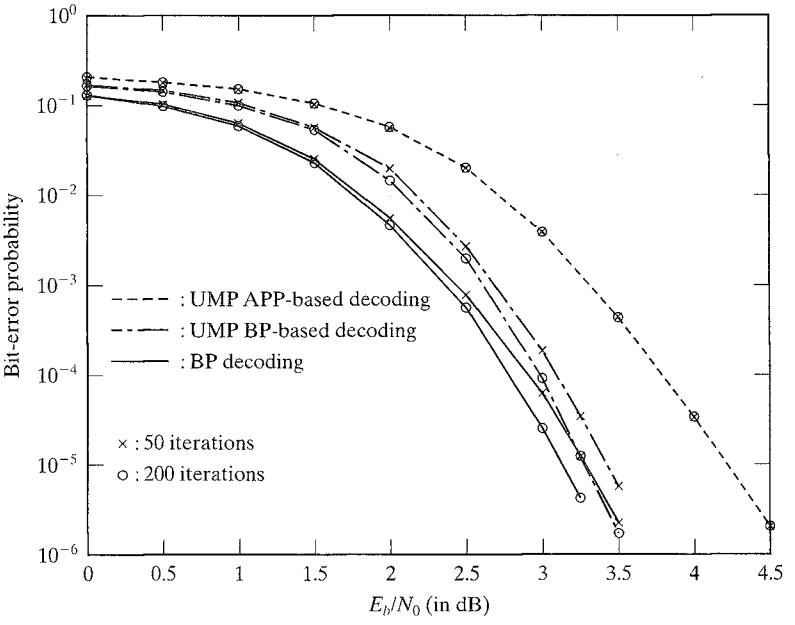

FIGURE 10.15: Error performance for iterative decoding of the (504, 252) LDPC code with BP, UMP BP-based, and UMP APP-based decoding algorithms, and at most 50 and 200 iterations.

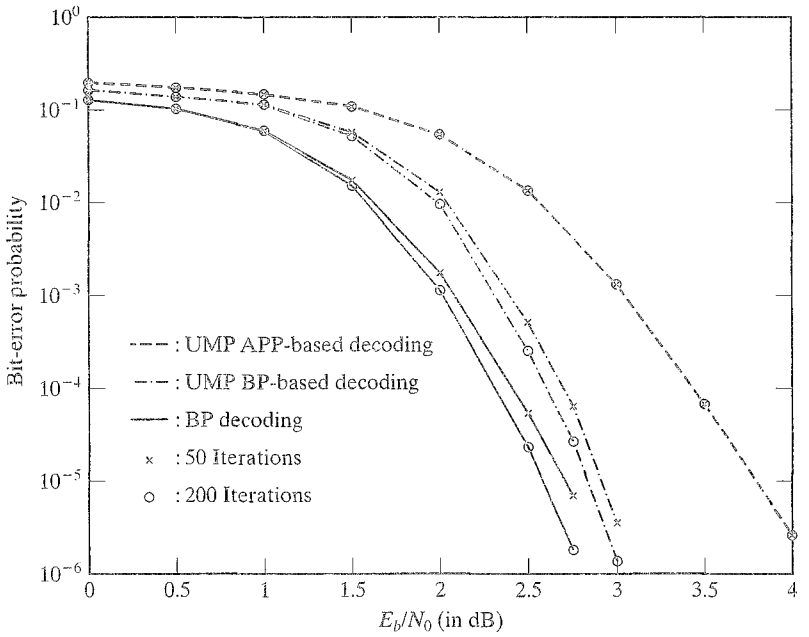

FIGURE 10.16: Error performance for iterative decoding of the (1008, 504) LDPC code with BP, UMP BP-based, and UMP APP-based decoding algorithms, and at most 50 and 200 iterations.

decoding of the (504, 252) and (1008, 504) LDPC codes (constructed by computer search), respectively, with the BP, UMP BP-based, and UMP APP-based decoding algorithms, and at most 50 and 200 iterations. The (504, 252) LDPC code has three check-sums of weight 6 orthogonal on each position, and the (1008, 504) LDPC code has four check-sums of weight 8 orthogonal on each position. We observe that the error performance of the simplified UMP BP-based algorithm is close to that of the BP algorithm and achieves a significant gain over the UMP APP-based decoding algorithm of Section 10.10.1; however, the number of required iterations is quite large, and little improvement is observed by increasing the number of iterations from 50 to 200 for BP-based decoding algorithms.

Construction of LDPC codes and various algorithms for decoding LDPC codes will be discussed in Chapter 17.

PROBLEMS

10.1 Prove the sufficient condition for optimality of a codeword given by (10.47).

10.2 Consider the value $G(\mathbf{v}_1, w_1; \mathbf{v}_2, w_1)$ given in (10.46).

a. Discuss $G(\mathbf{v}_1, w_1; \mathbf{v}_2, w_1)$ in the case where the codeword delivered by an algebraic decoder is known. What is the problem in trying to use this result for all received sequences?

b. Discuss $G(\mathbf{v}_1, w_1; \mathbf{v}_2, w_1)$ for $\mathbf{v}_1 = \mathbf{v}_2$.

10.3 GMD decoding considers only $\lfloor (d_{min} + 1)/2 \rfloor$ erasures in the $d_{min} - 1$ LRPs. Explain why not all $d_{min} - 1$ possible erasures are considered.

10.4 Consider an (n, k) binary linear code with even minimum distance d_{min}. Show that it is possible to achieve the same error performance as for the conventional Chase algorithm-2 by erasing one given position among the $\lfloor d_{min}/2 \rfloor$ least reliable positions (LRPs) of the received sequence and adding to the hard-decision decoding of the received sequence $\mathbf{r}$ all possible combinations of 0's and 1's in the remaining $\lfloor d_{min}/2 \rfloor - 1$ LRPs.

10.5 Consider an error-and-erasure algebraic decoder that successfully decodes any input sequence with t errors and s erasures satisfying $s + 2t < d_{min}$ and fails to decode otherwise. Define $S_e(a)$ as the set of candidate codewords generated by the algorithm $A_e(a)$ presented in Section 10.4. For $a = 1, \cdots, \lceil d_{min}/2 \rceil - 1$, show that $S_e(a) \subseteq S_e(a + 1)$.

10.6 In the KNIH algorithm presented in Section 10.6, show that any codeword $\mathbf{v}$ in $J(i)$ rather that the one that has the smallest correlation discrepancy with the received sequence $\mathbf{r}$ can be used for evaluating $G_i(\mathbf{v})$. Discuss the implications of this remark (advantages and drawbacks).

10.7 In the RLSD algorithm presented in Section 10.7, show that there exists at most one $(n - k)$-pattern that is not $(n - k - 1)$-eliminated.

10.8 For the RLSD algorithm presented in Section 10.7, determine the complete reduced list for the (15, 11, 3) Hamming code.

10.9 Determine the complete reduced list for the (8, 4, 4) extended Hamming code. Show that this complete list can be divided into two separate lists depending on whether the syndrome $\mathbf{s}$ is a column of the parity check matrix H. (*Hint:* Each list is composed of five distinct patterns).

10.10 In the RLSD algorithm presented in Section 10.7, prove that all $n(\mathbf{v})$-patterns with $n(\mathbf{v}) > n - k$ can be eliminated from all reduced lists. For $n(\mathbf{v}') < n(\mathbf{v})$, determine an $n(\mathbf{v}')$-pattern that justifies this elimination.

10.11 Let C and C_1 be the two codes defined in Section 10.8.1. Explain why if $\hat{\mathbf{v}}$ is the decoded codeword in C_1, then $\pi_1^{-1}\pi_2^{-1}[\hat{\mathbf{v}}]$ is simply the decoded codeword in C.

10.12 Prove that the most reliable basis and the least reliable basis are information sets of a code and its dual, respectively.

10.13 Prove that order-1 reprocessing achieves maximum likelihood decoding for the (8, 4, 4) RM code.

10.14 Which order of reprocessing achieves maximum likelihood decoding of an $(n, n - 1, 2)$ single parity-check code? Based on your answer, propose a much simpler method for achieving maximum likelihood decoding of single parity-check codes.

10.15 Describe the types of errors that can be corrected by Chase algorithm-2, but not by order-i reprocessing.

10.16 Assume that an rth-order RM code RM(r, m) is used for error control.

a. Show that all error patterns of weight at most t, as well as all error patterns of weight $t + 1$ with one error in a given position can be corrected.

b. Assuming reliability values are available at the decoder, propose a simple modification of majority-logic decoding (Reed algorithm) of RM(r, m) RM codes in which the error performance can be improved based on (a).

BIBLIOGRAPHY

1. J. G. Proakis. *Digital Communications, 3d ed.,* New York: McGraw-Hill, 1995.

2. M. P. C. Fossorier, S. Lin, and D. Rhee, "Bit Error Probability for Maximum Likelihood Decoding of Linear Block Codes and Related Soft Decision Decoding Methods," *IEEE Trans. Inform. Theory*, IT-44: 3083–90, November 1998.

3. A. J. Viterbi, "Error Bounds for Convolutional Codes and Asymptotically Optimum Decoding Algorithm," *IEEE Trans. Inform. Theory*, IT-13: 260–69, April 1967.

4. G. D. Forney, Jr., "The Viterbi Algorithm," *Proc. IEEE*, 61: 268–78, March 1973.

5. G. D. Forney, Jr., "Generalized Minimum Distance Decoding," *IEEE Trans. Inform. Theory*, IT-12: 125–31, April 1966.

6. G. Einarsson and C. E. Sundberg, "A Note on Soft Decision Decoding with Successive Erasures," *IEEE Trans. Inform. Theory*, IT-22: 88–96, January 1976.

7. D. J. Taipale and M. B. Pursley, "An Improvement to Generalized-Minimum-Distance Decoding," *IEEE Trans. Inform. Theory*, IT-37: 167–72, January 1991.

8. H. T. Moorthy, S. Lin, and T. Kasami, "Soft Decision Decoding of Binary Linear Block Codes Based on an Iterative Search Algorithm," *IEEE Trans. Inform. Theory*, IT-43: 1030–40, May 1997.

9. D. Chase, "A Class of Algorithms for Decoding Block Codes with Channel Measurement Information," *IEEE Trans. Inform. Theory*, IT-18: 170–82, January 1972.

10. M. P. C. Fossorier and S. Lin, "Chase-Type and GMD-Type Coset Decoding," *IEEE Trans. Commun.*, COM-48: 345–50, March 2000.

11. E. J. Weldon, Jr., "Decoding Binary Block Codes on Q-ary Output Channels," *IEEE Trans. Inform. Theory*, IT-17: 713–18, November 1971.

12. W. J. Chen, M. Fossorier, and S. Lin, "Quantization Issues for Soft-Decision Decoding of Linear Block Codes," *IEEE Trans. Commun.*, COM-47: 789–95, June 1999.

13. H. Tanaka and K. Kakigahara, "Simplified Correlation Decoding by Selecting Codewords Using Erasure Information," *IEEE Trans. Inform. Theory*, IT-29: 743–48, September 1983.

14. T. Kaneko, T. Nishijima, H. Inazumi, and S. Hirasawa, "An Efficient Maximum Likelihood Decoding of Linear Block Codes with Algebraic Decoder," *IEEE Trans. Inform. Theory*, IT-40: 320–27, March 1994.

15. T. Kaneko, T. Nishijima, and S. Hirasawa, "An Improvement of Soft-Decision Maximum-Likelihood Decoding Algorithm Using Hard-Decision Bounded-Distance Decoding," *IEEE Trans. Inform. Theory*, IT-43: 1314–19, July 1997.

16. J. Snyders, "Reduced Lists of Error Patterns for Maximum Likelihood Soft Decoding," *IEEE Trans. Inform. Theory*, IT-37: 1194–1200, July 1991.

17. J. Snyders and Y. Be'ery, "Maximum Likelihood Soft Decoding of Binary Block Codes and Decoders for the Golay Codes," *IEEE Trans. Inform. Theory*, IT-35: 963–75, September 1989.

18. N. J. C. Lous, P. A. H. Bours, and H. C. A. van Tilborg, "On Maximum Likelihood Soft-Decision Decoding of Binary Linear Codes," *IEEE Trans. Inform. Theory*, IT-39: 197–203, January 1993.

19. M. Fossorier, S. Lin, and J. Snyders, "Reliability-Based Syndrome Decoding of Linear Block Codes," *IEEE Trans. Inform. Theory*, IT-44: 388–98, January 1998.

20. Y. S. Han, C. R. P. Hartmann, and C. C. Chen, "Efficient Priority-First Search Maximum-Likelihood Soft-Decision Decoding of Linear Block Codes," *IEEE Trans. Inform. Theory*, IT-39: 1514–23, September 1993.

21. Y. S. Han, C. R. P. Hartmann, and K. G. Mehrotra, "Decoding Linear Block Codes Using a Priority-First Search: Performance Analysis and Suboptimal Version," *IEEE Trans. Inform. Theory*, IT-44: 1233–46, May 1998.

22. B. G. Dorsch, "A Decoding Algorithm for Binary Block Codes and J-ary Output Channels". *IEEE Trans. Inform. Theory*, IT-20: 391–94, May 1974.

23. G. Battail and J. Fang, "Décodage pondéré optimal des codes linéaires en blocs" *Annales des Télécommunications*, AT-41 (11–12): 580–604, November–December 1986.

24. A. Valembois and M. Fossorier, "An Improved Method to Compute Lists of Binary Vectors That Optimize a Given Weight Function with Application to Soft Decision Decoding," *IEEE Commun. Lett.*, CL-5: 456–58, November 2001.

25. A. Valembois and M. Fossorier, "A Comparison between "Most-Reliable-Basis Reprocessing" Strategies," *IEICE Trans. Fundamentals*, E85-A: 1727–41, July 2002.

26. M. Fossorier and S. Lin, "Soft-Decision Decoding of Linear Block Codes based on Ordered Statistics," *IEEE Trans. Inform. Theory*, IT-41: 1379–96, September 1995.

27. D. Agrawal and A. Vardy, "Generalized Minimum Distance Decoding in Euclidean Space: Performance Analysis," *IEEE Trans. Inform. Theory*, IT-46: 60–83, January 2000.

28. M. Fossorier and S. Lin, "Error Performance Analysis for Reliability-Based Decoding Algorithms," *IEEE Trans. Inform. Theory*, IT-48: 287–93, January 2002.

29. M. P. C. Fossorier, T. Koumoto, T. Takata, T. Kasami, and S. Lin, "The Least Stringent Sufficient Condition on the Optimality of a Suboptimally Decoded Codeword Using the Most Reliable Basis," *Proc. IEEE Int. Symp. Inform. Theory*, Ulm, Germany, p. 430, June 1997.

30. D. Gazelle and J. Snyders, "Reliability-Based Code-Search Algorithm for Maximum Likelihood Decoding of Block Codes," *IEEE Trans. Inform. Theory*, IT-43: 239–49, January 1997.

31. M. P. C. Fossorier and S. Lin, "Computationally Efficient Soft-Decision Decoding, of Linear Block Codes Based on Ordered Statistics," *IEEE Trans. Inform. Theory*, IT-42: 738–50, May 1996.

32. M. P. C. Fossorier and S. Lin, "Reliability-Based Information Set Decoding of Binary Linear Codes," *IEICE Trans. Fundamentals*, E82-A: 2034–42, October 1999.

33. M. Fossorier, "Reliability-Based Soft-Decision Decoding with Iterative Information Set Reduction," *IEEE Trans. Inform. Theory*, IT-48: 3101–06, December 2002.

34. A. Valembois and M. Fossorier, "Box and Match Techniques Applied to Soft Decision Decoding," to appear in *IEEE Trans. Inform. Theory*, 2004.

35. M. P. C. Fossorier and S. Lin, "Complementary Reliability-Based Decodings of Binary Linear Block Codes," *IEEE Trans. Inform. Theory*, IT-43: 1667–72, September 1997.

36. I. Dumer, "Suboptimal Decoding of Linear Codes: Partition Technique," *IEEE Trans. Inform. Theory*, IT-42: 1971–86, November 1996.

37. I. Dumer, "Ellipsoidal Coverings and Error-Free Search in Maximum Likelihood Decoding," *Proc. IEEE Int. Symp. Inform. Theory*, Boston, Mass., p. 367, August 1998.

38. J. L. Massey. *'Threshold Decoding*,' MIT Press, Cambridge, 1963.

39. V. D. Kolesnik, "Probability Decoding of Majority Codes," *Probl. Peredachi Inform.*, 7: 3–12, July 1971.

40. G. C. Clark, Jr., and J. B. Cain, *Error-Correcting Coding for Digital Communications*, Plenum, New York, 1982.

41. W. Meier and O. Staffelbach, "Fast Correlation Attacks on Certain Stream Ciphers" *J. Cryptology*, 1: 159–76, 1989.

42. R. Lucas, M. Bossert, and M. Breitbach, "On Iterative Soft-Decision Decoding of Linear Binary Block Codes and Product Codes," *IEEE J. Select. Areas Commun.*, JSAC-16: 276–98, February 1998.

43. K. Yamaguchi, H. Iizuka, E. Nomura, and H. Imai, "Variable Threshold Soft Decision Decoding," *IEICE Trans. Elect. Commun.*, 72: 65–74, September 1989.

44. J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Morgan Kaufmann, San Mateo, Calif., 1988.

45. R. G. Gallager, *Low-Density Parity-Check Codes*, MIT Press, Cambridge, 1963.

46. D. J. C. MacKay, "Good Error-Correcting Codes Based on Very Sparse Matrices," *IEEE Trans. Inform. Theory*, IT-45: 399–431, March 1999.

47. R. J. McEliece, D. J. C. MacKay, and J.-F. Cheng, "Turbo Decoding As an Instance of Pearl's "Belief Propagation" Algorithm," *IEEE J. Select. Areas Commun.*, JSAC-16: 140–52, February 1998.

48. F. R. Kschischang and B. J. Frey, "Iterative Decoding of Compound Codes by Probability Propagation in Graphical Models," *IEEE J. Select. Areas Commun.*, JSAC-16: 219–30, February 1998.

49. M. P. C. Fossorier, M. Mihaljevič, and H. Imai, "Reduced Complexity Iterative Decoding of Low Density Parity Check Codes Based on Belief Propagation," *IEEE Trans. Commun.*, COM-47: 673–80, May 1999.

50. R. Lucas, M. Fossorier, Y. Kou, and S. Lin, "Iterative Decoding of One-Step Majority-Logic Decodable Codes Based on Belief Propagation," *IEEE Trans. Commun.*, COM-48: 931–37, June 2000.

51. Y. Kou, S. Lin, and M. P. C. Fossorier, "Low Density Parity Check Codes Based on Finite Geometries: A Rediscovery," *IEEE Trans. Inform. Theory*, IT-47: 2711–36, November 2001.