TRELLISES AND TRELLIS-BASED DECODING ALGORITHMS FOR LINEAR BLOCK CODES

Part I

Shu Lin and Marc Fossorier

January 15, 1998

1 INTRODUCTION

1.1 TRELLIS REPRESENTATION OF CODES

A code trellis is a graphical representation of a code, block or convolutional, in which every path represents a codeword (or a code sequence for a convolutional code). This representation makes it possible to implement maximum likelihood decoding (MLD) of a code with reduced decoding complexity. The most well known trellis-based MLD algorithm is the Viterbi algorithm [23, 79, 105]. The trellis representation was first introduced and used for convolutional codes [23]. This representation, together with the Viterbi decoding algorithm, has resulted in a wide range of applications of convolutional codes for error control in digital communications over the last two decades.

The recent search for efficient MLD schemes for linear block codes has motivated some coding theorists to study the trellis structure of these codes so that trellis-based decoding algorithms can be devised to reduce decoding complexity. Trellis representation of linear block codes was first presented in [1] and then

in [67, 109]. The first serious study of trellis structure and trellis construction for linear block codes was due to Wolf. In his 1978 paper [109], Wolf presented the first method for constructing trellises for linear block codes and proved that an N-section trellis diagram for a q-ary (N, K) linear block code has at most $q^{\min(K, N-K)}$ states. He also presented a method for labeling the states based on the parity-check matrix of a code. Right after Wolf's work, Massey presented a simple but elegant paper [67] in which he gave a precise definition of a code trellis, derived some fundamental properties, and provided implications of the trellis structure for encoding and decoding of codes. However, these early works in trellis representation of linear block codes did not arouse much enthusiasm, and for the next 10 years, there was basically no research in this area.

There are two major reasons for this inactive period of research in this area. First, most coding theorists at that time believed that block codes did not have simple trellis structure like convolutional codes and maximum likelihood decoding of linear block codes using the Viterbi algorithm was practically impossible, except for very short block codes. Second, since almost all of the linear block codes are constructed algebraically or based on finite geometries, it was the belief of many coding theorists that algebraic decoding was the only way to decode these codes. These two reasons seriously hindered the development of efficient soft-decision decoding methods for linear block codes and their applications to error control in digital communications. This led to a general belief that block codes are inferior to convolutional codes and hence, that they were not useful.

In fact, for more than two decades, most of the practicing communication engineers believed that the rate-1/2 convolutional code of constraint length 7 with Viterbi decoding was the only effective error control coding scheme for digital communications, except for perhaps ARQ schemes. To achieve higher reliability for certain applications such as NASA's satellite and deep space communications, this convolutional code concatenated with a Reed-Solomon outer code was thought the best solution.

It was really Forney's paper in 1988 [24] that aroused enthusiasm for research in the trellis structure of linear block codes. In this paper, Forney showed that some block codes, such as Reed-Muller (RM) codes and some lattice codes, do have relatively simple trellis structures, and he presented a method for con-

structing sectionalized trellises for linear block codes and asserted that the construction results in minimal trellises with respect to the state complexity (the number of states). Motivated by Forney's work and the desire to achieve maximum likelihood decoding of linear block codes to improve error performance over traditional hard-decision algebraic decoding, there have been significant efforts in studying the trellis structure and devising trellis-based decoding algorithms for linear block codes over the last eight years. Developments have been dramatic and rapid, and the new results are exciting and encouraging. Trellis-based decoding algorithms that are more efficient than the conventional Viterbi decoding algorithm have recently been devised [32, 37] and implementation of trellis-based high-speed decoders for NASA's high-speed satellite communications is now underway [52, 63]. All of these new developments make block codes more competitive with convolutional codes.

1.2 ORGANIZATION OF THE BOOK

Chapter 2 gives a brief review of linear block codes. The goal is to provide the essential background material for the development of trellis structure and trellis-based decoding algorithms for linear block codes in the later chapters. Chapters 3 through 6 present the fundamental concepts, finite-state machine model, state space formulation, basic structural properties, state labeling, construction procedures, complexity, minimality, and sectionalization of trellises. Chapter 7 discusses trellis decomposition and subtrellises for low-weight codewords. Chapter 8 first presents well known methods for constructing long powerful codes from short component codes or component codes of smaller dimensions, and then provides methods for constructing their trellises which include Shannon and Cartesian product techniques. Chapter 9 deals with convolutional codes, puncturing, zero-tail termination and tail-biting. It shows that trellis construction procedures for both block and convolutional codes are essentially the same, except that the trellises for convolutional codes or terminated convolutional codes are time-invariant and the trellises for block codes are in general time-varying. For both types of codes, trellis states are defined based on a certain set of information bits, called the state-defining information set.

Chapters 10 through 13 present various trellis-based decoding algorithms, old and new. Chapter 10 first discusses the application of the well known Viterbi decoding algorithm to linear block codes, optimum sectionalization of a code trellis to minimize computation complexity, and design issues for IC (integrated circuit) implementation of a Viterbi decoder. Then it presents a new decoding algorithm for convolutional codes, named differential trellis decoding (DTD) algorithm. DTD algorithm is devised based on the principle of **compare-select-add** (CSA) which is simply the opposite of the principle of **add-compare-select** (ACS) used in the Viterbi algorithm. This new algorithm is more efficient than the Viterbi decoding algorithm. For rate-1/2 antipodal convolutional codes and their higher rate punctured codes, it requires about 1/3 less real number operations than the Viterbi decoding algorithm. This DTD algorithm can also be applied to trellis decoding of block codes. Chapter 11 presents a trellis-based recursive MLD for linear block codes, the RMLD algorithm. This decoding algorithm is devised based on the **divide and conquer principle**. The implementation of this algorithm does not require the construction of the entire code trellis; only some special one-section trellises of much smaller state and branch complexities for constructing path metric tables recursively are needed. This reduces the decoding complexity significantly and it is more efficient than the Viterbi decoding algorithm. Furthermore, it allows **parallel/pipeline** processing of received sequences to speed up decoding. Chapter 12 presents a suboptimum reliability-based iterative decoding algorithm with a low-weight trellis search for the most likely codeword. This decoding algorithm provides a good trade-off between error performance and decoding complexity. All the decoding algorithms presented in Chapters 10 through 12 are devised to minimize word error probability. Chapter 13 presents decoding algorithms that minimize bit error probability and provide the corresponding soft (reliability) information at the output of the decoder. Decoding algorithms presented are the MAP (**maximum a posteriori probability**) decoding algorithm and the SOVA (**soft-output Viterbi algorithm**) algorithm. Finally, the minimization of bit error probability in trellis-based MLD is discussed.

2 LINEAR BLOCK CODES

Chapter 2 gives a brief review of linear block codes. The goal is to provide the essential background material for the development of trellis structure and trellis-based decoding algorithms for linear block codes in the later chapters. We mainly present the basic concepts of encoding and decoding of linear block codes and state some facts without derivations or proofs. Since in most present digital data communication systems, information is coded in binary digits, '0' or '1', we discuss only linear block codes with symbols from the binary field GF(2). First, linear block codes are defined and described in terms of generator and parity-check matrices. Second, coset partition of a linear block code is discussed, which is needed in analyzing the code trellis structure and construction. Third, the concepts of minimum distance, weight distribution and distance profile are presented, which are needed in the later chapters for presenting decoding algorithms and their error performances. Finally, the concepts of hard-decision, soft-decision, and maximum likelihood decoding are presented.

References 3, 9, 14, 59, 63, 78 and 79 contain excellent treatments of linear block codes.

2.1 GENERATION OF LINEAR BLOCK CODES

In block coding, an information sequence of binary digits (called bits) is divided into message blocks of fixed length; each message block consists of K information bits. There are a total of 2^K distinct messages. Each message is encoded into a codeword (or code sequence) of N bits according to certain rules, where $N \geq K$. Therefore, corresponding to the 2^K possible messages, there are 2^K codewords. This set of 2^K codewords forms a block code of length N. For a block code to be useful, the 2^K codewords must be distinct. Hence, there should be a one-to-one correspondence between a message and a codeword.

Definition 2.1 A binary block code of length N and 2^K codewords is called an (N, K) linear block code if and only if its 2^K codewords form a K-dimensional subspace of the vector space of all the N-tuples over the binary field GF(2). The parameter K is called the **dimension** of the code space.

An (N, K) linear block code C is generated by a $K \times N$ generator matrix over GF(2),

$$G = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_K \end{bmatrix} = \begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1i} & \cdots & g_{1N} \\ g_{21} & g_{22} & \cdots & g_{2i} & \cdots & g_{2N} \\ \vdots & \vdots & & \vdots & & \vdots \\ g_{K1} & g_{K2} & \cdots & g_{Ki} & \cdots & g_{KN} \end{bmatrix} \tag{2.1}$$

where the rows, $g_1, g_2, \ldots, g_K$, are linearly independent over GF(2). The 2^K linear combinations of the K rows of G form the codewords of C. We say that the rows of G span the code C, or C is the **row space** of G. Let

$$\boldsymbol{a} = (a_1, a_2, \ldots, a_K)$$

be a message to be encoded. A natural encoding mapping is that the codeword

$$\boldsymbol{u} = (u_1, u_2, \ldots, u_N)$$

for the message $\boldsymbol{a} = (a_1, a_2, \ldots, a_K)$ is given by

$$\boldsymbol{u} = \boldsymbol{a} \cdot G$$

$$= (a_1, a_2, \ldots, a_K) \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_K \end{bmatrix}$$

$$= a_1 \cdot g_1 + a_2 \cdot g_2 + \cdots + a_K \cdot g_K. \tag{2.2}$$

From (2.1) and (2.2), we find that for $1 \leq i \leq N$, the i-th component of u is given by

$$u_i = a_1 \cdot g_{1i} + a_2 \cdot g_{2i} + \cdots + a_K \cdot g_{Ki}. \tag{2.3}$$

During an encoding interval, K information bits are encoded into N code bits. These N code bits are shifted onto the channel, one at a time, in N units of time. An encoding interval, denoted Γ, is represented by a set of $N+1$ time instants,

$$\Gamma = \{0, 1, 2, \ldots, N\}. \tag{2.4}$$

For $1 \leq i \leq N$, the i-th unit of time is the interval from time-$(i-1)$ to time-i. During this interval, the i-th code bit u_i is formed and transmitted. By time-i, the transmission is completed. This interval is called a **bit interval**.

Example 2.1 Consider a binary $(8, 4)$ linear block code which is generated by the following generator matrix:

$$G = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}. \tag{2.5}$$

If $a = (1101)$ is the message to be encoded, its corresponding codeword, according to (2.2), is given by

$$\begin{aligned} u &= 1 \cdot g_1 + 1 \cdot g_2 + 0 \cdot g_3 + 1 \cdot g_4 \\ &= (11111111) + (00001111) + (01010101) \\ &= (10100101). \end{aligned}$$

The 16 codewords of this code are listed in Table 2.1.

△△

Table 2.1. The codewords of the code generated by (2.5).

Messages	Codewords	Messages	Codewords
(0000)	(00000000)	(0001)	(01010101)
(1000)	(11111111)	(1001)	(10101010)
(0100)	(00001111)	(0101)	(01011010)
(1100)	(11110000)	(1101)	(10100101)
(0010)	(00110011)	(0011)	(01100110)
(1010)	(11001100)	(1011)	(10011001)
(0110)	(00111100)	(0111)	(01101001)
(1110)	(11000011)	(1111)	(10010110)

A binary (N, K) linear block code C is also uniquely specified by an $(N - K) \times N$ matrix over GF(2), called a **parity-check matrix**,

$$H = \begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1N} \\ h_{21} & h_{22} & \cdots & h_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ h_{N-K,1} & h_{N-K,2} & \cdots & h_{N-K,N} \end{bmatrix}, \tag{2.6}$$

where the rows are linearly independent. A binary N-tuple $\boldsymbol{u} = (u_1, u_2, \ldots, u_N)$ is a codeword in C if and only if the following condition holds:

$$\boldsymbol{u} \cdot H^T = \mathbf{0}, \tag{2.7}$$

where $\mathbf{0}$ denotes the all-zero $(N - K)$-tuple, $(0, 0, \ldots, 0)$. Code C is called the **dual** (or **null**) space of H. H itself generates an $(N, N - K)$ linear code, denote $C^{\perp}$. For any codeword $\boldsymbol{u} = (u_1, u_2, \ldots, u_N) \in C$ and any codeword $\boldsymbol{v} = (v_1, v_2, \ldots, v_N) \in C^{\perp}$, the **inner product**

$$\begin{aligned} \boldsymbol{u} \cdot \boldsymbol{v} &\triangleq u_1 \cdot v_1 + u_2 \cdot v_2 + \cdots + u_N \cdot v_N \\ &= 0. \end{aligned}$$

$C^{\perp}$ is called the **dual code** of C and vise versa.

In general, the generator matrix G of a linear block code C is used for encoding, while the parity-check matrix H is used for decoding, particularly for error detection.

Example 2.2 Consider the (8,4) linear block code given in Example 2.1. A parity-check matrix for this code is the generator matrix itself given by (2.5), i.e.,

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

In this case, $C = C^{\perp}$ and C is said to be **self-dual**.

$\triangle\triangle$

For an (N, K) linear block code C, the ratio $R = K/N$ is called the **code rate** which represents the average number of information bits carried by a code symbol (or the average number of information bits transmitted per channel usage).

2.2 COSET PARTITION OF A LINEAR BLOCK CODE

Consider a binary (N, K) linear block code C with a generator matrix G. Let K_1 be a nonnegative integer such that $0 \leq K_1 \leq K$. A subset of 2^{K_1} codewords in C is said to be a **linear subcode** of C if this subset itself is a K_1-dimensional subspace of the vector space of all the N-tuples over GF(2). Any K_1 rows of the generator matrix G span an (N, K_1) linear subcode of C, and they form a generator matrix for the subcode. If $K_1 = 0$, the subcode consists of only the all-zero codeword $\mathbf{0}$ of C. For $K_1 = K$, the subcode is just the code itself.

Let C_1 be an (N, K_1) linear subcode of C. Then C can be partitioned into 2^{K-K_1} disjoint **cosets** of C_1; each coset is of the following form:

$$\boldsymbol{v}_l \oplus C_1 \triangleq \{\boldsymbol{v}_l + \boldsymbol{u} : \boldsymbol{u} \in C_1\} \tag{2.8}$$

with $1 \leq l \leq 2^{K-K_1}$, where for $\boldsymbol{v}_l \neq \mathbf{0}$, $\boldsymbol{v}_l$ is in C but not in C_1 and for $\boldsymbol{v}_l = \mathbf{0}$, the coset $\mathbf{0} \oplus C_1$ is just the subcode C_1 itself. This partition of C with respect to C_1 is denoted with C/C_1, and the codewords $\boldsymbol{v}_l$ for $1 \leq l \leq 2^{K-K_1}$ are called

the coset representatives. Any codeword in a coset can be used as the coset representative without changing the composition (the codewords) of the coset. Important properties of cosets are:

(1) The sum of two codewords in a coset is a codeword in the subcode C_1.

(2) Let x and y be two codewords in cosets $v_i \oplus C_1$ and $v_j \oplus C_1$, respectively, where $i \neq j$. Then the sum $x+y$ is a codeword in the coset $(v_i+v_j)\oplus C_1$ with $v_i + v_j$ as the coset representative.

The set of representatives for the cosets in the partition C/C_1 is denoted $[C/C_1]$ which is called the coset representative space for the partition C/C_1. Code C can be expressed as the direct-sum of C_1 and $[C/C_1]$ as follows:

$$C = [C/C_1] \oplus C_1 \triangleq \{v + u : v \in [C/C_1] \text{ and } u \in C_1\}. \tag{2.9}$$

Let G_1 be the subset of K_1 rows of the generator matrix G which generates the subcode C_1. Then the 2^{K-K_1} codewords generated by the $K - K_1$ rows in the set $G \setminus G_1$ can be used as the representatives for the cosets in the partition C/C_1. These 2^{K-K_1} codewords form an $(N, K - K_1)$ linear subcode of C.

Let C_2 be an (N, K_2) linear subcode of C_1 with $0 \leq K_2 \leq K_1$. We can further partition each coset $v_l \oplus C_1$ in the partition C/C_1 based on C_2 into $2^{K_1-K_2}$ cosets of C_2; each coset consists of the following codewords in C:

$$v_l \oplus (w_k \oplus C_2) \triangleq \{v_l + w_k + u : u \in C_2\} \tag{2.10}$$

with $1 \leq l \leq 2^{K-K_1}$ and $1 \leq k \leq 2^{K_1-K_2}$ where for $w_k \neq 0$, w_k is a codeword in C_1 but not in C_2. We denote this partition with $C/C_1/C_2$. This partition consists of 2^{K-K_2} cosets of C_2. Now C can be expressed as the following direct-sum:

$$C = [C/C_1] \oplus [C_1/C_2] \oplus C_2. \tag{2.11}$$

Let $C_1, C_2, \ldots, C_m$ be a sequence of linear subcodes of C with dimensions $K_1, K_2, \ldots, K_m$, respectively, such that

$$C \supseteq C_1 \supseteq C_2 \supseteq \cdots \supseteq C_m \tag{2.12}$$

and

$$K \geq K_1 \geq K_2 \geq \cdots \geq K_m \geq 0. \tag{2.13}$$

Then we can form a chain of partitions,

$$C/C_1, C/C_1/C_2, \ldots, C/C_1/C_2/\cdots/C_m, \tag{2.14}$$

and C can be expressed as the following direct-sum:

$$C = [C/C_1] \oplus [C_1/C_2] \oplus \cdots \oplus [C_{m-1}/C_m] \oplus C_m. \tag{2.15}$$

2.3 THE MINIMUM DISTANCE AND WEIGHT DISTRIBUTION OF A LINEAR BLOCK CODE

Let $\boldsymbol{u}$ and $\boldsymbol{v}$ be two N-tuples over GF(2). The **Hamming distance** between $\boldsymbol{u}$ and $\boldsymbol{v}$, denoted $d(\boldsymbol{u}, \boldsymbol{v})$, is defined as the number of places where they differ. The **minimum (Hamming) distance** of a block code C, denoted $d_{\min}(C)$, is defined as the minimum Hamming distance between all distinct pairs of codewords in C, i.e.,

$$d_{\min}(C) \triangleq \min\{d(\boldsymbol{u}, \boldsymbol{v}) : \boldsymbol{u}, \boldsymbol{v} \in C, \boldsymbol{u} \neq \boldsymbol{v}\}. \tag{2.16}$$

The (Hamming) **weight** of an N-tuple $\boldsymbol{v}$, denoted $w(\boldsymbol{v})$, is defined as the number of nonzero components of $\boldsymbol{v}$. It follows from the definition of Hamming distance and the fact that the sum of two N-tuples over GF(2) is an another N-tuple over GF(2) that the Hamming distance between two N-tuples, $\boldsymbol{u}$ and $\boldsymbol{v}$, is equal to the Hamming weight of the sum of $\boldsymbol{u}$ and $\boldsymbol{v}$, i.e.,

$$d(\boldsymbol{u}, \boldsymbol{v}) = w(\boldsymbol{u} + \boldsymbol{v}). \tag{2.17}$$

For a linear block code C, it follows from (2.16) and (2.17) that

$$\begin{aligned} d_{\min}(C) &= \min\{w(\boldsymbol{u} + \boldsymbol{v}) : \boldsymbol{u}, \boldsymbol{v} \in C, \boldsymbol{u} \neq \boldsymbol{v}\} \\ &= \min\{w(\boldsymbol{x}) : \boldsymbol{x} \in C, \boldsymbol{x} \neq \boldsymbol{0}\} \\ &\triangleq w_{\min}(C). \end{aligned} \tag{2.18}$$

The parameter $w_{\min}(C) \triangleq \min\{w(\boldsymbol{x}) : \boldsymbol{x} \in C, \boldsymbol{x} \neq \boldsymbol{0}\}$ is called the **minimum weight** of C. Eq.(2.18) simply says that the minimum distance of a linear block code is equal to the minimum weight of its nonzero codewords. The minimum

weight of the $(8,4)$ linear block code given in Table 2.1 is 4; therefore, its minimum distance is 4.

Let C be an (N,K) linear block code. For $0 \leq i \leq N$, let A_i be the number of codewords with weight i. The numbers $A_0, A_1, A_2, \ldots, A_N$ are called the **weight distribution** of C. It is clear that $A_0 = 1$. The weight distribution of the $(8,4)$ linear block code given in Table 2.1 is

$$A_0 = 1,\ A_1 = A_2 = A_3 = 0,\ A_4 = 14,\ A_5 = A_6 = A_7 = 0,\ A_8 = 1.$$

The weight distribution is often expressed as a polynomial,

$$A(X) = A_0 + A_1 X + \cdots + A_N X^N,$$

which is called the **weight enumerator** of C. Let $W = \{0, w_1, w_2, \ldots, w_m\}$ denote the set of all weights of codewords in C such that:

(i) $0 < w_1 < w_2 < \cdots < w_m \leq N$; and

(ii) For $1 \leq i \leq m$, the number of codewords in C with weight w_i is not equal to zero.

This set is called the **weight profile** of C. The weight profile of the $(8,4)$ linear block code given by Table 2.1 is $\{0,4,8\}$. Let $\boldsymbol{u}$ be any codeword in C. The weight distribution of C actually gives the distribution of distances of codewords in C from the codeword $\boldsymbol{u}$. The weight profile W of C gives the profile of distances of codewords in C from the codeword $\boldsymbol{u}$.

The error performance of a linear block code is determined by its minimum distance and weight distribution. For an (N,K) linear block code with minimum distance $d_{\min}$, we often use the notation $(N,K,d_{\min})$ to represent the code. Therefore, the code given by Table 2.1 is an $(8,4,4)$ linear block code.

2.4 DECODING

Suppose an (N,K) linear block code C is used for error control over an additive white Gaussian noise (AWGN) channel. Let $\boldsymbol{u} = (u_1, u_2, \ldots, u_N)$ be the codeword to be transmitted. Before the transmission, a modulator maps each code bit into an elementary signal waveform. Binary PSK or FSK are commonly used signal waveforms for transmitting the bits in a codeword. The

resultant signal sequence is then transmitted over the channel and corrupted by noise. At the receiving end, the received signal sequence is processed by a demodulator and sampled at the end of each signal (bit) interval. This results in sequence of real numbers,

$$r = (r_1, r_2, \ldots, r_N),$$

which is called the **received sequence**. For $1 \leq i \leq N$, the i-th received component r_i is the sum of a fixed real number c_i and a Gaussian random variable n_i of zero-mean and variance $N_0/2$ where c_i corresponds to the transmitted code bit u_i at time-i. These received components may or may not be quantized. At one extreme, the demodulator can be used to make **firm decisions** on whether each transmitted code bit is a '0' or a '1'. Thus the output is quantized to two levels, denoted as 0 and 1. We say that the demodulator has made a **"hard decision"** on each transmitted code bit. This hard decision results in a binary received sequence,

$$z = (z_1, z_2, \ldots, z_N),$$

which may contain **transmission errors**, i.e., for some i, $z_i \neq u_i$. This binary hard-decision sequence is fed into a decoder which attempts to correct the transmission errors (if any) and recover the transmitted codeword u. Since the decoder operates on the hard decisions made by the demodulator, the decoding process is called **hard-decision decoding**. At the other extreme, the unquantized outputs from the demodulator can be fed directly into the decoder for processing. We refer to the resulting decoding as **soft-decision decoding**. Since the decoder makes use of the additional information contained in the unquantized received samples to recover the transmitted codeword, soft-decision decoding provides better error performance than hard-decision decoding. Decoding based on the quantized outputs from the demodulator, where the number of quantization levels exceeds two, is also referred to as soft-decision decoding. Soft-decision decoding provides better error performance than hard-decision decoding; however, hard-decision decoding is much simpler to implement. Various hard-decision decoding algorithms based on the algebraic structures of linear block codes have been devised. These hard-decision decoding algorithms are also termed algebraic decoding algorithms. Recently, effective soft-decision

decoding algorithms have been devised, and they achieve either optimum error performance or suboptimum error performance with reduced decoding complexity.

Let $\boldsymbol{u}^*$ be the estimate of the transmitted codeword at the output of the decoder. If the codeword $\boldsymbol{u}$ was transmitted, a decoding error occurs if and only if $\boldsymbol{u}^* \neq \boldsymbol{u}$. Given that $\boldsymbol{r}$ is received, the conditional error probability of the decoder is defined as

$$P(E \mid \boldsymbol{r}) \triangleq P(\boldsymbol{u}^* \neq \boldsymbol{u} \mid \boldsymbol{r}). \tag{2.19}$$

The error probability of the decoder is then given by

$$P(E) = \sum_{\boldsymbol{r}} P(E \mid \boldsymbol{r}) P(\boldsymbol{r}). \tag{2.20}$$

$P(\boldsymbol{r})$ is independent of the decoding rule used since $\boldsymbol{r}$ is produced prior to decoding. Hence, an optimum decoding rule (i.e., one that minimizes $P(E)$) must minimize $P(E \mid \boldsymbol{r}) = P(\boldsymbol{u}^* \neq \boldsymbol{u} \mid \boldsymbol{r})$ for all $\boldsymbol{r}$. Since minimizing $P(\boldsymbol{u}^* \neq \boldsymbol{u} \mid \boldsymbol{r})$ is equivalent to maximizing $P(\boldsymbol{u}^* = \boldsymbol{u} \mid \boldsymbol{r})$, $P(E \mid \boldsymbol{r})$ is minimized for a given $\boldsymbol{r}$ by choosing $\boldsymbol{u}^*$ as the codeword that maximizes

$$P(\boldsymbol{u} \mid \boldsymbol{r}) = \frac{P(\boldsymbol{r} \mid \boldsymbol{u}) P(\boldsymbol{u})}{P(\boldsymbol{r})}, \tag{2.21}$$

that is, $\boldsymbol{u}^*$ is chosen as the **most likely** codeword given $\boldsymbol{r}$ is received. If all the codewords are equally likely, maximizing (2.21) is equivalent to maximizing $P(\boldsymbol{r} \mid \boldsymbol{u})$. For an AWGN channel,

$$P(\boldsymbol{r} \mid \boldsymbol{u}) = \prod_{i=1}^{N} P(r_i \mid u_i), \tag{2.22}$$

since each received symbol depends on the corresponding transmitted symbol. A decoder that chooses its estimate to maximize (2.22) is called a **maximum likelihood decoder** and the decoding process is called the **maximum likelihood decoding (MLD)**. Maximizing (2.22) is equivalent to maximizing

$$\log P(\boldsymbol{r} \mid \boldsymbol{u}) = \sum_{i=1}^{N} \log P(r_i \mid u_i) \tag{2.23}$$

which is called the **log-likelihood function**.

Suppose BPSK signaling is used. Assume that each signal has unit energy. Let $\boldsymbol{u} = (u_1, u_2, \ldots, u_N)$ be the codeword to be transmitted. The modulator maps this codeword into a bipolar sequence represented by

$$\boldsymbol{c} = (c_1, c_2, \ldots, c_N)$$

where for $1 \leq i \leq N$,

$$c_i = 2u_i - 1. \tag{2.24}$$

From (2.24), we see that

$$c_i = \begin{cases} -1, & \text{if } u_i = 0, \\ +1, & \text{if } u_i = 1. \end{cases} \tag{2.25}$$

The **squared Euclidean distance** between the received sequence $\boldsymbol{r} = (r_1, r_2, \ldots, r_N)$ and $\boldsymbol{c}$ is given by

$$|\boldsymbol{r} - \boldsymbol{c}|^2 \triangleq \sum_{i=1}^{N} (r_i - c_i)^2. \tag{2.26}$$

For an AWGN channel, maximizing the log-likelihood function is equivalent to minimizing the squared Euclidean distance between $\boldsymbol{r}$ and $\boldsymbol{c}$. If we expand the right-hand side of (2.26), we have

$$|\boldsymbol{r} - \boldsymbol{c}|^2 = \sum_{i=1}^{N} r_i^2 - 2\sum_{i=1}^{N} r_i \cdot c_i + \sum_{i=1}^{N} c_i^2. \tag{2.27}$$

In computing $|\boldsymbol{r} - \boldsymbol{c}|^2$ for all codewords in C, $\sum_{i=1}^{N} r_i^2$ is a common term and $\sum_{i=1}^{N} c_i^2 = N$. Therefore, minimizing $|\boldsymbol{r} - \boldsymbol{c}|^2$ of (2.26) is equivalent to maximizing

$$\begin{aligned} \boldsymbol{r} \cdot \boldsymbol{c} &\triangleq \sum_{i=1}^{N} r_i \cdot c_i \\ &= \sum_{i=1}^{N} r_i \cdot (2u_i - 1). \end{aligned} \tag{2.28}$$

The inner product given by (2.28) is called the **correlation** between the received sequence $\boldsymbol{r}$ and the codeword $\boldsymbol{u}$.

Furthermore, (2.28) can be expanded as follows:

$$\boldsymbol{r} \cdot \boldsymbol{c} = 2 \sum_{i=1}^{N} r_i \cdot u_i - \sum_{i=1}^{N} r_i. \tag{2.29}$$

Since the second term $\sum_{i=1}^{N} r_i$ in (2.29) is a common term in computing $\boldsymbol{r} \cdot \boldsymbol{c}$ for all codewords in C, maximizing $\boldsymbol{r} \cdot \boldsymbol{c}$ is equivalent to maximizing

$$\boldsymbol{r} \cdot \boldsymbol{u} \triangleq \sum_{i=1}^{N} r_i \cdot u_i. \tag{2.30}$$

The inner product given by (2.30) is called the **binary correlation** between the received sequence $\boldsymbol{r}$ and the codeword $\boldsymbol{u}$.

Summarizing the above, MLD can be stated in four equivalent ways:

(1) **Log-likelihood function**
Decode the received sequence $\boldsymbol{r}$ into a codeword $\boldsymbol{u}$ for which the log-likelihood function

$$\log P(\boldsymbol{r} \mid \boldsymbol{u}) = \sum_{i=1}^{N} \log P(r_i \mid u_i)$$

is maximized.

(2) **Squared Euclidean distance**
Decode the received sequence $\boldsymbol{r}$ into a codeword $\boldsymbol{u}$ for which the squared Euclidean distance

$$|\boldsymbol{r} - \boldsymbol{u}|^2 \triangleq \sum_{i=1}^{N} (r_i - (2u_i - 1))^2$$

is minimized.

(3) **Correlation function**
Decode the received sequence $\boldsymbol{r}$ into a codeword $\boldsymbol{u}$ for which the correlation function

$$m(\boldsymbol{r}, \boldsymbol{u}) \triangleq \sum_{i=1}^{N} r_i \cdot (2u_i - 1)$$

is maximized.

(4) **Binary correlation function**

Decode the received sequence $\boldsymbol{r}$ into a codeword $\boldsymbol{u}$ for which the binary correlation function

$$b(\boldsymbol{r}, \boldsymbol{u}) = \sum_{i=1}^{N} r_i \cdot u_i$$

is maximized.

2.5 REED-MULLER CODES

Reed-Muller (RM) codes form a class of multiple error-correction codes. These codes were discovered by Muller in 1954 [78], but the first decoding algorithm for these codes was devised by Reed, also in 1954 [83]. They are finite geometry codes and rich in algebraic and geometric structures. The purpose of including these codes in this reviewing chapter is that they have very simple and regular trellis structures and their trellises can be easily constructed. These codes can be decoded effectively with trellis-based decoding algorithms. Furthermore, they provide good example codes. Throughout this book, many example codes are RM codes.

For any nonnegative integers m and r with $0 \leq r \leq m$, there exists a binary r-th order RM code, denoted $\mathrm{RM}_{r,m}$, with the following parameters:

Length	$N_{r,m} = 2^m$
Dimension	$K_{r,m} = 1 + \binom{m}{1} + \cdots + \binom{m}{r}$
Minimum distance	$d_{r,m} = 2^{m-r}$.

In the following, we first present the original construction of RM codes and then we describe an alternate construction for these codes. For $1 \leq i \leq m$, let $\boldsymbol{v}_i$ be a 2^m-tuple over GF(2) of the following form:

$$\boldsymbol{v}_i = (\underbrace{0\cdots0}_{2^{i-1}}, \underbrace{1\cdots1}_{2^{i-1}}, \underbrace{0\cdots0}_{2^{i-1}}, \ldots, \underbrace{1\cdots1}_{2^{i-1}}) \tag{2.31}$$

which consists of 2^{m-i+1} alternate all-zero and all-one 2^{i-1}-tuples. For $m = 3$, we have the following three 8-tuples:

$$\begin{aligned} \boldsymbol{v}_3 &= (00001111), \\ \boldsymbol{v}_2 &= (00110011), \\ \boldsymbol{v}_1 &= (01010101). \end{aligned}$$

Let $\boldsymbol{a} = (a_1, a_2, \ldots, a_N)$ and $\boldsymbol{b} = (b_1, b_2, \ldots, b_N)$ be two binary N-tuples. Define the following logic (**boolean**) product of $\boldsymbol{a}$ and $\boldsymbol{b}$,

$$\boldsymbol{a} \cdot \boldsymbol{b} \triangleq (a_1 \cdot b_1, a_2 \cdot b_2, \ldots, a_N \cdot b_N),$$

where '$\cdot$' denotes the logic product (or **AND** operation), i.e. $a_i \cdot b_i = 1$ if and only if both a_i and b_i are '1'. For $m = 3$,

$$\boldsymbol{v}_3 \cdot \boldsymbol{v}_1 = (00000101).$$

For simplicity, we use $\boldsymbol{ab}$ for $\boldsymbol{a} \cdot \boldsymbol{b}$.

Let $\mathbf{1}$ denote the all-one 2^m-tuple, $\mathbf{1} = (1, 1, \ldots, 1)$. For $1 \leq i_1 < i_2 < \cdots < i_l \leq m$, the product

$$\boldsymbol{v}_{i_1} \boldsymbol{v}_{i_2} \cdots \boldsymbol{v}_{i_l}$$

is said to have degree l. Since the weights of $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_m$ are even and powers of 2, it can be shown that the weight of the product $\boldsymbol{v}_{i_1} \boldsymbol{v}_{i_2} \cdots \boldsymbol{v}_{i_l}$ is also even and a power of 2, in fact 2^{m-l}.

The r-th order RM code, $\mathrm{RM}_{r,m}$, of length 2^m is generated by the following set of vectors:

$$\begin{aligned} G_{\mathrm{RM}}(r,m) \;=\; \{&\mathbf{1}, \boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_m, \boldsymbol{v}_1\boldsymbol{v}_2, \boldsymbol{v}_1\boldsymbol{v}_3, \ldots, \boldsymbol{v}_{m-1}\boldsymbol{v}_m, \\ &\ldots \text{up to products of degree } r\}. \end{aligned} \tag{2.32}$$

There are

$$K_{r,m} = 1 + \binom{m}{1} + \binom{m}{2} + \cdots + \binom{m}{r}$$

vectors in $G_{\mathrm{RM}}(r,m)$ and they are linearly independent. If the vectors in $G_{\mathrm{RM}}(r,m)$ are arranged as rows of a matrix, then the matrix is a generator matrix of the RM code, $\mathrm{RM}_{r,m}$. For $0 \leq l \leq r$, there are exactly $\binom{m}{l}$ rows in $G_{\mathrm{RM}}(r,m)$ of weight 2^{m-l}. All the codewords of the RM code, $\mathrm{RM}_{r,m}$ with $0 \leq r < m$, have even weights. It is also clear that the $(r-1)$-th order RM code, $\mathrm{RM}_{r-1,m}$, is a proper subcode of the r-th order RM code, $\mathrm{RM}_{r,m}$.

Example 2.3 Let $m = 4$. The 2nd order RM code of length 16 is generated by the following 11 vectors:

$$\begin{array}{ll} v_0 = \mathbf{1} & 1111111111111111 \\ v_4 & 0000000011111111 \\ v_3 & 0000111100001111 \\ v_2 & 0011001100110011 \\ v_1 & 0101010101010101 \\ v_3v_4 & 0000000000001111 \\ v_2v_4 & 0000000000110011 \\ v_1v_4 & 0000000001010101 \\ v_2v_3 & 0000001100000011 \\ v_1v_3 & 0000010100000101 \\ v_1v_2 & 0001000100010001 \end{array}$$

This is a $(16, 11)$ code with minimum distance 4.

$\triangle\triangle$

The code given in Example 2.1 is the 1st order RM code, $\mathrm{RM}_{1,3}$, of length 8. Let

$$G_{(2,2)} \triangleq \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \tag{2.33}$$

be a 2×2 matrix over GF(2). The **two-fold Kronecker product** of $G_{(2,2)}$ is defined as

$$\begin{aligned} G_{(2^2,2^2)} &\triangleq \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{2.34}$$

where $\otimes$ denotes the **Kronecker product**. The 3-fold Kronecker product of $G_{(2,2)}$ is defined as

$$G_{(2^3,2^3)} \triangleq \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.35}$$

Similarly, we can define the m-fold Kronecker product of $G_{(2,2)}$. Let $N = 2^m$. We use $G_{(N,N)}$ to denote the m-fold Kronecker product of $G_{(2,2)}$. $G_{(N,N)}$ is a $2^m \times 2^m$ matrix over GF(2). The rows of $G_{(N,N)}$ have weights, $2^0, 2^1, 2^2, \ldots, 2^m$, and the number of rows with weight 2^{m-l} is $\binom{m}{l}$ for $0 \leq l \leq m$.

The RM codes of length $N = 2^m$ can be expressed in terms of the rows of $G_{(N,N)}$. Let $G_{\mathrm{RM}}(r,m)$ denote the matrix formed by the rows of $G_{(N,N)}$ with weights equal to or greater than 2^{m-r}. Then $G_{\mathrm{RM}}(r,m)$ is a generator matrix for the r-th order RM code, $\mathrm{RM}_{r,m}$, of length $N = 2^m$ [24]. Actually, $G_{\mathrm{RM}}(r,m)$ is the same set of rows as that given in (2.32). In the above construction of RM codes, we can also set the base matrix $G_{(2,2)}$ of (2.33) as

$$G_{(2,2)} \triangleq \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}.$$

For $m = 3$ and $r = 1$, the rows of weights 4 and 8 in $G_{(2^3,2^3)}$ of (2.35) form the generator matrix of the 1st order $(8,4)$ RM code given in Example 2.1.

Let $\boldsymbol{u} = (u_1, u_2, \ldots, u_N)$ and $\boldsymbol{v} = (v_1, v_2, \ldots, v_N)$ be two N-tuples over GF(2). From $\boldsymbol{u}$ and $\boldsymbol{v}$, we form the following binary $2N$-tuple:

$$|\boldsymbol{u}|\boldsymbol{u}+\boldsymbol{v}| \triangleq (u_1, \ldots, u_N, u_1+v_1, \ldots, u_N+v_N). \tag{2.36}$$

Let C_1 and C_2 be an (N, K_1, d_1) and an (N, K_2, d_2) binary linear codes, respectively. Assume that $d_2 > d_1$. Form the following code:

$$|C_1|C_1+C_2| \triangleq \{|\boldsymbol{u}|\boldsymbol{u}+\boldsymbol{v}| : \boldsymbol{u} \in C_1 \text{ and } \boldsymbol{v} \in C_2\}. \tag{2.37}$$

Then $|C_1|C_1+C_2|$ is an $(2N, K_1+K_2, d)$ binary linear code with

$$d = \min\{2d_1, d_2\}. \tag{2.38}$$

The above construction of a code from two component codes is called the $|u|u+v|$-construction [66] which is a powerful technique for constructing powerful long codes from short codes.

RM codes of length 2^m can be constructed from RM codes of length 2^{m-1} using the $|u|u+v|$-construction [66]. For $m \geq 2$, the r-th order RM code $\mathrm{RM}_{r,m}$ in $|u|u+v|$-construction is given below:

$$\mathrm{RM}_{r,m} = \{|u|u+v| : u \in \mathrm{RM}_{r,m-1} \text{ and } v \in \mathrm{RM}_{r-1,m-1}\} \tag{2.39}$$

with generator matrix

$$G_{\mathrm{RM}}(r,m) = \begin{bmatrix} G_{\mathrm{RM}}(r,m-1) & G_{\mathrm{RM}}(r,m-1) \\ \mathbf{0} & G_{\mathrm{RM}}(r-1,m-1) \end{bmatrix}, \tag{2.40}$$

where $\mathbf{0}$ is a zero matrix. Equation (2.40) shows that a RM code can be constructed from short RM codes by a sequence of $|u|u+v|$-constructions.

Consider a boolean function $f(x_1, x_2, \ldots, x_m)$ of m variables, $x_1, x_2, \ldots, x_m$, which take values 0 or 1. For each combination of values of $x_1, x_2, \ldots,$ and x_m, the function f takes a truth value either 0 or 1. For the 2^m combinations of values of $x_1, x_2, \ldots, x_m$, the truth values of f form a 2^m-tuple over GF(2).

For a nonnegative integer l less than 2^m, let $(b_{l1}, b_{l2}, \ldots, b_{lm})$ be the standard binary representation of l, such that $l = b_{l1} + b_{l2}2 + b_{l3}2^2 + \cdots + b_{lm}2^{m-1}$. For a given boolean function $f(x_1, x_2, \ldots, x_m)$, we form the following 2^m-tuple (truth vector):

$$v = (v_1, v_2, \ldots, v_{l+1}, \ldots, v_{2^m}) \tag{2.41}$$

where

$$v_{l+1} \triangleq f(b_{l1}, b_{l2}, \ldots, b_{lm}) \tag{2.42}$$

and $(b_{l1}, b_{l2}, \ldots, b_{lm})$ is the standard binary representation of the index integer l. We say that the boolean function $f(x_1, x_2, \ldots, x_m)$ represents the vector v. We use the notation $b(f)$ for the vector represented by $f(x_1, x_2, \ldots, x_m)$.

For $1 \leq i \leq m$, consider the boolean function

$$f(x_1, x_2, \ldots, x_m) = x_i. \tag{2.43}$$

It is easy to see that this boolean function represent the vector v_i defined by (2.31). For $1 \leq i, j \leq m$, the function

$$f(x_1, x_2, \ldots, x_m) = x_i x_j \tag{2.44}$$

represents the logic product of v_i and v_j, represented by $g(x_1, x_2, \ldots, x_m) = x_i$ and $h(x_1, x_2, \ldots, x_m) = x_j$, respectively. For $1 < i_1 < i_2 < \cdots < i_r \leq m$, the boolean function

$$f(x_1, x_2, \ldots, x_m) = x_{i_1} x_{i_2} \cdots x_{i_r} \tag{2.45}$$

represents the logic product of $v_{i_1}, v_{i_2}, \ldots,$ and v_{i_r}. Therefore, the generator vectors of the r-th order RM code of length $N = 2^m$ are represented by the boolean functions in the following set

$$\begin{aligned} B(r,m) = \{1, x_1, x_2, \ldots, x_m, x_1x_2, x_1x_3, \ldots, x_{m-1}x_m, \\ \ldots \text{ up to all products of } r \text{ variables}\}. \end{aligned} \tag{2.46}$$

Let $P(r,m)$ denote the set of all boolean functions (or polynomials) of degree r or less with m variables. Then

$$\mathrm{RM}_{r,m} = \{b(f) : f \in P(r,m)\}. \tag{2.47}$$

Finally, we want to point out that the dual code of the r-th order RM code, $\mathrm{RM}_{r,m}$, is the $(m-r-1)$-th order RM code, $\mathrm{RM}_{m-r-1,m}$.

3 TRELLIS REPRESENTATION OF LINEAR BLOCK CODES

A code trellis is a graphical representation of a code, block or convolutional, in which every path represents a codeword (or code sequence). This representation makes it possible to implement maximum likelihood decoding (MLD) of a code with a significant reduction in decoding complexity. Chapter 3 presents the fundamental concepts and basic structural properties of trellises for linear block codes. An encoder with finite memory for a linear code is modeled as a finite-state machine. With this model, representation of the dynamic behavior of the encoder by a trellis diagram is easy to conceive. During an encoding interval, the state of the encoder at a specific time instant is simply defined by the information bits stored in the memory which affect both the past and future outputs of the encoder. To facilitate the construction of a code trellis, the generator matrix of a code is put in trellis oriented form. From this trellis oriented generator matrix, some basic structural properties can be derived.

3.1 TRELLIS REPRESENTATION OF CODES

An encoder for a linear code C with a finite memory, for which the output code bits at any time instant during an encoding interval $\Gamma = \{0, 1, 2, \ldots\}$ are uniquely determined by the current input information bits and the state of the encoder at the time can be modeled as a finite-state machine. The dynamic behavior of such an encoder can be graphically represented by a state diagram expanded in time, called a **trellis diagram** (or simply trellis) as shown in Figure 3.1.

The encoder starts from some initial state, denoted σ_0. At any time instant i during its encoding interval Γ, the encoder resides in one and only one allowable state in a finite set. In the trellis diagram, the set of allowable states at time-i is represented by a set of vertices (or nodes) at the i-th level, one for each allowable state. The encoder moves from one allowable state at one time instant to another allowable state at the next time instant in one unit of time. This is called a **state transition** which, in the trellis diagram, is represented by a directed edge (or branch) connecting the starting state to the destination state. Each edge is labeled with the code bits that are generated during the state transition. The set of allowable states at a given time instant i is called the **state space** of the encoder at time-i, denoted $\Sigma_i(C)$. A state $\sigma_i \in \Sigma_i(C)$ is said to be reachable if there exists an information sequence that takes the encoder from the initial state σ_0 to state σ_i at time-i. Every state of the encoder is reachable from the initial state σ_0. In the trellis, every vertex at level-i for $i \in \Gamma$ is connected by a path from the initial state σ_0. The label sequence of this path is a code sequence (or a prefix of a code sequence). Every vertex in the trellis has at least one incoming edge except for the initial state and at least one outgoing edge except for a state called the final state. Encoding of an information sequence is equivalent to tracing a path in the trellis starting from the initial vertex σ_0. If the encoding interval Γ is semi infinite, the trellis continues indefinitely; otherwise it terminates at a **final state**, denoted σ_f. Convolutional codes have semi infinite trellises, while the trellises for linear block codes terminate at the end of each encoding interval.

For $i \in \Gamma$, let I_i and O_i denote the input information block and its corresponding output code block, respectively, during the interval from time-i to

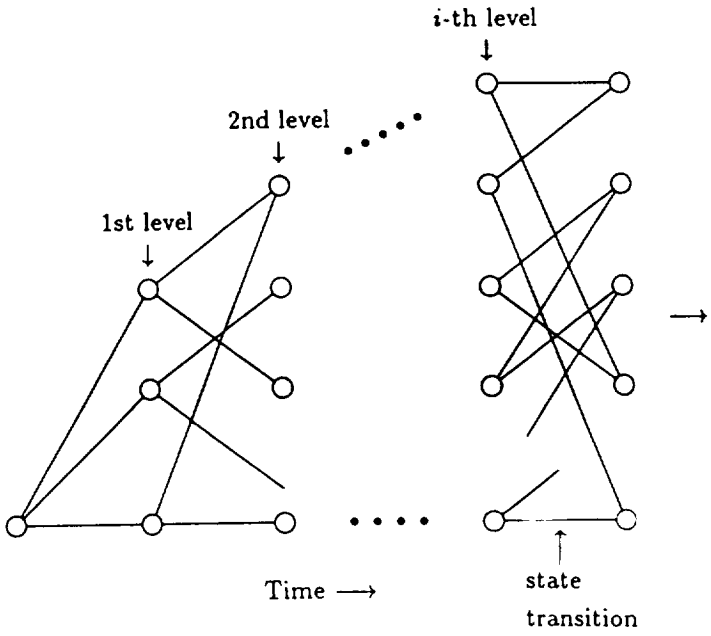

Figure 3.1. Trellis representation of a finite state encoder.

time-$(i+1)$. Then the dynamic behavior of the encoder for a linear code is governed by two functions:

(i) **Output function,**

$$O_i = f_i(\sigma_i, I_i),$$

where $f_i(\sigma_i, I_i) \neq f_i(\sigma_i, I_i')$ for $I_i \neq I_i'$.

(ii) **State transition function,**

$$\sigma_{i+1} = g_i(\sigma_i, I_i),$$

where $\sigma_i \in \Sigma_i(C)$ and $\sigma_{i+1} \in \Sigma_{i+1}(C)$ are called the **current and next states,** respectively. In the trellis diagram for C, the current and next states are connected by an edge (σ_i, σ_{i+1}) labeled with O_i.

A code trellis is said to be **time-invariant** if there exists a finite period $\{0, 1, \ldots, \nu\} \subset \Gamma$ and a state space $\Sigma(C)$ such that

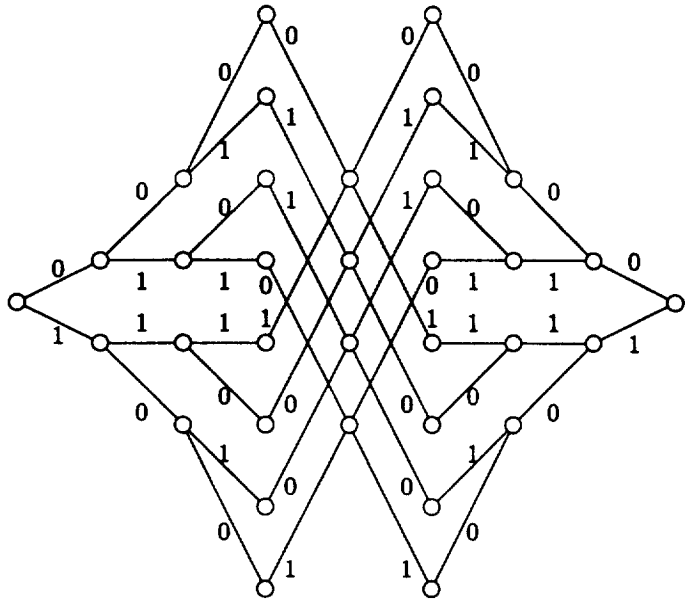

Figure 3.2. A time-varying trellis diagram for a block code.

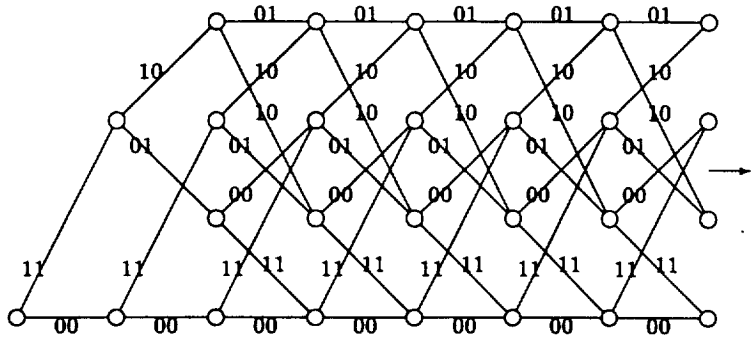

Figure 3.3. A time-invariant trellis diagram.

(1) $\Sigma_i(C) \subset \Sigma(C)$ for $0 \leq i < \nu$ and $\Sigma_i(C) = \Sigma(C)$ for $i \geq \nu$ and

(2) $f_i = f$ and $g_i = g$ for all $i \in \Gamma$.

A code trellis that is not time-invariant is said to be **time-varying**. A trellis diagram for a linear block code is, in general, time-varying. However, a trellis diagram for a convolutional code is usually time-invariant. Figure 3.2 and 3.3 depict a time-varying trellis diagram for a block code and a time-invariant trellis diagram for a convolutional code, respectively.

3.2 BIT-LEVEL TRELLISES FOR BINARY LINEAR BLOCK CODES

Consider a binary (N, K) linear block code C with generator and parity-check matrices, G and H, respectively. During each encoding interval, a message of K information bits is shifted into the encoder memory and encoded into a codeword of N code bits. The N code bits are formed and shifted onto the channel in N bit times. Therefore, the encoding span Γ is finite and consists of $N + 1$ time instants,

$$\Gamma = \{0, 1, 2, \ldots, N\}.$$

C can be represented by an N-section trellis diagram over the time span Γ. Let $\mathcal{E}(C)$ denote the encoder for C.

Definition 3.1 An N-section trellis diagram for a binary linear block code C of length N, denoted T, is a directed graph consisting of $N+1$ levels of vertices (called states) and edges (called branches) such that:

(1) For $0 \leq i \leq N$, the vertices at the i-th level represent the states in the state space $\Sigma_i(C)$ of the encoder $\mathcal{E}(C)$ at time-i. At time-0 (or the 0-th level) there is only one vertex, denoted σ_0, called the **initial vertex** (or **state**). At time-N (or the N-th level), there is only one vertex, denoted σ_f, called the **final vertex** (or **state**).

(2) For $0 < i \leq N$, a branch in the i-th section of the trellis T connects a state $\sigma_{i-1} \in \Sigma_{i-1}(C)$ to a state $\sigma_i \in \Sigma_i(C)$ and is labeled with a code bit u_i that represents the encoder output in the bit interval from time-$(i-1)$ to time-i. A branch represents a state transition.

(3) Except for the initial state, every state has at least one, but no more than two, incoming branches. Except for the final state, every state has at least one, but no more than two, outgoing branches. The initial state has no incoming branches. The final state has no outgoing branches. Two branches diverging from the same state have different labels.

(4) There is a **directed** path from the initial state σ_0 to the final state σ_f with a label sequence $(u_1, u_2, \ldots, u_N)$ if and only if $(u_1, u_2, \ldots, u_N)$ is a codeword in C.

$\Delta\Delta$

Two states in the code trellis are said to be **adjacent** if they are connected by a branch. During one encoding interval Γ, the encoder starts from the initial state σ_0, transverses a sequence of states

$$(\sigma_0, \sigma_1, \ldots, \sigma_i, \ldots, \sigma_f),$$

generates a code sequence

$$(u_1, u_2, \ldots, u_i, \ldots, u_N),$$

and then reaches the final state σ_f. The bit-level 8-section trellis diagram for the $(8, 4)$ linear block code given in Example 2.1 (Table 2.1) is shown in Figure 3.2.

For $0 \leq i \leq N$, let $|\Sigma_i(C)|$ denote the cardinality of the state space $\Sigma_i(C)$. Then, the sequence,

$$(|\Sigma_0(C)|, |\Sigma_1(C)|, \ldots, |\Sigma_{N-1}(C)|, |\Sigma_N(C)|),$$

is called the **state space complexity profile**, which is a measure of the **state complexity** of the N-section code trellis T. We will show later that for $0 \leq i \leq N$, $|\Sigma_i(C)|$ is a power of 2. Define

$$\rho_i(C) \triangleq \log_2 |\Sigma_i(C)|,$$

which is called the **state space dimension** at time-i. When there is no confusion, we simply use ρ_i for $\rho_i(C)$ for simplicity. The sequence,

$$(\rho_0, \rho_1, \ldots, \rho_N),$$

is called **state space dimension profile**. From Figure 3.2, we see that the state space complexity and dimension profiles for the $(8,4)$ code given in Example 2.1 are $(1,2,4,8,4,8,4,2,1)$ and $(0,1,2,3,2,3,2,1,0)$, respectively.

3.3 TRELLIS ORIENTED GENERATOR MATRIX

To facilitate the code trellis construction, we put the generator matrix G in a special form. Let $\boldsymbol{u} = (u_1, u_2, \ldots, u_N)$ be a nonzero binary N-tuple. The first nonzero component of $\boldsymbol{u}$ is called the **leading** '1' of $\boldsymbol{u}$ and the last nonzero component of $\boldsymbol{u}$ is called the **trailing** '1' of $\boldsymbol{u}$.

A generator matrix G for C is said to be in **trellis oriented form** (TOF) if the following two conditions hold:

(1) The leading '1' of each row appears in a column before the leading '1' of any row below it.

(2) No two rows have their trailing "ones" in the same column.

Any generator matrix for C can be put in TOF by two steps of **Gaussian elimination**.

Example 3.1 Consider the $(8,4)$ RM code given in Example 2.1 with following generator matrix,

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

It is not in TOF. By interchanging the second and the fourth rows, we have

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Add the fourth row of the above matrix to the first, second and third rows. These additions result in the following matrix in TOF:

$$G = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

△△

The span of a row $g = (g_1, g_2, \ldots, g_N)$ in a trellis oriented generator matrix (TOGM) G is defined as the **smallest interval** $\{i, i+1, \ldots, j\}$ which contains all the nonzero bits of g. This is denoted as span$(g) \triangleq [i, j]$. For a row g in a TOGM G whose span is $[i, j]$, the **active span** of g, denoted aspan(g), is defined as aspan$(g) \triangleq [i, j-1]$ for $i < j$ and aspan$(g) \triangleq \emptyset$ (empty set) for $i = j$.

Let $g_1, g_2, \ldots, g_K$ be the K rows of a TOGM G with

$$g_l = (g_{l1}, g_{l2}, \ldots, g_{lN})$$

for $1 \leq l \leq K$. Then

$$G = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_K \end{bmatrix} = \begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1N} \\ g_{21} & g_{22} & \cdots & g_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ g_{K1} & g_{K2} & \cdots & g_{KN} \end{bmatrix}.$$

Let $(a_1, a_2, \ldots, a_K)$ be the block of K information bits (called a message) to be encoded. The corresponding codeword is given by

$$\begin{aligned} u &= (u_1, u_2, \ldots, u_N) \\ &= a_1 \cdot g_1 + a_2 \cdot g_2 + \cdots + a_K \cdot g_K. \end{aligned}$$

We see that the l-th information bit a_l affects the output u of the encoder $\mathcal{E}(C)$ over the span of the l-th row g_l of the TOGM G. This span(g_l) may be regarded as the **constraint length** of the code associated with the l-th input information bit a_l.

At time-i with $1 \leq i \leq N$, the number of information bits that affect the next output code bit u_{i+1} is equal to the number of rows in G whose active spans contain i. These information bits define the state of the encoder at time-i.

3.4 STATE SPACE FORMULATION

In the following, we give a mathematical formulation of the state space of the N-section trellis for an (N, K) linear block code C over GF(2) with a TOGM G.

At time-i, $0 \leq i \leq N$, the rows of G are divided into three disjoint subsets:

(1) G_i^p consists of those rows of G whose spans are contained in the interval $[1, i]$.

(2) G_i^f consists of those rows of G whose spans are contained in the interval $[i+1, N]$.

(3) G_i^s consists of those rows of G whose active spans contain i.

Let A_i^p, A_i^f and A_i^s denote the subsets of information bits that correspond to the rows of G_i^p, G_i^f and G_i^s, respectively. The information bits in A_i^p do not affect the encoder outputs after time-i, and hence they become the past with respect to time-i. The information bits in A_i^f only affect the encoder outputs after time-i. Since the active spans of the rows in G_i^s contain the time instant i, the information bits in A_i^s affect not only the past encoder outputs up to time-i but also the future encoder outputs beyond time-i. We say that the information bits in A_i^s define a state of encoder $\mathcal{E}(C)$ for the code C at time-i. Let $\rho_i \triangleq |A_i^s| = |G_i^s|$. Then there are 2^{ρ_i} distinct states that the encoder $\mathcal{E}(C)$ can occupy at time-i; each state is defined by a specific combination of the ρ_i information bits in A_i^s. These states form the state space $\Sigma_i(C)$ of the encoder $\mathcal{E}(C)$ (or simply of the code C). The parameter ρ_i is the dimension of the state space $\Sigma_i(C)$. In the trellis representation of C, the states in $\Sigma_i(C)$ are represented by 2^{ρ_i} vertices at the i-th level of the trellis.

Example 3.2 Consider the TOGM G for the $(8, 4)$ RM code given in Example 3.1. The spans of the four rows are: $\text{span}(g_1) = [1, 4]$, $\text{span}(g_2) = [2, 7]$, $\text{span}(g_3) = [3, 6]$, and $\text{span}(g_4) = [5, 8]$. Their active spans are therefore: $\text{aspan}(g_1) = [1, 3]$, $\text{aspan}(g_2) = [2, 6]$, $\text{aspan}(g_3) = [3, 5]$ and $\text{aspan}(g_4) = [5, 7]$. For each i with $0 \leq i \leq 8$, counting the number of rows which are active at time-i yields the state space dimension profile $(0, 1, 2, 3, 2, 3, 2, 1, 0)$.

△△

The above formulation of a state space actually provides a sequential machine model for the encoder $\mathcal{E}(C)$.

3.5 STATE TRANSITION AND OUTPUT

For $0 \leq i \leq N$, suppose the encoder $\mathcal{E}(C)$ is in state $\sigma_i \in \Sigma_i(C)$. From time-i to time-$(i+1)$, $\mathcal{E}(C)$ generates a code bit u_{i+1} and moves from state σ_i to a state $\sigma_{i+1} \in \Sigma_{i+1}(C)$. Let

$$G_i^s = \{g_1^{(i)}, g_2^{(i)}, \ldots, g_{\rho_i}^{(i)}\} \tag{3.1}$$

and

$$A_i^s = \{a_1^{(i)}, a_2^{(i)}, \ldots, a_{\rho_i}^{(i)}\} \tag{3.2}$$

where $\rho_i = |G_i^s|$. The current state σ_i of the encoder is defined by a specific combination of the information bits in A_i^s.

Let g^* be the row in G_i^f whose leading '1' is at position-$(i+1)$. The uniqueness of this row g^* (if it exists) is guaranteed by the first condition in the definition of a generator matrix in TOF given in section 3.3. Let g_{i+1}^* denote the $(i+1)$-th component of g^*. Then $g_{i+1}^* = 1$. Let a^* denote the information bit that corresponds to row g^*. It follows from (2.3) and the structure of the TOGM G that the output code bit u_{i+1} generated during the bit interval between time-i and time-$(i+1)$ is given by

$$u_{i+1} = a^* + \sum_{l=1}^{\rho_i} a_l^{(i)} g_{l,i+1}^{(i)}, \tag{3.3}$$

where $g_{l,i+1}^{(i)}$ is the $(i+1)$-th component of $g_l^{(i)}$ in G_i^s. Note that a^* begins to affect the output of the encoder $\mathcal{E}(C)$ at time-$(i+1)$. For this reason, the bit a^* is regarded as the **current input information bit**. The second term in (3.3) is the contribution from the state σ_i defined by the information bits in $A_i^s = \{a_1^{(i)}, a_2^{(i)}, \ldots, a_{\rho_i}^{(i)}\}$ which are stored in memory. From (3.3), we see that the current output u_{i+1} is uniquely determined by the current state σ_i of the encoder $\mathcal{E}(C)$ and the current input a^*. The output bit u_{i+1} can have two possible values depending on the current input information bit a^*; each value takes the encoder $\mathcal{E}(C)$ to a different state at time-$(i+1)$. That is, there are two possible transitions from the current state σ_i to two states in $\Sigma_{i+1}(C)$ at

time-$(i+1)$. In the code trellis, there are two edges (or branches) diverging from the vertex σ_i labeled with '0' and '1', respectively.

Suppose there is no such row g^* in G_i^f. Then the output code bit is given by

$$u_{i+1} = \sum_{l=1}^{\rho_i} a_l^{(i)} \cdot g_{l,i+1}^{(i)}. \tag{3.4}$$

In this case, we may regard that the current input information bit a^* is being set to "0", i.e. $a^* = 0$ (this is called a **dummy** information bit). The output code bit u_{i+1} can take only one value given by (3.4) and there is **only one** possible transition from the current state σ_i to a state in $\Sigma_{i+1}(C)$. In the trellis T, there is only one branch diverging from the vertex σ_i.

Example 3.3 Again we consider the $(8,4)$ code with its TOGM G given in Example 3.1. Consider time-2. Then we find that $G_2^p = \emptyset, G_2^f = \{g_3, g_4\}$ and $G_2^s = \{g_1, g_2\}$. Therefore, the information bits a_1 and a_2 define the state of the encoder at time-2 and there are 4 distinct states defined by four combinations of values of a_1 and a_2, $\{00, 01, 10, 11\}$. We also see that $g^* = g_3$. Therefore, the current input information bit is $a^* = a_3$. The current output code bit u_3 is given by

$$\begin{aligned} u_3 &= a_3 + a_1 \cdot g_{13} + a_2 \cdot g_{23} \\ &= a_3 + a_1. \end{aligned}$$

For every state defined by a_1 and a_2, u_3 has two possible values depending on a_3. In the trellis, there are two branches diverging from each state at time-2, as shown in Figure 3.2.

Now consider time-3. At $i = 3$, we find that $G_3^p = \emptyset$, $G_3^f = \{g_4\}$ and $G_3^s = \{g_1, g_2, g_3\}$. Therefore, the information bits a_1, a_2 and a_3 define 8 states at time-3, as shown in Figure 3.2. There is no row g^* in G_3^f with leading '1' at position (or time) $i = 4$. Hence we set the current input information bit $a^* = 0$. The output code bit u_4 is given by

$$\begin{aligned} u_4 &= a_1 \cdot g_{14} + a_2 \cdot g_{24} + a_3 \cdot g_{34} \\ &= a_1 + a_2 + a_3. \end{aligned}$$

In the trellis, there is only one branch diverging from each of the 8 states, as shown in Figure 3.2.

$\triangle\triangle$

Let g^0 be the row in G_i^s whose trailing '1' is at the position-$(i+1)$. (Note that this row g^0 may not exist.) The uniqueness of the row g^0 (if it exists) is guaranteed by the second condition of a generator matrix in TOF given in Section 3.3. Let a^0 be the information bit in A_i^s that corresponds to row g^0. Then at time-$(i+1)$,

$$G_{i+1}^s = (G_i^s \backslash \{g^0\}) \cup \{g^*\} \tag{3.5}$$

and

$$A_{i+1}^s = (A_i^s \backslash \{a^0\}) \cup \{a^*\}. \tag{3.6}$$

The information bits in A_{i+1}^s define the state space $\Sigma_{i+1}(C)$ at time-$(i+1)$. The change from A_i^s to A_{i+1}^s defines a state transition from the current state σ_i defined by A_i^s to the next state σ_{i+1} defined by A_{i+1}^s. Therefore from A_i^s, A_{i+1}^s, (3.3) and (3.4), we can construct the N-section code trellis T for C.

The construction of the N-section trellis T is carried out serially, section by section. Suppose the trellis has been constructed up to section-i. Now we want to construct the $(i+1)$-th section from time-i to time-$(i+1)$. The state space $\Sigma_i(C)$ is known. The $(i+1)$-th section is constructed by taking the following steps:

(1) Determine G_{i+1}^s and A_{i+1}^s from (3.5) and (3.6). Form the state space $\Sigma_{i+1}(C)$ at time-$(i+1)$.

(2) For each state $\sigma_i \in \Sigma_i(C)$, determine its state transition(s) following the state transition rules given above. Connect σ_i to its adjacent state(s) in $\Sigma_{i+1}(C)$ by edge(s).

(3) For each state transition, determine the output code bit u_{i+1} from the output function of (3.3) or (3.4), and label the corresponding edge in the trellis with u_{i+1}.

3.6 TIME-VARYING STRUCTURE

During the encoding interval $\Gamma = \{0, 1, \ldots, N\}$, the output function of the encoder $\mathcal{E}(C)$ changes between (3.3) and (3.4). Also, the set $\{g_{1,i+1}^{(i)}, g_{2,i+1}^{(i)}, \ldots, g_{\rho_i,i+1}^{(i)}\}$ in the summations of (3.3) and (3.4) may change from one time instant to another. This is because each column in the TOGM is, in general, not a downward shift of the column before it. Therefore, the output function of $\mathcal{E}(C)$ is time-varying. As the encoder $\mathcal{E}(C)$ moves from time-i to time-$(i+1)$, its state space may also change, i.e., $\Sigma_{i+1}(C) \neq \Sigma_i(C)$. Consequently, the trellis for $\mathcal{E}(C)$ is time-varying.

To describe the time-varying state space of $\mathcal{E}(C)$, there are four cases to consider.

Case I: There is no such row g^0 in G_i^s, but there is a row g^* in G_i^f. As the encoder moves from time-i to time-$(i+1)$, the active span of g^* contains the time instant $i+1$. Therefore, g^* is added to the set G_i^s to form G_{i+1}^s. The information bit a^* that corresponds to g^* is now in the encoder memory and is included in determining the next and future states of the encoder. The next state σ_{i+1} is determined by the information bits

$$a_1^{(i)}, a_2^{(i)}, \ldots, a_{\rho_i}^{(i)}, a^*.$$

Since $|G_{i+1}^s| = |G_i^s| + 1$, $\rho_{i+1} = \rho_i + 1$. This results in state space expansion.

Case II: There is a row $g^0 \in G_i^s$ and a row $g^* \in G_i^f$. When the encoder moves from time-i to time-$(i+1)$, the span of g^0 moves into the interval $[1, i+1]$ and g^0 is replaced by g^* in G_{i+1}^s. In this case, the information bit a^0 that corresponds to g^0 becomes part of the past with respect to time-$(i+1)$ and will not affect the encoder outputs further; however, the information bit a^* is now in the memory and is included in determining the next and future states of the encoder. Assuming that $a^0 = a_1^{(i)}$, the next state σ_{i+1} of the encoder is then determined by the information bits

$$a_2^{(i)}, a_3^{(i)}, \ldots, a_{\rho_i}^{(i)}, a^*.$$

Therefore, from time-i to time-$(i+1)$, the state space of the encoder and its dimension remain the same, i.e., $\rho_{i+1} = \rho_i$.

Case III: There is no such row g^0 in G_i^s and no such row g^* in G_i^f. In this case, $G_{i+1}^s = G_i^s$ and there in no change in the state space dimension as the encoder moves from time-i to time-$(i+1)$. The next state is determined by the same set of information bits as at time-i, i.e.,

$$a_1^{(i)}, a_2^{(i)}, \ldots, a_{\rho_i}^{(i)}.$$

Case IV: There exists a row $g^0 \in G_i^s$ but there is no such row $g^* \in G_i^f$. In this case, g^0 is excluded from G_i^s to form G_{i+1}^s and its corresponding information bit a^0 becomes part of the past as the encoder moves from time-i to time-$(i+1)$. Assuming that $a^0 = a_1^{(i)}$, the state σ_{i+1} of the encoder is determined by the information bits

$$a_2^{(i)}, a_3^{(i)}, \ldots, a_{\rho_i}^{(i)}.$$

Consequently, $|G_{i+1}^s| = |G_i^s| - 1$ and $\rho_{i+1} = \rho_i - 1$. This results in state space reduction.

Example 3.4 Consider the $(8,4)$ code given in Example 3.1. From its TOGM G, we see that for $i = 0, 1$ and 2, there is no such row g^0 in G_i^s, but there is a row g^* in G_i^f. Hence there is state space expansion from time-0 to time-3 as shown Figure 3.2. We note that there is such a row g^0 in G_3^s and there is no such row g^* in G_3^f. Therefore, there is state space reduction from time-3 to time-4, as shown in Figure 3.2.

$\triangle\triangle$

From the above analysis of the N-section trellis for an (N, K) linear block code C, we have the following observations. At time-i, with $0 \leq i \leq N$,

(1) The information bits in A_i^p become the past and do not affect the future outputs of the encoder beyond time-i.

(2) The information bits in A_i^f affect the encoder outputs only beyond time-i, i.e., they are the future input information bits.

(3) The information bits in A_i^s are the bits stored in the encoder memory that define the encoder state at time-i.

The above observations make the formulation of a trellis diagram for a block code the same as for a convolutional code [62].

3.7 STRUCTURAL PROPERTIES

For $0 \leq i < j \leq N$, let $C_{i,j}$ denote the subcode of C consisting of those codewords in C whose nonzero components are confined to the span of $j-i$ consecutive positions in the set $\{i+1, i+2, \ldots, j\}$. Clearly, every codeword in $C_{i,j}$ is of the form,

$$(\underbrace{0,0,\ldots,0}_{i}, u_{i+1}, u_{i+2}, \ldots, u_j, \underbrace{0,0,\ldots,0}_{N-j}).$$

It follows from the definition of $C_{i,j}$ and the structure of the TOGM G for C that $C_{i,j}$ is spanned by those rows in G whose spans are contained in the interval $[i+1, j]$. The two subcodes, $C_{0,i}$ and $C_{i,N}$, are spanned by the rows in G_i^p and G_i^f, respectively, and they are called the **past** and **future** subcodes of C with respect to time-i.

For a linear code D, let $k(D)$ denote its dimension. Then, $k(C_{0,i}) = |G_i^p|$ and $k(C_{i,N}) = |G_i^f|$. Recall that the dimension of the state space $\Sigma_i(C)$ at time-i is

$$\begin{aligned} \rho_i(C) &= |G_i^s| = K - |G_i^p| - |G_i^f| \\ &= K - k(C_{0,i}) - k(C_{i,N}). \end{aligned} \tag{3.7}$$

This gives a relationship between the state space dimension $\rho_i(C)$ at time-i and the dimensions of the past and future subcodes, $C_{0,i}$ and $C_{i,N}$, of C with respect to time-i.

Note that $C_{0,i}$ and $C_{i,N}$ have only the all-zero codeword $\mathbf{0}$ in common. The direct-sum of $C_{0,i}$ and $C_{i,N}$, denoted $C_{0,i} \oplus C_{i,N}$, is a subcode of C with dimension

$$k(C_{0,i}) + k(C_{i,N}).$$

Let $C/(C_{0,i} \oplus C_{i,N})$ denote the **partition** of C with respect to $C_{0,i} \oplus C_{i,N}$. Then this partition consists of

$$\begin{aligned} |C/(C_{0,i} \oplus C_{i,N})| &= 2^{K-k(C_{0,i})-k(C_{i,N})} \\ &= 2^{\rho_i} \end{aligned} \tag{3.8}$$

cosets of $C_{0,i} \oplus C_{i,N}$. Eq.(3.8) says that the number of states in the state space $\Sigma_i(C)$ at time-i is equal to the number of cosets in the partition $C/(C_{0,i} \oplus C_{i,N})$.

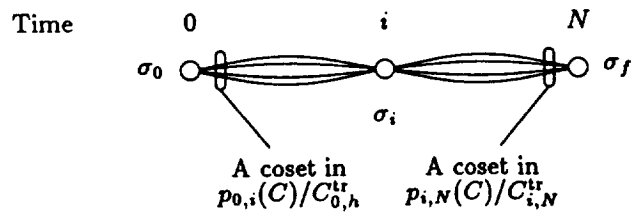

Figure 3.4. The paths in the code trellis that represent the $2^{K-\rho_i}$ codewords in $v \oplus (C_{0,i} \oplus C_{i,N})$.

Let S_i denote the subspace of C that is spanned by the rows in G_i^s. Then each codeword in S_i is given by

$$\begin{aligned} v &= (a_1^{(i)}, a_2^{(i)}, \ldots, a_{\rho_i}^{(i)}) \cdot G_i^s \\ &= a_1^{(i)} \cdot g_1^{(i)} + a_2^{(i)} \cdot g_2^{(i)} + \cdots + a_{\rho_i}^{(i)} \cdot g_{\rho_i}^{(i)} \end{aligned} \tag{3.9}$$

where $a_l^{(i)} \in A_i^s$ for $1 \leq l \leq \rho_i$. The 2^{ρ_i} codewords in S_i can be used as the representatives for the cosets in the partition $C/(C_{0,i} \oplus C_{i,N})$. Therefore, S_i is the coset representative space for the partition $C/(C_{0,i} \oplus C_{i,N})$. From (3.9), we see that there is **one-to-one correspondence** between v and the state $\sigma_i \in \Sigma_i(C)$ defined by $(a_1^{(i)}, a_2^{(i)}, \ldots, a_{\rho_i}^{(i)})$. Since there is a **one-to-one correspondence** between v and a coset in $C/(C_{0,i} \oplus C_{i,N})$, therefore, there is a **one-to-one correspondence** between a state in the state space $\Sigma_i(C)$ and a coset in the partition $C/(C_{0,i} \oplus C_{i,N})$.

With the above one-to-one correspondence in the trellis T, the codeword v given by (3.9) is represented by a path that passes through the state σ_i defined by the information bits, $a_1^{(i)}$, $a_2^{(i)}$, ..., $a_{\rho_i}^{(i)}$ (i.e., a path that connects the initial state σ_0 to the final state σ_f through the state σ_i). If we fix the information bits, $a_1^{(i)}$, $a_2^{(i)}$, ..., $a_{\rho_i}^{(i)}$, and allow the other $K - \rho_i$ information bits to vary, we obtain $2^{K-\rho_i}$ codewords of C in the coset

$$v \oplus (C_{0,i} \oplus C_{i,N}) \triangleq \{v + u : u \in C_{0,i} \oplus C_{i,N}\} \tag{3.10}$$

with v as the coset representative. In the trellis, these $2^{K-\rho_i}$ codewords are represented by paths that connect the initial state σ_0 to the final state σ_f

through the state σ_i at time-i defined by the information bits, $a_1^{(i)}$, $a_2^{(i)}$, ..., $a_{\rho_i}^{(i)}$, as shown in Figure 3.4. Note that

$$K - \rho_i = k(C_{0,i}) + k(C_{i,N}) \tag{3.11}$$

which is simply the dimension of $C_{0,i} \oplus C_{i,N}$.

For $0 \leq i < j \leq N$, let $p_{i,j}(C)$ denote the linear code of length $j-i$ obtained from C by removing the first i and last $N-j$ components of each codeword in C. This code is called a **punctured (or truncated)** code of C. Let $C_{i,j}^{\mathrm{tr}}$ denote the punctured code of the subcode $C_{i,j}$, i.e.,

$$C_{i,j}^{\mathrm{tr}} \triangleq p_{i,j}(C_{i,j}). \tag{3.12}$$

It follows from the structure of the TOGM G that

$$k(p_{i,j}(C)) = K - k(C_{0,i}) - k(C_{j,N}) \tag{3.13}$$

and

$$k(C_{i,j}^{\mathrm{tr}}) = k(C_{i,j}). \tag{3.14}$$

Consider the punctured code $p_{0,i}(C)$. Partition $p_{0,i}(C)$ based on $C_{0,i}^{\mathrm{tr}}$. It follows from (3.13) and (3.14) that the partition $p_{0,i}(C)/C_{0,i}^{\mathrm{tr}}$ consists of

$$2^{K-k(C_{0,i})-k(C_{i,N})} = 2^{\rho_i}$$

cosets of $C_{0,i}^{\mathrm{tr}}$. We can readily see that there is a **one-to-one correspondence** between the cosets in $p_{0,i}(C)/C_{0,i}^{\mathrm{tr}}$ and the cosets in $C/(C_{0,i} \oplus C_{i,N})$, and hence a **one-to-one correspondence** between the cosets in $p_{0,i}(C)/C_{0,i}^{\mathrm{tr}}$ and the states in the state space $\Sigma_i(C)$. The codewords in a coset in $p_{0,i}(C)/C_{0,i}^{\mathrm{tr}}$ are simply the prefixes of the codewords in its corresponding coset in $C/(C_{0,i} \oplus C_{i,N})$. Hence the codewords in a coset of $p_{0,i}(C)/C_{0,i}^{\mathrm{tr}}$ will take the encoder $\mathcal{E}(C)$ to a unique state $\sigma_i \in \Sigma_i(C)$. In the trellis T, they are the paths connecting the initial state σ_0 to the state σ_i as shown in Figure 3.4. Let $L(\sigma_0, \sigma_i)$ denote the paths in the trellis T that connect the initial state σ_0 to the state σ_i in $\Sigma_i(C)$. Then $L(\sigma_0, \sigma_i)$ is a coset in the partition $p_{0,i}(C)/C_{0,i}^{\mathrm{tr}}$.

Now we consider the punctured code $p_{i,N}(C)$. Partition $p_{i,N}(C)$ based on $C_{i,N}^{\mathrm{tr}} = p_{i,N}(C_{i,N})$. Then it follow from (3.13) and (3.14) that the partition

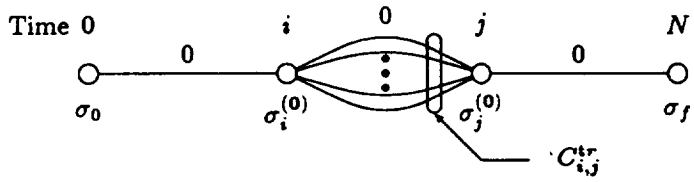

Figure 3.5. Paths in the code trellis that represent the codewords in $C_{i,j}$.

$p_{i,N}(C)/C_{i,N}^{tr}$ consists of

$$2^{K-k(C_{0,i})-k(C_{i,N})} = 2^{\rho_i}$$

cosets of $C_{i,N}^{tr}$. Again we see that there is a **one-to-one correspondence** between the cosets in $p_{i,N}(C)/C_{i,N}^{tr}$ and the cosets in $C/(C_{0,i} \oplus C_{i,N})$, and hence a **one-to-one correspondence** between the cosets in $p_{i,N}(C)/C_{i,N}^{tr}$ and the states in the state space $\Sigma_i(C)$. In the trellis, the codewords in a coset $p_{i,N}(C)/C_{i,N}^{tr}$ form the paths that connect a state $\sigma_i \in \Sigma_i(C)$ to the final state σ_f as shown in Figure 3.4. Let $L(\sigma_i, \sigma_f)$ denote the paths in the trellis T that connect the state $\sigma_i \in \Sigma_i(C)$ to the final state σ_f. Then $L(\sigma_i, \sigma_f)$ is a coset in the partition $p_{i,N}(C)/C_{i,N}^{tr}$.

For $0 \leq i < j \leq N$, let $\sigma_i^{(0)}$ and $\sigma_j^{(0)}$ denote two states on the all-zero path **0** in the trellis T at time-i and time-j, respectively. Let $L(\sigma_i^{(0)}, \sigma_j^{(0)})$ denote the paths of length $j-i$ in T that connect $\sigma_i^{(0)}$ to $\sigma_j^{(0)}$. Consider the paths in T that start from the initial state σ_0, follow the all-zero path **0** to the state $\sigma_i^{(0)}$, transverse through the paths in $L(\sigma_i^{(0)}, \sigma_j^{(0)})$ to the state $\sigma_j^{(0)}$, then follow the all-zero path **0** until they reach the final state σ_f as shown in Figure 3.5. These paths represent the codewords in the subcode $C_{i,j}$ of C. This implies that

$$L(\sigma_i^{(0)}, \sigma_j^{(0)}) = C_{i,j}^{tr}. \tag{3.15}$$

Let $v = (v_1, v_2, \ldots, v_N)$ be a path in the code trellis T. For $0 \leq i < j \leq N$, let $\sigma_i^{(v)}$ and $\sigma_j^{(v)}$ be two states on the path v at time-i and time-j, respectively. Let $L(\sigma_i^{(v)}, \sigma_j^{(v)})$ denote the paths of length $j-i$ that connect $\sigma_i^{(v)}$ to $\sigma_j^{(v)}$. Consider the paths in T that start form the initial state σ_0, follow the path v to the state $\sigma_i^{(v)}$, transverse through the paths in $L(\sigma_i^{(v)}, \sigma_j^{(v)})$, then follow the

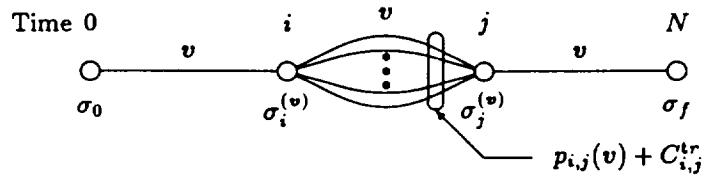

Figure 3.6. Paths in the code trellis that represent the codewords in the coset $u \oplus C_{i,j}$.

path v until they reach the final state σ_f as shown in Figure 3.6. These paths represent the codewords in the following coset of $C_{i,j}$:

$$v \oplus C_{i,j} \triangleq \{v + u : u \in C_{i,j}\}. \tag{3.16}$$

This is a coset in the partition $C/C_{i,j}$. This implies that $L(\sigma_i^{(v)}, \sigma_j^{(v)})$ is a coset of $C_{i,j}^{tr}$ in $p_{i,j}(C)$, i.e.,

$$L(\sigma_i^{(v)}, \sigma_j^{(v)}) = p_{i,j}(v) + C_{i,j}^{tr} \in p_{i,j}(C)/C_{i,j}^{tr}, \tag{3.17}$$

where $p_{i,j}(v)$ denotes the vector of length $j-i$ obtained from v by removing the first i and last $N-j$ components of v. For any two connected states $\sigma_i \in \Sigma_i(C)$ and $\sigma_j \in \Sigma_j(C)$ with $0 \leq i < j \leq N$, they must be on a path in the trellis T. It follows from (3.17) that

$$L(\sigma_i, \sigma_j) \in p_{i,j}(C)/C_{i,j}^{tr}. \tag{3.18}$$

Therefore, the number of paths that connect a state $\sigma_i \in \Sigma_i(C)$ to a state $\sigma_j \in \Sigma_j(C)$ is given by

$$|L(\sigma_i, \sigma_j)| = \begin{cases} 2^{k(C_{i,j}^{tr})}, & \text{if } \sigma_i \text{ and } \sigma_j \text{ are connected,} \\ 0, & \text{if } \sigma_i \text{ and } \sigma_j \text{ are not connected.} \end{cases} \tag{3.19}$$

For $0 \leq i < j < k \leq N$, let $\Sigma_j(\sigma_i, \sigma_k)$ denote the set of states in $\Sigma_j(C)$ through which the path in $L(\sigma_i, \sigma_k)$ connect the state σ_i to the state σ_k as shown in Figure 3.7. Let

$$L(\sigma_i, \sigma_j) \circ L(\sigma_j, \sigma_k) \triangleq \{u \circ v : u \in L(\sigma_i, \sigma_j), v \in L(\sigma_j, \sigma_k)\} \tag{3.20}$$

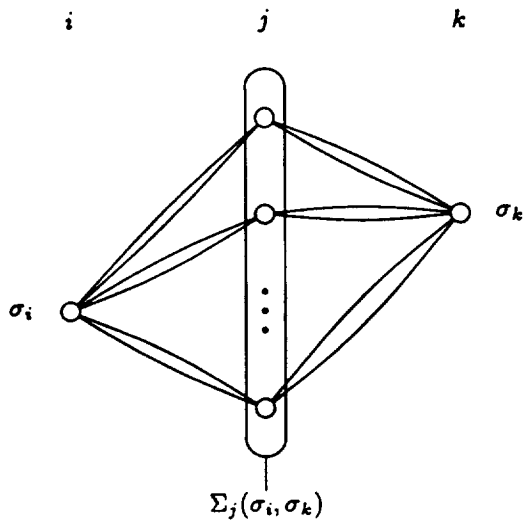

Figure 3.7. The state set $\Sigma_j(\sigma_i, \sigma_k)$.

where $u \circ v$ denotes the concatenation of two sequences u and v. In the trellis, $L(\sigma_i, \sigma_j) \circ L(\sigma_j, \sigma_k)$ consists of those paths in $L(\sigma_i, \sigma_k)$ that connect the state σ_i to the state σ_k through the state σ_j. Then,

$$L(\sigma_i, \sigma_k) = \bigcup_{\sigma_j \in \Sigma_j(\sigma_i, \sigma_k)} L(\sigma_i, \sigma_j) \circ L(\sigma_j, \sigma_k). \tag{3.21}$$

The above developments give some fundamental structural properties of an N-section trellis T for a linear block code C.

4 STATE LABELING, TRELLIS CONSTRUCTION PROCEDURES AND TRELLIS SYMMETRY

The construction of a code trellis can be facilitated by labeling the states at each level of the trellis. State labeling is also necessary in the implementation of a trellis-based decoding algorithm. This chapter presents three methods for labeling the states of the N-section trellis for an (N, K) linear block code. The first two methods are based on the information set that defines the state space at a particular encoding time instant and the third method is based on the parity-check matrix of the code. The first two methods are more efficient than the third one for codes with $K < N - K$; however, the third method is more efficient for codes with $N - K < K$. Based on these labeling methods, construction procedures for the N-section trellis for an (N, K) linear block code are presented. Also presented in this chapter is the mirror symmetry structure of a code trellis. This symmetry structure is useful in decoding.

4.1 STATE LABELING BY THE STATE DEFINING INFORMATION SET

In a code trellis, each state is labeled by a fixed sequence (or given a name). This can be accomplished by using a K-tuple λ with components corresponding to the K information bits, $a_1, a_2, \ldots, a_K$, in a message. At time-i, all the components of λ are set to zero except for the components at the positions corresponding to the information bits in $A_i^s = \{a_1^{(i)}, a_2^{(i)}, \ldots, a_{\rho_i}^{(i)}\}$. Every combination of the ρ_i bits at the positions corresponding to the information bits in A_i^s gives the label $l(\sigma_i)$ for the state σ_i defined by the information bits, $a_1^{(i)}, a_2^{(i)}, \ldots, a_{\rho_i}^{(i)}$.

Example 4.1 Consider the $(8, 4)$ code given in Example 3.1. At time-4, we find that $A_4^s = \{a_2, a_3\}$. There are 4 states corresponding to 4 combinations of a_2 and a_3. Therefore, the label for each of these 4 states is given by $(0, a_2, a_3, 0)$.

$\Delta\Delta$

The construction of the N-section trellis for an (N, K) linear block code C can be carried out as follows. Suppose the trellis T has been constructed up to section-i. At this point, G_i^s, A_i^s and $\Sigma_i(C)$ are known. Each state $\sigma_i \in \Sigma_i(C)$ is labeled by a K-tuple. The $(i+1)$-th section is constructed by taking the following steps:

(1) Determine G_{i+1}^s and A_{i+1}^s from (3.5) and (3.6).

(2) Form the state space $\Sigma_{i+1}(C)$ at time-$(i+1)$ and label each state in $\Sigma_{i+1}(C)$ based on A_{i+1}^s. The state in $\Sigma_{i+1}(C)$ form the vertices of the code trellis T at the $(i+1)$-th level.

(3) For each state $\sigma_i \in \Sigma_i(C)$ at time-i, determine its transition(s) to the state(s) in $\Sigma_{i+1}(C)$ based on the information bits of a^* and a^0. For each transition from a state $\sigma_i \in \Sigma_i(C)$ to a state $\sigma_{i+1} \in \Sigma_{i+1}(C)$, connect the state σ_i to the state σ_{i+1} by an edge (σ_i, σ_{i+1}).

(4) For each state transition (σ_i, σ_{i+1}), determine the output code bit u_{i+1} and label the edge (σ_i, σ_{i+1}) with u_{i+1}.

Recall that at time-i, there are two branches diverging from a state in $\Sigma_i(C)$ if there exists a current information bit a^*. One branch corresponds to $a^* =$

Table 4.1. State defining sets and state labels for the 8-section trellis for the $(8, 4)$ linear block code.

i	G_i^s	a^*	a^0	A_i^s	State Label
0	$\emptyset$	a_1	–	$\emptyset$	(0000)
1	$\{g_1\}$	a_2	–	$\{a_1\}$	(a_1000)
2	$\{g_1, g_2\}$	a_3	–	$\{a_1, a_2\}$	(a_1a_200)
3	$\{g_1, g_2, g_3\}$	–	a_1	$\{a_1, a_2, a_3\}$	$(a_1a_2a_30)$
4	$\{g_2, g_3\}$	a_4	–	$\{a_2, a_3\}$	$(0a_2a_30)$
5	$\{g_2, g_3, g_4\}$	–	a_3	$\{a_2, a_3, a_4\}$	$(0a_2a_3a_4)$
6	$\{g_2, g_4\}$	–	a_2	$\{a_2, a_4\}$	$(0a_20a_4)$
7	$\{g_4\}$	–	a_4	$\{a_4\}$	$(000a_4)$
8	$\emptyset$	–	–	$\emptyset$	(0000)

0 and the other corresponds to $a^* = 1$. For the convenience of graphical representation, in the code trellis T, we use the upper branch to represent $a^* = 0$ and the lower branch to represent $a^* = 1$. If a^* is a dummy information bit, then there is only one branch diverging from each state in $\Sigma_i(C)$. This single branch represents a dummy information bit. Using the above representation, we can easily extract the information bits from each path in the trellis (the dummy information bits are deleted).

Example 4.2 Consider the state labeling and trellis construction for the $(8, 4)$ RM code given in Example 3.1 whose TOGM G is repeated below,

$$G = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

For $0 \leq i \leq 8$, we determine the submatrix G_i^s and the state defining information set A_i^s as listed in Table 4.1. From A_i^s, we form the label for each state in $\Sigma_i(C)$ as shown in Table 4.1. The state transitions from time-i to time-$(i+1)$ are determined by the change from A_i^s to A_{i+1}^s. Following the trellis

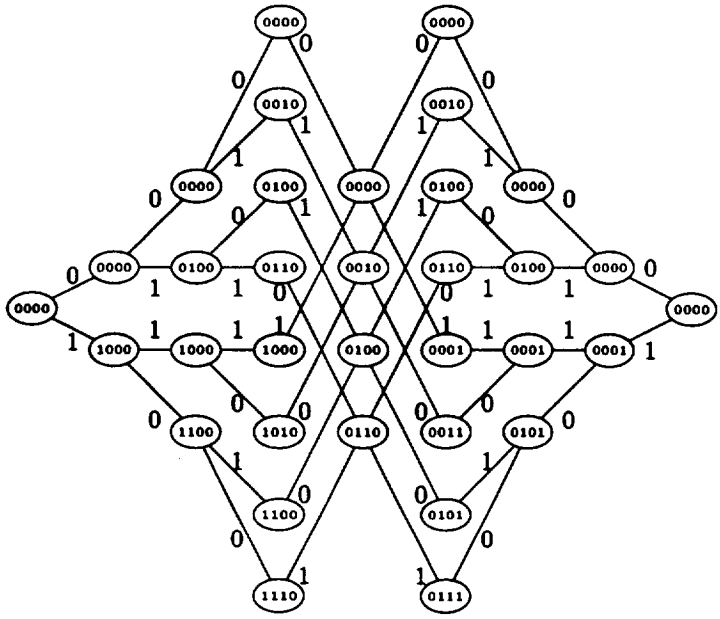

Figure 4.1. The 8-section trellis diagram for the $(8, 4)$ RM code with state labeling by the state defining information set.

construction procedure given above, we obtain the 8-section trellis diagram for the $(8, 4)$ RM code as shown in Figure 4.1. Each state in the trellis is labeled by a 4-tuple.

$\triangle\triangle$

In many cases, we do not need K bits for labeling the states of the N-section trellis for a binary (N, K) linear block code C. Let $(\rho_0, \rho_1, \ldots, \rho_N)$ be the state space dimension profile of the trellis. Define

$$\rho_{\max}(C) \triangleq \max_{0 \le i \le N} \rho_i \tag{4.1}$$

which is simply the maximum state space dimension of the trellis. From (3.7), we find that $\rho_{\max}(C) \le K$. In general, $\rho_{\max}$ is smaller than K. Since the number of states at any level of the trellis is less than or at most equal to

Table 4.2. State labeling for the $(8,4)$ RM code using $\rho_{\max}(C) = 3$ bits.

i	A_i^s	State Label
0	$\emptyset$	(000)
1	$\{a_1\}$	$(a_1 00)$
2	$\{a_1, a_2\}$	$(a_1 a_2 0)$
3	$\{a_1, a_2, a_3\}$	$(a_1 a_2 a_3)$
4	$\{a_2, a_3\}$	$(a_2 a_3 0)$
5	$\{a_2, a_3, a_4\}$	$(a_2 a_3 a_4)$
6	$\{a_2, a_4\}$	$(a_2 a_4 0)$
7	$\{a_4\}$	$(a_4 00)$
8	$\emptyset$	(000)

$2^{\rho_{\max}(C)}$, $\rho_{\max}(C)$ bits are sufficient for labeling the states in the trellis. Consider the state space $\Sigma_i(C)$ at time-i with $0 \leq i \leq N$ which is defined by the set $\{a_1^{(i)}, a_2^{(i)}, \ldots, a_{\rho_i}^{(i)}\}$ of ρ_i information bits. For each state $\sigma_i \in \Sigma_i(C)$, we form a $\rho_{\max}(C)$-tuple, denoted $l(\sigma_i)$, in which the first ρ_i components are simply $a_1^{(i)}, a_2^{(i)}, \ldots, a_{\rho_i}^{(i)}$ and the remaining $\rho_{\max}(C) - \rho_i$ components are set to 0, i.e.,

$$l(\sigma_i) \triangleq (a_1^{(i)}, a_2^{(i)}, \ldots, a_{\rho_i}^{(i)}, 0, 0, \ldots, 0). \tag{4.2}$$

Then $l(\sigma_i)$ is the label for the state σ_i.

Example 4.3 Again we consider the $(8,4)$ RM code given in Example 4.2. From the TOGM G of the code, we find the state space dimension profile of the 8-section trellis for the code to be $(0,1,2,3,2,3,2,1,0)$. Hence $\rho_{\max}(C) = 3$. Using 3 bits for labeling the states as described above, the state labels are given in Table 4.2. Compared to the state labeling given in Example 4.2, one bit is saved.

△△

4.2 STATE LABELING BY PARITY-CHECK MATRIX

Consider a binary (N, K) linear block code C with a parity-check matrix

$$H = [\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_j, \ldots, \boldsymbol{h}_N], \tag{4.3}$$

where, for $1 \leq j \leq N$, $\boldsymbol{h}_j$ denotes the j-th column of H and is a binary $(N-K)$-tuple. A binary N-tuple $\boldsymbol{c}$ is a codeword in C if and only if

$$\boldsymbol{c} \cdot H^T = \underbrace{(0, 0, \ldots, 0)}_{N-K}, \tag{4.4}$$

where H^T denotes the transpose of H. C is called the **null space** of H.

Let $\mathbf{0}_{N-K}$ denote the all-zero $(N-K)$-tuple $(0, 0, \ldots, 0)$. For $1 \leq i \leq N$, let H_i denote the submatrix that consists of the first i columns of H, i.e.,

$$H_i = [\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_i]. \tag{4.5}$$

It is clear that the rank of H_i is at most $N - K$, i.e.,

$$\text{Rank}(H_i) \leq N - K. \tag{4.6}$$

Then for each codeword $\boldsymbol{c} \in C^{\text{tr}}_{0,i}$,

$$\boldsymbol{c} \cdot H_i^T = \mathbf{0}_{N-K}. \tag{4.7}$$

$C^{\text{tr}}_{0,i}$ is the **null space** of H_i.

Now we consider the partition

$$p_{0,i}(C)/C^{\text{tr}}_{0,i}.$$

Let D be a coset in $p_{0,i}/C^{\text{tr}}_{0,i}$ and $D \neq C^{\text{tr}}_{0,i}$. For every vector $\boldsymbol{a} \in D$,

$$\boldsymbol{a} \cdot H_i^T = (s_1, s_2, \ldots, s_{N-K}) \neq \mathbf{0}_{N-K} \tag{4.8}$$

and is the same for all vectors in D, i.e., for $\boldsymbol{a}_1, \boldsymbol{a}_2 \in D$ and $\boldsymbol{a}_1 \neq \boldsymbol{a}_2$,

$$\boldsymbol{a}_1 \cdot H_i^T = \boldsymbol{a}_2 \cdot H_i^T = (s_1, s_2, \ldots, s_{N-K}). \tag{4.9}$$

The $(N-K)$-tuple $(s_1, s_2, \ldots, s_{N-K})$ is called the **label** for the coset D. Let D_1 and D_2 be two different cosets in $p_{0,i}(C)/C^{\text{tr}}_{0,i}$. Let $\boldsymbol{a}_1 \in D_1$ and $\boldsymbol{a}_2 \in D_2$. It follows from the theory of linear block codes that $\boldsymbol{a}_1 \neq \boldsymbol{a}_2$ and

$$\boldsymbol{a}_1 \cdot H_i^T \neq \boldsymbol{a}_2 \cdot H_i^T.$$

This says that different cosets in $p_{0,i}(C)/C_{0,i}^{tr}$ have different labels.

Recall the mathematical formulation of the state spaces of a code trellis. There is a one-to-one correspondence between a state σ in the state space $\Sigma_i(C)$ at time-i and a coset $D \in p_{0,i}(C)/C_{0,i}^{tr}$, and the codewords of $p_{0,i}(C)$ in D form the paths that connect the initial state σ_0 to state σ. This one-to-one correspondence leads to the definition of a state label.

Let $L(\sigma_0, \sigma)$ denote the set of paths in the code trellis for C that connect the initial state σ_0 to a state σ in the state space $\Sigma_i(C)$ at time-i.

Definition 4.1 For $0 \leq i \leq N$, the label of a state $\sigma \in \Sigma_i(C)$ based on a parity-check matrix H of C, denoted $l(\sigma)$, is defined as the binary $(N-K)$-tuple

$$l(\sigma) \triangleq a \cdot H_i^T = (s_1, s_2, \ldots, s_{N-K}), \tag{4.10}$$

for any $a \in L(\sigma_0, \sigma)$. For $i = 0, H_i = \emptyset$ and the initial state σ_0 is labeled with the all-zero $(N-K)$-tuple, 0_{N-K}. For $i = N, L(\sigma_0, \sigma_f) = C$ and the final state σ_f is also labeled with 0_{N-K}.

$\triangle\triangle$

It follows from the above definition of a state label, the one-to-one correspondence between the states in $\Sigma_i(C)$ and the cosets in $p_{0,i}(C)/C_{0,i}^{tr}$ for $0 \leq i \leq N$, and (4.10) that every state $\sigma \in \Sigma_i(C)$ has a **unique label** and different states have different labels.

For $0 \leq i < N$, let σ_i and σ_{i+1} be two adjacent states with $\sigma_i \in \Sigma_i(C)$ and $\sigma_{i+1} \in \Sigma_{i+1}(C)$. Let u_{i+1} be the label of the branch in the code trellis that connects state σ_i to state σ_{i+1}. The label u_{i+1} is simply the encoder output bit in the interval from time-i to time-$(i+1)$ and is given by (3.3) or (3.4). For every path $(u_1, u_2, \ldots, u_i) \in L(\sigma_0, \sigma_i)$, the path $(u_1, u_2, \ldots, u_i, u_{i+1})$ obtained by concatenating $(u_1, u_2, \ldots, u_i)$ with the branch u_{i+1} is a path that connects the initial state σ_0 to the state σ_{i+1} through the state σ_i. Hence, $(u_1, u_2, \ldots, u_i, u_{i+1}) \in L(\sigma_0, \sigma_{i+1})$. Then it follows from the definition of a state label that

$$\begin{aligned} l(\sigma_{i+1}) &= (u_1, u_2, \ldots, u_i, u_{i+1}) \cdot H_{i+1}^T \\ &= (u_1, u_2, \ldots, u_i) \cdot H_i^T + u_{i+1} \cdot h_{i+1}^T \\ &= l(\sigma_i) + u_{i+1} \cdot h_{i+1}^T. \end{aligned} \tag{4.11}$$

Eq.(4.11) simply says that given the starting state labeled $l(\sigma_i)$ at time-i and the output code bit u_{i+1} during the interval between time-i and time-$(i+1)$, the destination state labeled $l(\sigma_{i+1})$ at time-$(i+1)$ is uniquely determined.

Now we present a procedure for constructing the N-section trellis diagram for a binary (N, K) linear block code C by state labeling using the parity-check matrix of the code. Let $\boldsymbol{u} = (u_1, u_2, \ldots, u_N)$ be a binary N-tuple. For $0 < i \leq N$, let $p_{0,i}(\boldsymbol{u})$ denote the **prefix** of $\boldsymbol{u}$ that consists of the first i components, i.e.,

$$p_{0,i}(\boldsymbol{u}) = (u_1, u_2, \ldots, u_i). \tag{4.12}$$

Suppose that trellis has been completed up to the i-th section (or time-i). At this point, the rows of the TOGM G in the set $G_i^s = \{g_1^{(i)}, g_2^{(i)}, \ldots, g_{\rho_i}^{(i)}\}$ and their corresponding information bits $a_1^{(i)}, a_2^{(i)}, \ldots, a_{\rho_i}^{(i)}$ uniquely define a state $\sigma_i \in \Sigma_i(C)$. Let

$$\boldsymbol{u} = a_1^{(i)} \cdot g_1^{(i)} + a_2^{(i)} \cdot g_2^{(i)} + \cdots + a_{\rho_i}^{(i)} \cdot g_{\rho_i}^{(i)}.$$

Then $p_{0,i}(\boldsymbol{u})$ is a path connecting the initial state σ_0 to the state σ_i defined by $a_1^{(i)}, a_2^{(i)}, \ldots, a_{\rho_i}^{(i)}$. The label of state σ_i is given by

$$l(\sigma_i) = p_{0,i}(\boldsymbol{u}) \cdot H_i^T.$$

The construction of the $(i+1)$-section of the code trellis is accomplished by taking the following four steps:

(1) Identify the special row g^* (if any) in the submatrix G_i^f and its corresponding information bit a^*. Identify the special row g^0 (if any) in the submatrix G_i^s. Form the submatrix G_{i+1}^s by including g^* in G_i^s and excluding g^0 from G_i^s.

(2) Determine the set of information bits, $A_{i+1}^s = \{a_1^{(i+1)}, a_2^{(i+1)}, \ldots, a_{\rho_{i+1}}^{(i+1)}\}$, that correspond to the rows in G_{i+1}^s. Define and label the states in $\Sigma_{i+1}(C)$.

(3) For each state $\sigma_i \in \Sigma_i(C)$, form the next output code bit u_{i+1} from either (3.3) (if there is such a row g^* in G_i^f at time-i) or (3.4) (if there is no such row g^* in G_i^f at time-i).

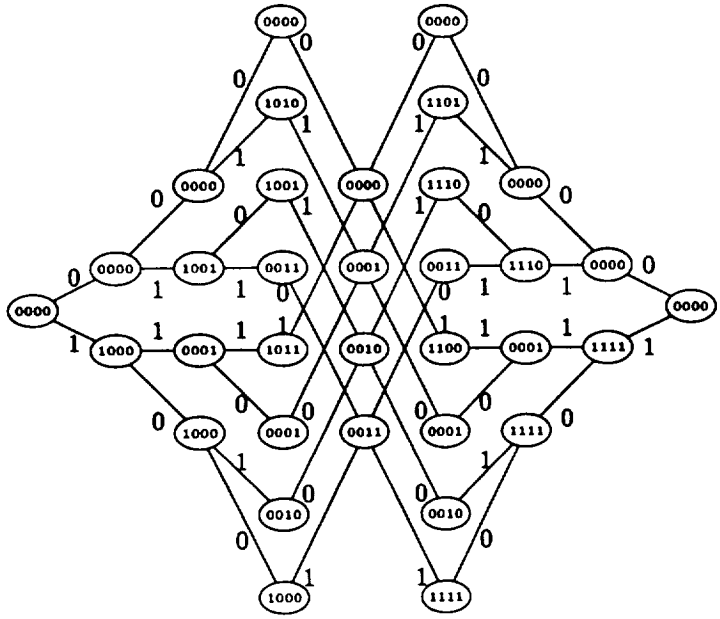

Figure 4.2. 8-section trellis for $(8, 4)$ RM code with state labeling by parity-check matrix.

(4) For each possible value of u_{i+1} (two if computed from (3.3) and one if computed from (3.4)), connect the state σ_i to the state $\sigma_{i+1} \in \Sigma_{i+1}(C)$ with label

$$l(\sigma_{i+1}) = l(\sigma_i) + u_{i+1} \cdot h_{i+1}^T.$$

The connecting branch, denoted $L(\sigma_i, \sigma_{i+1})$, is labeled with u_{i+1}. This completes the construction of the $(i+1)$-th section of the trellis.

Repeat the above steps until the entire code trellis is constructed.

Example 4.4 Consider the $(8, 4)$ RM code given in Example 3.1. This code is **self dual**. Therefore, a generator matrix is also a parity-check matrix. Suppose

we choose the parity-check matrix as follows:

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

Using this parity-check matrix for labeling and following the above trellis construction steps, we obtain the 8-section trellis with state labels shown in Figure 4.2. To illustrate the construction process, we assume that the trellis has been completed up to time-3. At this time instant, $G_3^s = \{g_1, g_2, g_3\}$ and $A_3^s = \{a_1, a_2, a_3\}$ are known. The eight states in $\Sigma_3(C)$ are defined by the eight combinations of a_1, a_2 and a_3. These 8 states and their labels are given below:

	states defined by (a_1, a_2, a_3)	state labels
$\sigma_3^{(0)}$	(000)	(0000)
$\sigma_3^{(1)}$	(001)	(1010)
$\sigma_3^{(2)}$	(010)	(1001)
$\sigma_3^{(3)}$	(011)	(0011)
$\sigma_3^{(4)}$	(100)	(1011)
$\sigma_3^{(5)}$	(101)	(0001)
$\sigma_3^{(6)}$	(110)	(0010)
$\sigma_3^{(7)}$	(111)	(1000)

Now we want to construct the 4-th section of the trellis up to time-4. At time-3, from the TOGM G, we find that $g^0 = g_1$ and there is no such row g^* with leading '1' at time-4. Therefore, $G_4^s = \{g_2, g_3\}$ and $A_4^s = \{a_2, a_3\}$. The four states in $\Sigma_4(C)$ at time-4 are defined by the four combinations of a_2 and a_3. The four codewords generated by the rows in G_4^s are:

(a_2, a_3)	paths
(0,0)	$u_0 = (00000000)$
(0,1)	$u_1 = (00111100)$
(1,0)	$u_2 = (01011010)$
(1,1)	$u_3 = (01100110)$

The four paths that connect the initial state σ_0 to the four states, denoted $\sigma_4^{(0)}$, $\sigma_4^{(1)}$, $\sigma_4^{(2)}$ and $\sigma_4^{(3)}$, in $\Sigma_4(C)$ are:

$$\begin{aligned} p_{0,4}(u_0) &= (0000), \\ p_{0,4}(u_1) &= (0011), \\ p_{0,4}(u_2) &= (0101), \\ p_{0,4}(u_3) &= (0110). \end{aligned}$$

The submatrix H_4 is

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

From $p_{0,4}(u_j)$, with $0 \leq j \leq 3$ and H_4, we can determine the labels for the four states, $\sigma_4^{(0)}, \sigma_4^{(1)}, \sigma_4^{(2)}$ and $\sigma_4^{(3)}$, in $\Sigma_4(C)$ which are given below:

	states defined by (a_2, a_3)	state labels
$\sigma_4^{(0)}$	(00)	(0000)
$\sigma_4^{(1)}$	(01)	(0001)
$\sigma_4^{(2)}$	(10)	(0010)
$\sigma_4^{(3)}$	(11)	(0011)

The four states and their labels are shown in Figure 4.3 at time-4. Now suppose the encoder is in the state $\sigma_3^{(5)}$ with label $l(\sigma_3^{(5)}) = (0001)$ at time-3. Since no such row g^* exists at $i = 3$, the output code bit u_4 is computed from (3.4) as follows:

$$\begin{aligned} u_4 &= 1 \cdot g_{14} + 0 \cdot g_{24} + 1 \cdot g_{34} \\ &= 1 \cdot 1 + 0 \cdot 1 + 1 \cdot 1 \\ &= 0. \end{aligned}$$

Then the state $\sigma_3^{(5)}$ is connected to the state in $\Sigma_4(C)$ with label

$$\begin{aligned} l(\sigma_3^{(5)}) + u_4 \cdot h_4^T &= (0001) + 0 \cdot (1011) \\ &= (0001), \end{aligned}$$

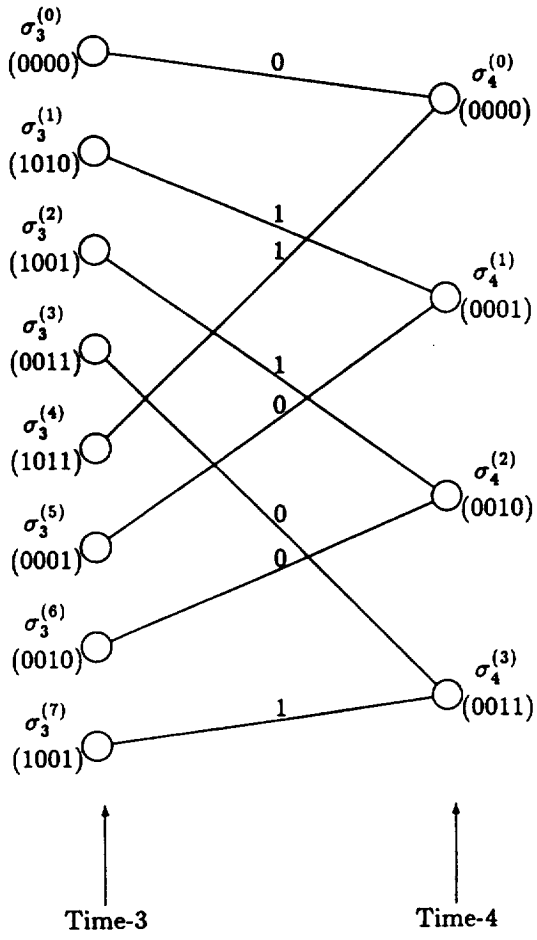

Figure 4.3. State labels at the two ends of the 4-th section of the trellis for $(8, 4)$ RM code.

which is state $\sigma_4^{(1)}$. The connecting branch is labeled with $u_4 = 0$. The connections from the other states in $\Sigma_3(C)$ to the states in $\Sigma_4(C)$ are accomplished in the same manner.

$\triangle\triangle$

State labeling based on the state defining information sets requires K (or $\rho_{\max}(C)$) bits to label each state of the trellis; however, state labeling based on

the parity-check matrix requires $N-K$ bits to label each state of the trellis. Therefore, labeling method-1 is more efficient for codes with $K < N-K$ while labeling method-2 is more efficient for codes with $K > N-K$.

4.3 STRUCTURAL SYMMETRY

Consider a binary (N, K) linear block code C with even length N and TOGM

$$G = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_K \end{bmatrix} = \begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1N} \\ g_{21} & g_{22} & \cdots & g_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ g_{K1} & g_{K2} & \cdots & g_{KN} \end{bmatrix}.$$

Let T denote the N-section trellis diagram for C. Suppose the TOGM G has the following symmetry property: For each row g in G with $\text{span}(g) = [a, b]$, there exists a row g' in G with $\text{span}(g') = [N+1-b, N+1-a]$. With this symmetry property in G, we can readily see that for $0 \leq i < N/2$, the number of rows in G that are active at time-$(N-i)$ is equal to the number of rows in G that are active at time-i. This implies that

$$|\Sigma_{N-i}(C)| = |\Sigma_i(C)|$$

for $0 \leq i < N/2$. We can permute the rows of G such that the resultant matrix, denoted G', is in a **reverse trellis oriented form**:

(1) The trailing '1' of each row appears in a column before the trailing '1' of any row below it.

(2) No two rows have their leading "ones" in the same column.

If we rotate the matrix G' by 180° counter clockwisely, we obtain a matrix G'' in which the i-th row g''_i is simply the $(K+1-i)$-th row g'_{K+1-i} of G' in reverse order (the trailing '1' of g'_{K+1-i} becomes the leading '1' of g''_i and the leading '1' of g'_{K+1-i} becomes the trailing '1' of g''_i). From the above, we see that G'' and G are structurally identical in the sense that

$$\text{span}(g''_i) = \text{span}(g_i)$$

for $1 \leq i \leq K$. Consequently, the N-section trellis T for C has the following **mirror symmetry** [101]: The last $N/2$ sections of T form the mirror image of the first $N/2$ sections of T (not including the path labels).

Example 4.5 Consider the $(8,4)$ RM code given in Example 4.2 with TOGM

$$G = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

We find that $\text{span}(g_1) = [1,4]$, $\text{span}(g_4) = [5,8]$, and g_1 and g_4 are symmetrical with each other. Row g_2 has span $[2,7]$ and is symmetrical with itself. Row g_3 has span $[3,6]$ and is also symmetrical with itself. Suppose we permute the second and third rows of G. We obtain the following matrix in reverse trellis oriented form:

$$G' = \begin{bmatrix} g'_1 \\ g'_2 \\ g'_3 \\ g'_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Rotating G' 180° counter clockwisely, we obtain the following matrix:

$$\begin{bmatrix} g''_1 \\ g''_2 \\ g''_3 \\ g''_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

We find that G'' and G are in fact identical, not just structurally identical. Therefore, the 8-section trellis T for the $(8,4)$ RM code has mirror symmetry with respect to the boundary location 4, the last four sections form the mirror image of the first four sections as shown in Figures 3.2 and 4.1.

△△

For the case that N is odd, if the TOGM G of a binary (N,K) code C has the mirror symmetry property, then the last $(N-1)/2$ sections of the N-section trellis T for C form the mirror image of the first $(N-1)/2$ sections of T.

For the case that $G'' = G$, the N-section trellis T of C has **full mirror symmetry** structure [101]. For N even, the last $N/2$ sections of T in **reverse direction** (the final state σ_f is being regarded as the initial state) is completely identical to the first $N/2$ sections of T (including the path labels). The 8-section trellis of the $(8, 4)$ RM code has full mirror symmetry as shown in Figure 4.1. For N odd, the last $(N-1)/2$ sections of T in reverse direction are completely identical to the first $(N-1)/2$ sections of T (including the path labels).

5 TRELLIS COMPLEXITY

This chapter is devoted to analyzing the complexity of an N-section trellis diagram for an (N, K) linear block code. Trellis complexity is, in general, measured in terms of the state and branch complexities. These two complexities determine the storage and computation requirements of a trellis-based decoding algorithm, such as the Viterbi decoding algorithm. The state complexity of a trellis is measured by its state space dimension profile and the branch complexity is measured by the total number of branches (or edges) in the trellis. In Section 5.1, a simple upper bound on the maximum state space dimension is derived. It is proved that the state complexity of a linear block code is the same as that of its dual code. In Section 5.2, the concepts of a minimal trellis diagram and optimum bit permutation in terms of state complexity are introduced. It is proved that the trellis construction based on a TOGM results in a minimal trellis. In Section 5.3, the branch complexity of an N-section trellis diagram is analyzed. Finally, in Section 5.4, the general structure of N-section

trellis diagrams for cyclic codes is given. It it shown that the maximum state space dimension meets the upper bound derived in Section 5.1.

5.1 STATE COMPLEXITY

For a binary (N, K) linear block code C, the state complexity of an N-section bit-level code trellis is measured by its state space dimension profile

$$(\rho_0, \rho_1, \rho_2, \ldots, \rho_N),$$

where for $0 \leq i \leq N$,

$$\rho_i = \log_2 |\Sigma_i(C)|.$$

Let $\rho_{\max}(C)$ denote the maximum among the state space dimensions, i.e.,

$$\rho_{\max}(C) = \max_{0 \leq i \leq N} \rho_i.$$

Using the construction method described in Chapter 3, the state space dimension at time-i is given by (3.7),

$$\rho_i = K - k(C_{0,i}) - k(C_{i,N}),$$

for $0 \leq i \leq N$. Since $k(C_{0,i})$ and $k(C_{i,N})$ are nonnegative, we have

$$\rho_{\max}(C) \leq K. \tag{5.1}$$

However, it follows from (4.6) and the definition and uniqueness of a state label at any time-i (see (4.10)) that

$$|\Sigma_i(C)| \leq 2^{N-K}$$

and

$$\rho_i \leq N - K \tag{5.2}$$

for $0 \leq i \leq N$. Eq.(5.2) implies that

$$\rho_{\max}(C) \leq N - K. \tag{5.3}$$

Combining (5.1) and (5.3), we have the following upper bound on the maximum state complexity:

$$\rho_{\max}(C) \leq \min\{K, N - K\}. \tag{5.4}$$

This bound was first proved by Wolf [109]. In general, this bound is quite loose. However, for cyclic (or shortened cyclic) codes, this bound gives the exact state complexity. For noncyclic codes, tighter upper bounds on $\rho_{\max}(C)$ have been obtained.

If the Viterbi algorithm is applied to the N-section trellis of a code, then the maximum numbers of survivors and path metrics needed to be stored are both $2^{\rho_{\max}(C)}$. Therefore, the parameter $\rho_{\max}(C)$ is a key measure of the decoding complexity (or trellis complexity).

For $0 \leq i \leq \min\{K, N-K\}$, it follows from the structure of a TOGM G that the number of rows in G whose active spans contain the time index i is no greater than i. For $i \geq \max\{K, N-K\}$, since there is one-to-one correspondence between the states in $\Sigma_i(C)$ and cosets in the partition $p_{i,N}(C)/C^{\rm tr}_{i,N}$,

$$\begin{aligned} \rho_i &= k(p_{i,N}(C)) - k(C^{\rm tr}_{i,N}) \\ &\leq k(p_{i,N}(C)) \\ &\leq N - i \\ &\leq \min\{K, N-K\}. \end{aligned} \tag{5.5}$$

Therefore, for $0 \leq i \leq N$, we have the following upper bound on ρ_i:

$$\rho_i \leq \min\{i, K, N-K, N-i\}. \tag{5.6}$$

Let $C^\perp$ denote the dual code of C. Then $C^\perp$ is an $(N, N-K)$ linear block code. Consider the N-section trellis diagram for $C^\perp$. For $0 \leq i \leq N$, let $\Sigma_i(C^\perp)$ denote the state space of $C^\perp$ at time-i. Then there is a one-to-one correspondence between the states in $\Sigma_i(C^\perp)$ and the cosets in the partition $p_{0,i}(C^\perp)/C^{\perp,{\rm tr}}_{0,i}$ where $C^{\perp,{\rm tr}}_{0,i}$ denotes the truncation of $C^\perp_{0,i}$ in the interval $[1, i]$. Therefore, the dimension of $\Sigma_i(C^\perp)$ is given by

$$\rho_i(C^\perp) = k(p_{0,i}(C^\perp)) - k(C^{\perp,{\rm tr}}_{0,i}). \tag{5.7}$$

Note that $p_{0,i}(C^\perp)$ is the dual code of $C^{\rm tr}_{0,i}$ and $C^{\perp,{\rm tr}}_{0,i}$ is the dual code of $p_{0,i}(C)$. Therefore,

$$\begin{aligned} k(p_{0,i}(C^\perp)) &= i - k(C^{\rm tr}_{0,i}) \\ &= i - k(C_{0,i}) \end{aligned} \tag{5.8}$$

and

$$k(C_{0,i}^{\perp,\mathrm{tr}}) = i - k(p_{0,i}(C)). \tag{5.9}$$

It follows from (5.7), (5.8) and (5.9) that

$$\rho_i(C^{\perp}) = K - k(C_{0,i}) - k(C_{i,N}). \tag{5.10}$$

From (3.7) and (5.10), we find that for $0 \leq i \leq N$,

$$\rho_i(C^{\perp}) = \rho_i(C). \tag{5.11}$$

This says that C and its dual code $C^{\perp}$ have the same state complexity.

5.2 MINIMAL TRELLISES

An N-section trellis is said to be **minimal** if the total number of states in the trellis is minimum. A minimal trellis is **unique** within isomorphism [77], i.e., two minimal trellises for the same code are isomorphic (**structurally identical**). The above definition of minimality is commonly used in the literature. However, a more meaningful and useful definition of minimality of a trellis is in terms of its state space dimension profile. An N-section trellis is said to be a **minimum state space dimension trellis** if the state space dimension at each time of the trellis is minimum. A more precise definition is given as follows. Let T be an N-section trellis for an (N, K) code C with state space dimension profile $(\rho_0, \rho_1, \ldots, \rho_N)$. T is said to be minimal if, for any other N-section trellis T' for C with state space dimension profile $(\rho'_0, \rho'_1, \ldots, \rho'_N)$, the following inequality holds:

$$\rho_i \leq \rho'_i,$$

for $0 \leq i \leq N$.

Suppose a minimum state space dimension trellis T exists. Then, it is clear that T is a minimal trellis in total number of states. The formulation of state spaces given in Section 3.4 results in a minimum state space dimension trellis (or minimal trellis) for an (N, K) linear block code. This will be proved in Theorem 5.1. This says that a minimum state space dimension trellis T exists for any linear block code C. From the uniqueness of a minimal trellis in total number of states within graph isomorphism, the minimal trellis is a

minimum state space dimension trellis. This gives the equivalence between the two definitions of minimality of a trellis for a linear block code.

Theorem 5.1 Let C be a binary (N, K) linear block code with trellis oriented generator matrix G. The N-section trellis T for C constructed based on G is a minimum state space dimension trellis.

Proof: We only need to prove that for $1 \leq i < N$, the number of states, 2^{ρ_i}, at time-i in the trellis T is minimum over all the trellises for C, where

$$\rho_i = K - k(C_{0,i}) - k(C_{i,N}).$$

Let C_i^s denote the linear subcode of C that is spanned by the rows in the submatrix G_i^s of G. Then $|C_i^s| = 2^{\rho_i}$. For two different codewords $\boldsymbol{u}$ and $\boldsymbol{v}$ in C_i^s, it follows from condition (1) of a TOGM that

$$p_{0,i}(\boldsymbol{u}) \neq p_{0,i}(\boldsymbol{v}).$$

This implies that

$$|\{p_{0,i}(\boldsymbol{u}) : \boldsymbol{u} \in C_i^s\}| = 2^{\rho_i}.$$

Suppose there is a trellis T' for C whose number of states at time-i is less than 2^{ρ_i}. Then, there must be two different codewords $\boldsymbol{u}$ and $\boldsymbol{v}$ in C_i^s such that: (1) there are two paths connecting the initial state to a state σ at time-i in T' whose label sequences are $p_{0,i}(\boldsymbol{u})$ and $p_{0,i}(\boldsymbol{v})$, respectively; and (2) there is a path connecting the state σ to the final state in T' whose label sequence is $p_{i,N}(\boldsymbol{u})$. Without loss of generality, we assume $\boldsymbol{u} \neq \boldsymbol{0}$.

Let $\boldsymbol{u}'$ denote the binary N-tuple such that $p_{0,i}(\boldsymbol{u}') = p_{0,i}(\boldsymbol{v})$ and $p_{i,N}(\boldsymbol{u}') = p_{i,N}(\boldsymbol{u})$. Since $\boldsymbol{u}'$ is a path in T' connecting the initial state to the final state, it follows from condition (4) of Definition 3.1 of an N-section trellis for a linear block code that $\boldsymbol{u}'$ is a codeword in C. Therefore, $\boldsymbol{u} + \boldsymbol{u}' \in C$. Note that $p_{0,i}(\boldsymbol{u} + \boldsymbol{u}') = p_{0,i}(\boldsymbol{u}) + p_{0,i}(\boldsymbol{v}) \neq \boldsymbol{0}$ and $p_{i,N}(\boldsymbol{u} + \boldsymbol{u}') = p_{i,N}(\boldsymbol{u}) + p_{i,N}(\boldsymbol{u}') = \boldsymbol{0}$. This implies that $\boldsymbol{u} + \boldsymbol{u}' \in C_{0,i}$. There are three cases to be considered:

(1) Suppose $\boldsymbol{u}' \in C_{0,i}$. This implies that $\boldsymbol{u} \in C_{0,i}$ which is a contradiction to the hypothesis that $\boldsymbol{u} \in C_i^s$ and $\boldsymbol{u} \neq \boldsymbol{0}$.

(2) Suppose $\boldsymbol{u}' \in C_i^s$. This implies that $\boldsymbol{u} + \boldsymbol{u}' \in C_i^s$. Since $\boldsymbol{u} + \boldsymbol{u}' \neq \boldsymbol{0}$, $\boldsymbol{u} + \boldsymbol{u}'$ can not be in both C_i^s and $C_{0,i}$. This results in a contradiction.

(3) Suppose $\boldsymbol{u}' \in C_{i,N}$. This implies that $\boldsymbol{u} \in C_{0,i} \oplus C_{i,N}$, which is not possible.

Therefore, $\boldsymbol{u} + \boldsymbol{u}'$ can not be a codeword in C. This results in a contradiction to our earlier hypothesis that there exists an N-section trellis T' for C whose number of states at time-i is less than 2^{ρ_i}. Therefore, the hypothesis is invalid and 2^{ρ_i} gives the minimum number of states at time-i for $1 \leq i < N$.

$\triangle\triangle$

It follows from Theorem 5.1 that Eq.(3.7) gives the minimum state space dimension ρ_i with $0 \leq i \leq N$ for an N-section trellis for an (N, K) linear block code. From (3.7), we see that the state space dimension ρ_i at time-i depends on the dimensions of the past and future codes, $C_{0,i}$ and $C_{i,N}$. For a given code C, $k(C_{0,i})$ and $k(C_{i,N})$ are fixed.

Given an (N, K) linear block code C, a permutation of the orders of the bit (or symbol) positions results in an equivalent code C' with the same weight distribution. Different permutations of the bit positions may result in different dimensions, $k(C_{0,i})$ and $k(C_{i,N})$, of the past and future subcodes, $C_{0,i}$ and $C_{i,N}$, and hence different state space dimensions ρ_i at time-i. A permutation that yields the smallest state space dimension at every time of the code trellis is called an **optimum** permutation (or bit ordering). It is clear that an optimum permutation reduces the state complexity and is often desirable. Optimum permutation is **hard to find**, however optimum permutations for RM codes are known [45] but they are unknown for other classes of codes.

5.3 BRANCH COMPLEXITY

The branch complexity of an N-section trellis diagram for an (N, K) linear block code C is defined as the total number of branches in the trellis. This complexity determines the number of additions required in a trellis-based decoding algorithm to decode a received sequence.

Consider the N-section trellis diagram T for C which is constructed based on the rules and procedures described in Chapters 3 and 4. Recall that at time-i with $0 \leq i < N$, there are two branches diverging from a state in $\Sigma_i(C)$ if there exists a row g^* in G_i^f; and there is only one branch diverging from a

state in $\Sigma_i(C)$ if there exists no such row g^* in G_i^f. Define

$$I_i(g^*) \triangleq \begin{cases} 1, & \text{if } g^* \notin G_i^f, \\ 2, & \text{if } g^* \in G_i^f. \end{cases} \tag{5.12}$$

Let E denote the total number of branches in the N-section trellis T. Then

$$\begin{aligned} E &= \sum_{i=0}^{N-1} |\Sigma_i(C)| \cdot I_i(g^*) \\ &= \sum_{i=0}^{N-1} 2^{\rho_i} \cdot I_i(g^*). \end{aligned} \tag{5.13}$$

Example 5.1 Again we consider the $(8,4)$ linear block code given in Example 3.1. From Table 4.1, we find that

$$I_0(g^*) = I_1(g^*) = I_2(g^*) = I_4(g^*) = 2$$

and

$$I_3(g^*) = I_5(g^*) = I_6(g^*) = I_7(g^*) = 1.$$

The state space dimension profile of the 8-section trellis for the code is (0, 1, 2, 3, 2, 3, 2, 1, 0). From (5.13), we have

$$\begin{aligned} E &= 2^0 \cdot 2 + 2^1 \cdot 2 + 2^2 \cdot 2 + 2^3 \cdot 1 + 2^2 \cdot 2 + 2^3 \cdot 1 + 2^2 \cdot 1 + 2^1 \cdot 1 \\ &= 2 + 4 + 8 + 8 + 8 + 8 + 4 + 2 \\ &= 44. \end{aligned}$$

△△

An N-section trellis diagram for an (N,K) linear block code is said to be a **minimal branch (or edge) trellis diagram** if it has the smallest branch complexity. A minimal trellis diagram has the smallest branch complexity [69]. Branch complexity also depends on the bit ordering of a code. Proper permutation of the bit positions of a code may result in a significant reduction in branch complexity. A permutation that results in minimal branch complexity is called an optimum permutation. From (5.13), we can readily see that a permutation which minimizes each product in the summation of (5.13) is a minimal edge trellis diagram. A good permutation in terms of branch complexity should have the following property: for $0 \leq i < N$, when ρ_i is large, $I_i(g^*)$ should be equal to 1.

5.4 TRELLIS STRUCTURE OF CYCLIC CODES

Consider an (N, K) cyclic code C over GF(2) generated by the following polynomial [62],

$$g(X) = 1 + g_1 X + g_2 X^2 + \cdots + g_{N-K-1} X^{N-K-1} + X^{N-K},$$

where for $1 \leq i < N-K$, $g_i \in \mathrm{GF}(2)$. A generator matrix for this cyclic code is given by

$$G = \begin{bmatrix} 1 & g_1 & g_2 & \cdots & \cdots & g_{N-K-1} & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & g_1 & g_2 & \cdots & \cdots & g_{N-K-1} & 1 & 0 & \cdots & 0 \\ & & & & & \ddots & & & & & \\ 0 & 0 & \cdots & 0 & 1 & g_1 & g_2 & \cdots & g_{N-K-1} & 1 & \end{bmatrix}. \tag{5.14}$$

The K rows of G are simply the K cyclic shifts of the first row. This generator matrix has the following properties:

(1) It is in trellis oriented form.

(2) For $1 \leq i \leq K$, the span of the i-th row g_i is

$$\mathrm{span}(g_i) = [i, N-K+i].$$

(3) The active spans of all the rows have the same length, $N-K$.

Now we consider the bit-level trellis structure for this (N, K) cyclic code. There are two cases to be considered: $K > N-K$ and $K \leq N-K$. Consider the case for which $K > N-K$. For $1 \leq i \leq N-K$, the number of rows whose active spans contain the time index i is i. These rows are simply the first i rows. For $N-K < i \leq K$, the number of rows whose active spans contain the time index i is $N-K$. For $K < i \leq N$, the number of rows whose active spans contain the time index-i is $N-i$. Since $i > K$,

$$N-i < N-K.$$

From the above analysis, we see that the maximum state space dimension is $\rho_{\max}(C) = N-K$ and the state space profile is

$$(0, 1, \ldots, N-K-1, N-K, \ldots, N-K, N-K-1, \ldots, 1, 0).$$

Now consider the second case for which $K \leq N-K$. For $1 \leq i \leq K$, the number of rows whose active spans contain the time index i is i (the first i rows). For $K \leq i \leq N-K$, the number of rows whose active spans contain the time index i is K. For $N-K < i \leq N$, the number of rows whose active spans contain i is $N-i$. Since $i > N-K$, $N-i < K$. From the above analysis, we find that the maximum state space dimension is

$$\rho_{\max}(C) = K,$$

and the state space dimension profile is

$$(0, 1, \ldots, K-1, K, \ldots, K, K-1, \ldots, 1, 0).$$

Putting the results of the above two cases together, we conclude that for an (N, K) cyclic code, the maximum state space dimension is

$$\rho_{\max}(C) = \min\{K, N-K\}.$$

This is to say that a code in cyclic form has the worst state complexity (i.e., it meets the upper bound on the state complexity).

The generator polynomial $g(X)$ of an (N, K) binary cyclic code C divides $X^N + 1$ [62]. Let

$$X^N + 1 = g(X)h(X).$$

Then $h(X)$ is a polynomial of degree K of the following form:

$$h(X) = 1 + h_1 X + h_2 X^2 + \cdots + h_{K-1} X^{K-1} + X^K$$

with $h_i \in \mathrm{GF}(2)$ for $1 \leq i < K$. This polynomial is called the parity-check polynomial. The dual code $C^\perp$ of C is an $(N, N-K)$ cyclic code with generator polynomial

$$X^K h(X^{-1}) = 1 + h_{K-1} X + \cdots + h_1 X^{K-1} + X^K.$$

The generator matrix for the dual code $C^\perp$ is

$$H = \begin{bmatrix} 1 & h_{K-1} & h_{K-2} & \cdots & \cdots & \cdots & \cdots & 1 & 0 & \cdots & 0 \\ 0 & 1 & h_{K-1} & h_{K-2} & \cdots & \cdots & \cdots & \cdots & 1 & 0 \cdots & 0 \\ 0 & 0 & 1 & h_{K-1} & h_{K-2} & \cdots & \cdots & \cdots & \cdots & 1 \cdots & 0 \\ \vdots & & & & & & & & & & \vdots \\ 0 & 0 & \cdots & \cdots & \cdots & 0 & 1 & h_{K-1} & h_{K-2} & \cdots & 1 \end{bmatrix} \tag{5.15}$$

which is in trellis oriented form. The trellis structure for $C^{\perp}$ can be analyzed in the same manner as for C.

The trellis of a cyclic code has mirror symmetry, i.e., the right-half and left-half of the trellis with respect to the center are structurally identical.

To reduce the state complexity of a cyclic code, a permutation of the bit position is needed [45, 46].

The branch complexity of the N-section trellis diagram T constructed based on the TOGM G given by (5.14) can be evaluated easily. First we note that

$$I_i(g^*) = \begin{cases} 2, & \text{for } 0 \le i < K, \\ 1, & \text{otherwise.} \end{cases}$$

Suppose $K < N - K$. Then branch complexity of the trellis T is

$$\begin{aligned} E &= \sum_{i=0}^{K-1} 2^i \cdot 2 + (N-2K) \cdot 2^K + \sum_{i=0}^{K-1} 2^{K-i} \cdot 1 \\ &= 2 \cdot (2^K - 1) + (N-2K) \cdot 2^K + 2 \cdot (2^K - 1) \\ &= (N - 2K + 4) \cdot 2^K - 4. \end{aligned} \tag{5.16}$$

For $K \ge N - K$, we have

$$\begin{aligned} E &= \sum_{i=0}^{N-K-1} 2^i \cdot 2 + 2 \cdot (2K - N) \cdot 2^{N-K} + \sum_{i=0}^{N-K-1} 2^{N-K-i} \\ &= (4K - 2N + 4) \cdot 2^{N-K} - 4. \end{aligned} \tag{5.17}$$

5.5 TRELLISES FOR NONBINARY LINEAR BLOCK CODES

The methods for constructing trellises for binary linear block codes can be generalized for constructing trellises for nonbinary linear block codes with symbols from GF(q) in a straightforward manner. The symbol-level N-section trellis diagram for an (N, K) linear block code C over GF(q) has the following basic properties: (1) every branch is labeled with a code symbol from GF(q); (2) except for the initial state, every state has at least one, but no more than q, incoming branches; (3) except for the final state, every state has at least one, but no more than q, outgoing branches; and (4) the initial state has no incoming branch and the final state has no outgoing branch. In the definition of

a trellis oriented generator matrix, the leading "1" and trailing "1" of a row are replaced by leading and trailing "nonzero components", respectively. The maximum state space dimension $\rho_{\max}(C)$ of the minimal N-section trellis for C is upper bounded by

$$\rho_{\max}(C) \leq \min\{K, N-K\},$$

and the maximum number of states, $|\Sigma(C)|_{\max}$, is upper bounded by

$$|\Sigma(C)|_{\max} \leq q^{\min\{K,N-K\}}.$$

For Reed-Solomon (RS) codes over GF(q), the above equalities hold, i.e.,

$$\rho_{\max}(C) = \min\{K, N-K\},$$

and

$$|\Sigma(C)|_{\max} = q^{\min\{K,N-K\}}.$$

6 TRELLIS SECTIONALIZATION

So far, we have only considered bit-level N-section trellis diagrams for linear block codes of length N. In a bit-level trellis diagram, every time instant in the encoding interval $\Gamma = \{0, 1, 2, \ldots, N\}$ is a section boundary location and every branch represents a code bit. It is possible to sectionalize a bit-level trellis with section boundary locations at selected instants in the encoding interval Γ. This sectionalization results in a trellis in which a branch may represent multiple code bits and two adjacent states may be connected by multiple branches. Proper sectionalization may result in useful trellis structural properties and allow us to devise efficient trellis-based decoding algorithms. This chapter is devoted in analyzing sectionalized trellis diagrams for linear block codes. Section 6.1 presents the concepts and rules for trellis sectionalization. In Section 6.2, the branch complexity and state connectivity are analyzed and expressed in terms of the dimensions of codes related to the code being considered. In Section 6.3, construction of a sectionalized trellis diagram for a

linear block code based on the trellis oriented generator matrix is presented. Section 6.4 studies the parallel structure of a sectionalized trellis diagram.

6.1 SECTIONALIZATION OF A CODE TRELLIS

For a positive integer $L \leq N$, let

$$U \triangleq \{h_0, h_1, h_2, \ldots, h_L\} \tag{6.1}$$

be a subset of $L+1$ time instants in the encoding interval $\Gamma = \{0, 1, 2, \ldots, N\}$ for an (N, K) linear block code C with $0 = h_0 < h_1 < h_2 < \cdots < h_L = N$. An L-section trellis diagram for C with **section boundaries** at the locations (time instants) in U, denoted $T(U)$, can be obtained from the N-section trellis T by: (1) deleting every state in $\Sigma_h(C)$ for $h \in \{0, 1, \ldots, N\} \setminus U$ and every branch to or from a deleted state, and (2) for $1 \leq j \leq L$, connecting a state $\sigma \in \Sigma_{h_{j-1}}$ to a state $\sigma' \in \Sigma_{h_j}$ by a branch with label α if and only if there is a path with label α from state σ to state σ' in the N-section trellis T. In an L-section trellis with boundary locations in $U = \{h_0, h_1, \ldots, h_L\}$, a branch from a state in $\Sigma_{h_{j-1}}(C)$ to a state in $\Sigma_{h_j}(C)$ represents $(h_j - h_{j-1})$ code symbols.

A **subgraph** of a trellis diagram is called a **subtrellis**. The subtrellis of $T(U)$ which consists of the state space $\Sigma_{h_{j-1}}(C)$ at time-h_{j-1}, state space $\Sigma_{h_j}(C)$ at time-h_j, and all the branches between the states in $\Sigma_{h_{j-1}}(C)$ and $\Sigma_{h_j}(C)$, is called the j-th **section** of $T(U)$. The length of the j-th section is $h_j - h_{j-1}$. If the lengths of all the sections of an L-section code trellis $T(U)$ are the same, $T(U)$ is said to be **uniformly sectionalized**. In an L-section trellis diagram with $L < N$, two adjacent states may be connected by multiple branches (called **parallel branches**) with different labels.

Let $\rho_{h_j} \triangleq \log_2 |\Sigma_{h_j}(C)|$ be the dimension of the state space $\Sigma_{h_j}(C)$ at time-h_j. Then

$$(\rho_0, \rho_{h_1}, \rho_{h_2}, \cdots, \rho_{h_{L-1}}, \rho_N)$$

is the state space dimension profile of the L-section code trellis $T(U)$ with section boundary set $U = \{0, h_1, h_2, \ldots, h_{L-1}, N\}$. From (3.7), we have

$$\rho_{h_j} = K - k(C_{0,h_j}) - k(C_{h_j,N}) \,. \tag{6.2}$$

If we choose the section boundaries, $U = \{h_0, h_1, \ldots, h_L\}$, at the places where $\rho_{h_1}, \rho_{h_2}, \ldots, \rho_{h_{L-1}}$ are small, then the resultant L-section code trellis $T(U)$ has

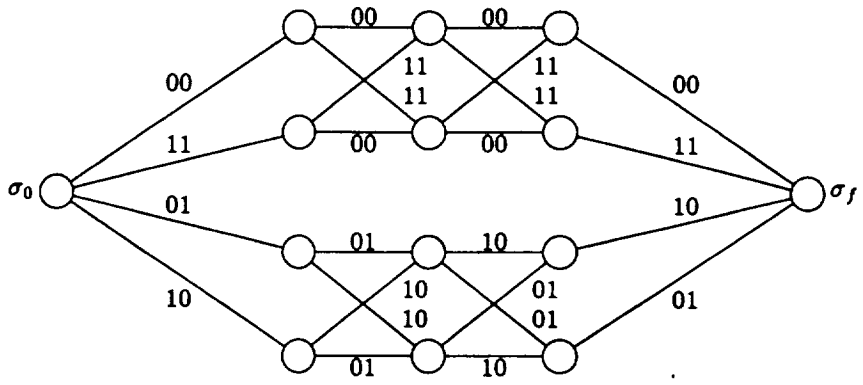

Figure 6.1. A 4-section trellis for the $(8, 4)$ RM code.

a small state space dimension profile. The maximum state space dimension is

$$\rho_{L,\max}(C) \triangleq \max_{0 \le j \le L} \rho_{h_j}. \tag{6.3}$$

In implementing a trellis-based decoder, such as a Viterbi decoder, a proper choice of the section boundary locations results in a significant reduction in decoding complexity.

Example 6.1 Again, we consider the $(8, 4)$ RM code given in Example 3.1 whose 8-section trellis diagram is shown in Figure 3.2 (or Figure 4.1). Suppose we choose $L = 4$ and the section boundary set $U = \{0, 2, 4, 6, 8\}$. Following the above rules of sectionalization of a code trellis, we obtain a uniform 4-section trellis diagram as shown in Figure 6.1, in which every branch represents 2 code bits. The state space dimension profile for this 4-section trellis is $(0, 2, 2, 2, 0)$ and the maximum state space dimension is $\rho_{4,\max}(C) = 2$. It is a 4-section, 4-state code trellis. From Figure 6.1, we notice that the right-half of the trellis (the third and fourth sections) is the **mirror image** of the left-half of the trellis (the first and second sections). This **mirror symmetry** allows **bidirectional** decoding. Furthermore, the code trellis consists of two **parallel** and **structurally** identical **(isomorphic)** subtrellises without cross connections between them. This **parallel structure** allows us to devise two identical 2-state (Viterbi) decoders to process the trellis in parallel. The mirror

symmetry and parallel structure not only simplify the decoding complexity but also speed up the decoding process. For large code trellises, these structural properties are very important in IC (integrated circuit) implementations.

△△

An L-section trellis diagram obtained from a minimal N-section trellis diagram by deleting states and branches at places other than the section boundary locations is minimal, i.e., it is a minimal L-section trellis diagram for a given section boundary set U.

6.2 BRANCH COMPLEXITY AND STATE CONNECTIVITY

Consider the j-th section of a minimal L-section trellis diagram $T(U)$ with section boundary set $U = \{h_0, h_1, \ldots, h_L\}$ for an (N, K) linear code C. The boundaries of this section are h_{j-1} and h_j. Each branch in this section is labeled with $h_j - h_{j-1}$ bits. Let σ and σ' be two adjacent states in the state spaces $\Sigma_{h_{j-1}}(C)$ and $\Sigma_{h_j}(C)$, respectively. Let $L(\sigma, \sigma')$ denote the set of parallel branches connecting σ and σ'. Let $L(\sigma_0, \sigma)$ denote the set of paths connecting the initial state σ_0 to the state σ. Sometimes, it is convenient to regard the parallel branches, $L(\sigma, \sigma')$, between two states as a single branch. This single branch is called a **composite branch**, and $L(\sigma, \sigma')$ is called a **composite branch label**.

The branch complexity of a trellis section is measured by: (1) the size of a composite branch; (2) the number of distinct composite branches in the trellis section; and (3) the total number of composite branches in the trellis section. The overall branch complexity of the trellis is then the sum of the trellis section branch complexities. These three branch complexity parameters can be expressed in terms of the dimensions of C_{h_{j-1},h_j}, $C_{0,h_{j-1}}$, $C_{h_j,N}$, and $p_{h_{j-1},h_j}(C)$ which can be obtained from the TOGM G of C.

Let $\sigma_{h_{j-1}}$ and σ_{h_j} be two adjacent states with $\sigma_{h_{j-1}} \in \Sigma_{h_{j-1}}(C)$ and $\sigma_{h_j} \in \Sigma_{h_j}(C)$. It has been shown in Section 3.7 that the parallel branches in $L(\sigma_{h_{j-1}}, \sigma_{h_j})$ form a coset in the partition

$$p_{h_{j-1},h_j}(C)/C^{\mathrm{tr}}_{h_{j-1},h_j}.$$

Therefore, the number of parallel branches between two adjacent states $\sigma_{h_{j-1}}$ and σ_{h_j} in the j-th section of $T(U)$ is

$$\left|L(\sigma_{h_{j-1}}, \sigma_{h_j})\right| = 2^{k(C_{h_{j-1},h_j})}. \tag{6.4}$$

In Section 3.7, we have also shown that

$$L(\sigma_0, \sigma_{h_{j-1}}) \in p_{0,h_{j-1}}(C)/C^{\rm tr}_{0,h_{j-1}} \tag{6.5}$$

and

$$L(\sigma_0, \sigma_{h_j}) \in p_{0,h_j}(C)/C^{\rm tr}_{0,h_j}. \tag{6.6}$$

If $\sigma_{h_{j-1}}$ and σ_{h_j} are adjacent, then

$$\begin{aligned} &L(\sigma_0, \sigma_{h_{j-1}}) \circ L(\sigma_{h_{j-1}}, \sigma_{h_j}) \triangleq \\ &\qquad \{u \circ v : u \in L(\sigma_0, \sigma_{h_{j-1}}) \text{ and } v \in L(\sigma_{h_{j-1}}, \sigma_{h_j})\}, \end{aligned} \tag{6.7}$$

is the set of paths in $T(U)$ that diverge from the initial state σ_0, converge at the state $\sigma_{h_{j-1}}$, and then transverse the parallel branches in $L(\sigma_{h_{j-1}}, \sigma_{h_j})$ to the state σ_{h_j} as shown in Figure 6.2. It has also been shown in Section 3.7 that $L(\sigma_0, \sigma_{h_{j-1}}) \circ L(\sigma_{h_{j-1}}, \sigma_{h_j})$ is a subcode of a coset in the partition $p_{0,h_j}(C)/C^{\rm tr}_{0,h_j}$. Let $\Sigma_{h_{j-1}}(\sigma_{h_j})$ denote the set of states in the state space $\Sigma_{h_{j-1}}(C)$ that are adjacent to state σ_{h_j} as shown in Figure 6.2. Then

$$\bigcup_{\sigma_{h_{j-1}} \in \Sigma_{h_{j-1}}(\sigma_{h_j})} L(\sigma_0, \sigma_{h_{j-1}}) \circ L(\sigma_{h_{j-1}}, \sigma_{h_j}) \triangleq L(\sigma_0, \sigma_{h_j}) \tag{6.8}$$

is a coset in the partition $p_{0,h_j}(C)/C^{\rm tr}_{0,h_j}$.

Note that the dimensions of $L(\sigma_0, \sigma_{h_{j-1}})$, $L(\sigma_{h_{j-1}}, \sigma_{h_j})$ and $L(\sigma_0, \sigma_{h_j})$ are $k(C^{\rm tr}_{0,h_{j-1}})$, $k(C^{\rm tr}_{h_{j-1},h_j})$ and $k(C^{\rm tr}_{0,h_j})$, respectively. It follows from (6.8) that the number of states in $\Sigma_{h_{j-1}}(\sigma_{h_j})$ is given by

$$\begin{aligned} \left|\Sigma_{h_{j-1}}(\sigma_{h_j})\right| &= 2^{k(C^{\rm tr}_{0,h_j}) - k(C^{\rm tr}_{0,h_{j-1}}) - k(C^{\rm tr}_{h_{j-1},h_j})} \\ &= 2^{k(C_{0,h_j}) - k(C_{0,h_{j-1}}) - k(C_{h_{j-1},h_j})}. \end{aligned} \tag{6.9}$$

This implies that the number of composite branches converging into a state $\sigma_{h_j} \in \Sigma_{h_j}(C)$, called the **incoming degree** of σ_{h_j}, is given by

$$\deg(\sigma_{h_j})_{\rm in} \triangleq 2^{k(C_{0,h_j}) - k(C_{0,h_{j-1}}) - k(C_{h_{j-1},h_j})}. \tag{6.10}$$

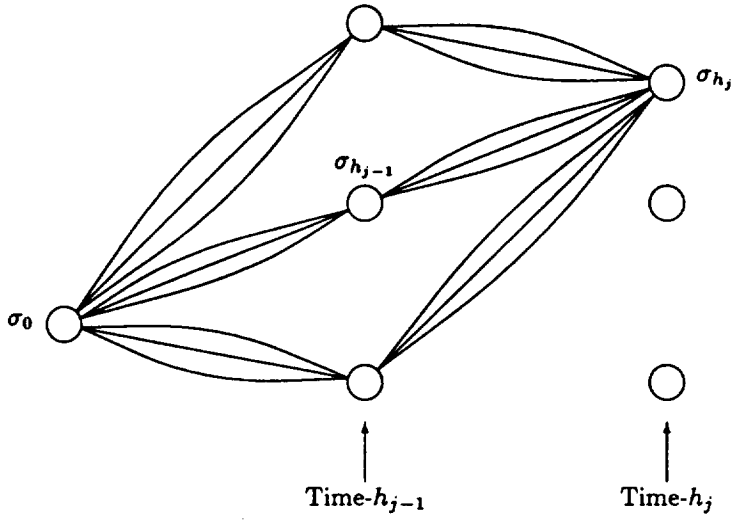

Figure 6.2. State connectivity.

This number is a measure of the state **connectivity** of the sectionalized code trellis $T(U)$. In an IC implementation of a Viterbi decoder, this number is known as the **radix number**, a key design parameter.

Since each composite branch $L(\sigma_{h_{j-1}}, \sigma_{h_j})$ in the j-th section of $T(U)$ is a coset in the partition $p_{h_{j-1},h_j}(C)/C^{tr}_{h_{j-1},h_j}$, the number of distinct composite branches in the j-th section of $T(U)$ is

$$2^{k(p_{h_{j-1},h_j}(C))-k(C_{h_{j-1},h_j})}. \tag{6.11}$$

It follows from (6.2) and (6.10) that the total number of composite branches in the j-th section of $T(U)$ is given by

$$2^{K-k(C_{0,h_{j-1}})-k(C_{h_j,N})-k(C_{h_{j-1},h_j})}. \tag{6.12}$$

From (6.11) and (6.12), we find that each distinct composite branch appears in the j-th section of $T(U)$

$$2^{K-k(C_{0,h_{j-1}})-k(C_{h_j,N})-k(p_{h_{j-1},h_j}(C))} \tag{6.13}$$

times.

From (6.2) (with h_j replaced by h_{j-1}) and (6.12), we can compute the number of composite branches diverging from a state $\sigma_{h_{j-1}} \in \Sigma_{h_{j-1}}(C)$ at time-h_{j-1} as

$$2^{k(C_{h_{j-1},N})-k(C_{h_j,N})-k(C_{h_{j-1},h_j})}, \tag{6.14}$$

which is called the **outgoing degree** of $\sigma_{h_{j-1}}$, denoted $\deg(\sigma_{h_{j-1}})_{\text{out}}$. Equations (6.4), (6.10)–(6.12) and (6.14) give the branch complexity and state connectivity of the j-th section of a minimal L-section trellis $T(U)$ with section boundary locations in $U = \{h_0, h_1, \ldots, h_L\}$.

Define

$$\delta_j \triangleq \log_2 \deg(\sigma_{h_j})_{\text{in}}$$

with $\sigma_{h_j} \in \Sigma_{h_j}(C)$. The ordered sequence $(\delta_1, \delta_2, \ldots, \delta_L)$ is called the **converging branch dimension profile (CBDP)**. Define

$$\lambda_j \triangleq \log_2 \deg(\sigma_{h_j})_{\text{out}}.$$

The ordered sequence $(\lambda_0, \lambda_1, \ldots, \lambda_{L-1})$ is called the **diverging branch dimension profile (DBDP)**.

Let M_j denote the total number of composite branches in the j-th trellis section (given by (6.12)) and define

$$\beta_j \triangleq \log_2 M_j.$$

The ordered sequence

$$(\beta_1, \beta_2, \ldots, \beta_L)$$

is called the **branch complexity profile (BCP)**. The branch complexity of the minimal L-section trellis $T(U)$ in terms of the total number of branches in the trellis is given by

$$B = \sum_{j=1}^{L} 2^{\beta_j} \cdot 2^{k(C_{h_{j-1},h_j})}. \tag{6.15}$$

Since each branch in the j-th section of $T(U)$ represents $h_j - h_{j-1}$ code bits, it is equivalent to $h_j - h_{j-1}$ branches in the bit-level N-section trellis T for the code. Therefore, the branch complexity in terms of bit branches is given by

$$E = \sum_{j=1}^{L} (h_j - h_{j-1}) \cdot 2^{\beta_j} \cdot 2^{k(C_{h_{j-1},h_j})}. \tag{6.16}$$

If the section boundary is $U = \{0, 1, 2, \ldots, N\}$, then (6.16) gives the branch (or edge) complexity of the bit-level N-section minimal trellis of the code.

6.3 A PROCEDURE FOR CONSTRUCTING A MINIMAL L-SECTION TRELLIS

A minimal L-section trellis diagram for an (N, K) linear block code C can be constructed directly from the TOGM G. Let $U = \{h_0, h_1, \ldots, h_L\}$ be the set of section boundary locations with $h_0 = 0 < h_1 < \cdots < h_{L-1} < h_L = N$. Again the construction of the minimal L-section trellis diagram $T(U)$ with section boundary set U is carried out serially, section by section. Suppose $T(U)$ has been constructed up to the j-th section (i.e., up to time-h_j) with $1 \leq j < L$. Now we begin to construct the $(j+1)$-th section from time-h_j to time-h_{j+1}. Partition the rows of the TOGM G into three disjoint subsets, $G^p_{h_j}$, $G^f_{h_j}$, and $G^s_{h_j}$ as follows (also shown in Figure 6.3):

(1) $G^p_{h_j}$ consists of those rows in G whose spans are contained in the interval $[1, h_j]$.

(2) $G^f_{h_j}$ consists of those rows in G whose spans are contained in the interval $[h_j + 1, N]$.

(3) $G^s_{h_j}$ consists of those rows in G whose active spans contain the time index h_j.

It is clear that $G^p_{h_j}$ and $G^f_{h_j}$ generate the past and future codes, C_{0,h_j} and $C_{h_j,N}$, respectively. Let $A^s_{h_j}$ be the set of information bits that correspond to the rows of $G^s_{h_j}$. Then the bits in $A^s_{h_j}$ define the state space $\Sigma_{h_j}(C)$ at time-h_j. That is, for any binary $\rho_{h_j} = |A^s_{h_j}|$-tuple, which represents values of information bits in $A^s_{h_j}$, there is a corresponding state in $\Sigma_{h_j}(C)$.

To determine the composite branches between states in $\Sigma_{h_j}(C)$ and states in $\Sigma_{h_{j+1}}(C)$ and the parallel branches between two adjacent states, we further partition the rows of $G^f_{h_j}$ into three subsets, $G^{f,p}_{h_j,h_{j+1}}$, $G^{f,s}_{h_j,h_{j+1}}$ and $G^f_{h_{j+1}}$ as follows (see Figure 6.3):

(1) $G^{f,p}_{h_j,h_{j+1}}$ consists of those rows of $G^f_{h_j}$ whose spans are contained in the interval $[h_j + 1, h_{j+1}]$.

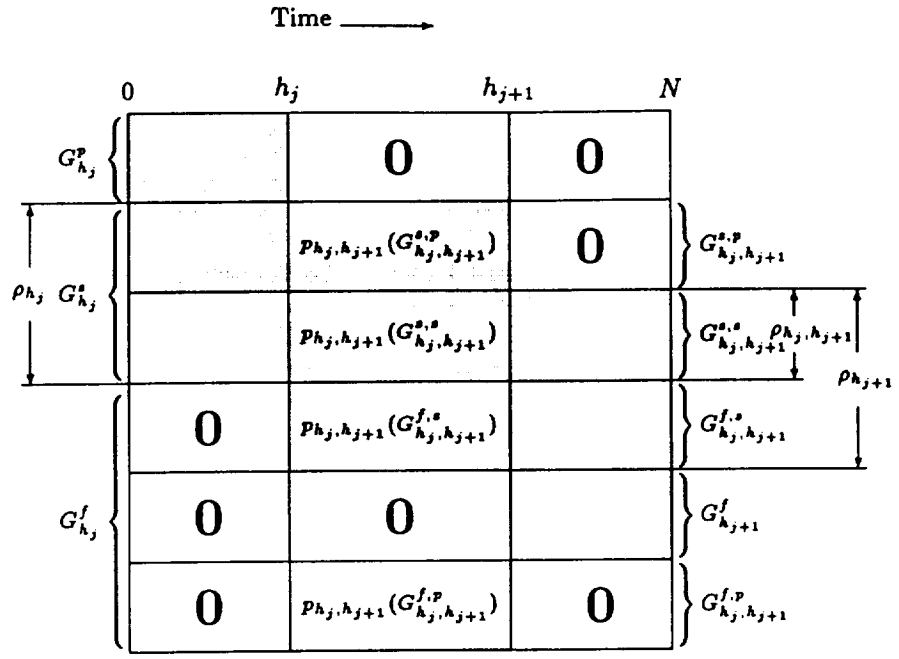

Figure 6.3. Partition of the TOGM G.

(2) $G^{f,s}_{h_j,h_{j+1}}$ consists of those rows of $G^f_{h_j}$ whose active spans contain the time index h_{j+1}.

(3) The remaining rows in $G^f_{h_j}$ form $G^f_{h_{j+1}}$.

Let $A^{f,p}_{h_j,h_{j+1}}$ and $A^{f,s}_{h_j,h_{j+1}}$ denote the subsets of information bits that correspond to the rows in $G^{f,p}_{h_j,h_{j+1}}$ and $G^{f,s}_{h_j,h_{j+1}}$, respectively. Then the information bits in these two sets may be regarded as the current input information bits. These input bits together with the state of the encoder at time-h_j uniquely determine the output code bits between time-h_j and time-h_{j+1}. Note that the information bits in $A^{f,p}_{h_j,h_{j+1}}$ only affect the output during the interval between time-h_j and time-h_{j+1}. Therefore, they determine the parallel branches

between two adjacent states. The information bits in $A^{f,s}_{h_j,h_{j+1}}$ determine the diverging composite branches from a state in $\Sigma_{h_j}(C)$.

Let $p_{h_j,h_{j+1}}(G^{f,p}_{h_j,h_{j+1}})$, $p_{h_j,h_{j+1}}(G^{f,s}_{h_j,h_{j+1}})$ and $p_{h_j,h_{j+1}}(G^{s}_{h_j})$ denote the truncations of $G^{f,p}_{h_j,h_{j+1}}$, $G^{f,s}_{h_j,h_{j+1}}$ and $G^{s}_{h_j}$ from time-h_j to time-h_{j+1}. The rows in $p_{h_j,h_{j+1}}(G^{f,p}_{h_j,h_{j+1}})$ span the code $C^{\mathrm{tr}}_{h_j,h_{j+1}}$, and the rows in $p_{h_j,h_{j+1}}(G^{f,p}_{h_j,h_{j+1}})$, $p_{h_j,h_{j+1}}(G^{f,s}_{h_j,h_{j+1}})$ and $p_{h_j,h_{j+1}}(G^{s}_{h_j})$ span the truncated code $p_{h_j,h_{j+1}}(C)$. Then every composite branch between a state $\sigma_{h_j} \in \Sigma_{h_j}(C)$ and a state $\sigma_{h_{j+1}} \in \Sigma_{h_{j+1}}(C)$ is a coset in the partition $p_{h_j,h_{j+1}}(C)/C^{\mathrm{tr}}_{h_j,h_{j+1}}$. The number of parallel branches between two adjacent states is therefore $|C^{\mathrm{tr}}_{h_j,h_{j+1}}|$.

Let σ_{h_j} be the state at time-h_j defined by the binary ρ_{h_j}-tuple formed by the binary information bits in the set $A^{s}_{h_j}$,

$$(a^{(j)}_1, a^{(j)}_2, \ldots, a^{(j)}_{\rho_{h_j}}), \tag{6.17}$$

where $\rho_{h_j} = \log_2 |\Sigma_{h_j}(C)| = |G^{s}_{h_j}|$. Let $\{g^{(j)}_1, g^{(j)}_2, \ldots, g^{(j)}_{\rho_{h_j}}\}$ denote the rows in $G^{s}_{h_j}$. Then

$$u = a^{(j)}_1 \cdot g^{(j)}_1 + a^{(j)}_2 \cdot g^{(j)}_2 + \cdots + a^{(j)}_{\rho_{h_j}} \cdot g^{(j)}_{\rho_{h_j}} \tag{6.18}$$

is a codeword (or path) passing through the state σ_{h_j} at time-h_j. Let $p_{h_j,h_{j+1}}(u)$ denote the branch on u from time-h_j to time-h_{j+1}. Let $B_{h_j,h_{j+1}}$ denote the code of length $h_{j+1} - h_j$ generated by $p_{h_j,h_{j+1}}(G^{f,s}_{h_j,h_{j+1}})$. Then for every vector $b \in B_{h_j,h_{j+1}}$, there is a composite branch diverging from the state σ_{h_j} which consists of the following parallel branches,

$$\{p_{h_j,h_{j+1}}(u) + b + x : x \in C^{\mathrm{tr}}_{h_j,h_{j+1}}\}. \tag{6.19}$$

Therefore, the number of composite branches diverging from the state σ_{h_j} is $|B_{h_j,h_{j+1}}|$.

Next, we analyze the state transitions. Partition the matrix $G^{s}_{h_j}$ into two submatrices, $G^{s,p}_{h_j,h_{j+1}}$ and $G^{s,s}_{h_j,h_{j+1}}$, where

(1) $G^{s,p}_{h_j,h_{j+1}}$ consists of those rows in $G^{s}_{h_j}$ whose active spans do not contain the time index h_{j+1}, and

(2) $G^{s,s}_{h_j,h_{j+1}}$ consists of those rows in $G^{s}_{h_j}$ whose active spans contain the time index h_{j+1}.

Let $A^{s,p}_{h_j,h_{j+1}}$ and $A^{s,s}_{h_j,h_{j+1}}$ denote the sets of information bits corresponding to $G^{s,p}_{h_j,h_{j+1}}$ and $G^{s,s}_{h_j,h_{j+1}}$, respectively. Then the set of information bits that defines the state space $\Sigma_{h_{j+1}}(C)$ at time-h_{j+1} is given by

$$\begin{aligned} A^{s}_{h_{j+1}} &= (A^{s}_{h_j} \setminus A^{s,p}_{h_j,h_{j+1}}) \cup A^{f,s}_{h_j,h_{j+1}} \\ &= A^{s,s}_{h_j,h_{j+1}} \cup A^{f,s}_{h_j,h_{j+1}}. \end{aligned} \tag{6.20}$$

Therefore, the state transitions from time-h_j to time-h_{j+1} are completely specified by the change from $A^{s}_{h_j}$ to $A^{s}_{h_{j+1}}$.

Define

$$\rho_{h_j,h_{j+1}} \triangleq |A^{s,s}_{h_j,h_{j+1}}| = |G^{s,s}_{h_j,h_{j+1}}|. \tag{6.21}$$

Then it follows from (6.20) (also Figure 6.3) that

$$|A^{s,p}_{h_j,h_{j+1}}| = \rho_{h_j} - \rho_{h_j,h_{j+1}}, \tag{6.22}$$

and

$$|A^{f,s}_{h_j,h_{j+1}}| = \rho_{h_{j+1}} - \rho_{h_j,h_{j+1}}. \tag{6.23}$$

Let $\boldsymbol{a}^0_{h_j}$ be the binary $(\rho_{h_j} - \rho_{h_j,h_{j+1}})$-tuple formed by the binary information bits in the set $A^{s,p}_{h_j,h_{j+1}}$, $\boldsymbol{a}_{h_j}$ be the binary $\rho_{h_j,h_{j+1}}$-tuple formed by the information bits in the set $A^{s,s}_{h_j,h_{j+1}}$, and $\boldsymbol{a}^*_{h_j}$ be the binary $(\rho_{h_{j+1}} - \rho_{h_j,h_{j+1}})$-tuple formed by the information bits in the set $A^{f,s}_{h_j,h_{j+1}}$. Then $(\boldsymbol{a}^0_{h_j}, \boldsymbol{a}_{h_j})$ defines a state, denoted $\sigma(\boldsymbol{a}^0_{h_j}, \boldsymbol{a}_{h_j})$, in the state space $\Sigma_{h_j}(C)$ at time-h_j, and $(\boldsymbol{a}_{h_j}, \boldsymbol{a}^*_{h_j})$ defines a state, denoted $\sigma(\boldsymbol{a}_{h_j}, \boldsymbol{a}^*_{h_j})$, in the state space $\Sigma_{h_{j+1}}(C)$ at time-h_{j+1}. State $\sigma(\boldsymbol{a}^0_{h_j}, \boldsymbol{a}_{h_j})$ is adjacent to state $\sigma(\boldsymbol{a}_{h_j}, \boldsymbol{a}^*_{h_j})$. The composite branch that connects these two states in the trellis is given by (6.19) with

$$\boldsymbol{b} = \boldsymbol{a}^*_{h_j} \cdot p_{h_j,h_{j+1}}(G^{f,s}_{h_j,h_{j+1}}). \tag{6.24}$$

Note that these two states share a common $\boldsymbol{a}_{h_j}$. For $\boldsymbol{a}'_{h_j} \neq \boldsymbol{a}_{h_j}$, the state $\sigma(\boldsymbol{a}^0_{h_j}, \boldsymbol{a}_{h_j})$ at time-h_j is not adjacent to the state $\sigma(\boldsymbol{a}'_{h_j}, \boldsymbol{a}^*_{h_j})$ at time-h_{j+1}. Therefore, from each state $\sigma(\boldsymbol{a}^0_{h_j}, \boldsymbol{a}_{h_j})$ in $\Sigma_{h_j}(C)$, there are $2^{\rho_{h_{j+1}} - \rho_{h_j,h_{j+1}}}$ possible transitions to the states $\sigma(\boldsymbol{a}_{h_j}, \boldsymbol{a}^*_{h_j})$ in $\Sigma_{h_{j+1}}(C)$ with $\boldsymbol{a}^*_{h_j} \in \{0,1\}^{\rho_{h_{j+1}} - \rho_{h_j,h_{j+1}}}$. This completely specifies the state transitions from time-h_j to time-h_{j+1}.

State labeling based on the state defining information set $A^{s}_{h_j}$ with $0 \leq j \leq L$ is exactly the same as described in Section 4.1. We may use either a K-tuple

or a $\rho_{L,\max}(C)$-tuple to label a state. In general, $\rho_{L,\max}(C)$ is much smaller than K, and hence using a $\rho_{L,\max}(C)$-bit label for a state is more efficient.

Suppose an L-section trellis diagram $T(U)$ with boundary location set $U = \{h_0, h_1, \ldots, h_L\}$ has been constructed up to time-h_j. The trellis section from time-h_j to time-h_{j+1} can be constructed by the following procedure:

(1) Form and label the states in $\Sigma_{h_{j+1}}(C)$ based on $A^s_{h_{j+1}}$.

(2) For each state in $\Sigma_{h_j}(C)$, determine its transitions to the states in $\Sigma_{h_{j+1}}(C)$ based on the state transition rules described above.

(3) For two adjacent states, $\sigma(a^0_{h_j}, a_{h_j})$ and $\sigma(a_{h_j}, a^*_{h_j})$, at time-$h_j$ and time-h_{j+1}, connect them by parallel branches given by (6.19).

Repeat the above procedure until the L-section trellis $T(U)$ is completed.

Example 6.2 Consider the $(8, 4, 4)$ RM code with the TOGM G as

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

Suppose we want to construct a 4-section trellis for this code with boundary locations in $U = \{0, 2, 4, 6, 8\}$. First, we find from G that the state space dimension profile is $(0, 2, 2, 2, 0)$. Therefore, $\rho_{4,\max}(C) = 2$. We also find that

$$C^{\mathrm{tr}}_{0,2} = C^{\mathrm{tr}}_{2,4} = C^{\mathrm{tr}}_{4,6} = C^{\mathrm{tr}}_{6,8} = \{\mathbf{0}\},$$

$$p_{0,2}(G^{f,s}_{0,2}) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \qquad p_{2,4}(G^{f,s}_{2,4}) = \begin{bmatrix} 1 & 1 \end{bmatrix},$$

$$p_{4,6}(G^{f,s}_{4,6}) = \begin{bmatrix} 1 & 1 \end{bmatrix}, \qquad p_{6,8}(G^{f,s}_{6,8}) = \emptyset.$$

The state defining information sets at the boundary locations, 0, 2, 4, 6, and 8, are given in Table 6.1. Following the constructing procedure given above, we obtain the 4-section trellis diagram for the $(8, 4, 4)$ RM code as shown in Figure 6.4, where the states are labeled based on the information defining sets using $\rho_{4,\max}(C)$-tuples with $\rho_{4,\max}(C) = 2$.

△△

Table 6.1. State defining information sets for a 4-section trellis for the $(8, 4, 4)$ RM code.

	Time $\longrightarrow$				
	0	2	4	6	8
$A^s_{h_j}$	$\emptyset$	$\{a_1, a_2\}$	$\{a_2, a_3\}$	$\{a_2, a_4\}$	$\emptyset$
$A^{s,p}_{h_j,h_{j+1}}$	$\emptyset$	$\{a_1\}$	$\{a_3\}$	$\{a_2, a_4\}$	$\emptyset$
$A^{f,s}_{h_j,h_{j+1}}$	$\{a_1, a_2\}$	$\{a_3\}$	$\{a_4\}$	$\emptyset$	$\emptyset$

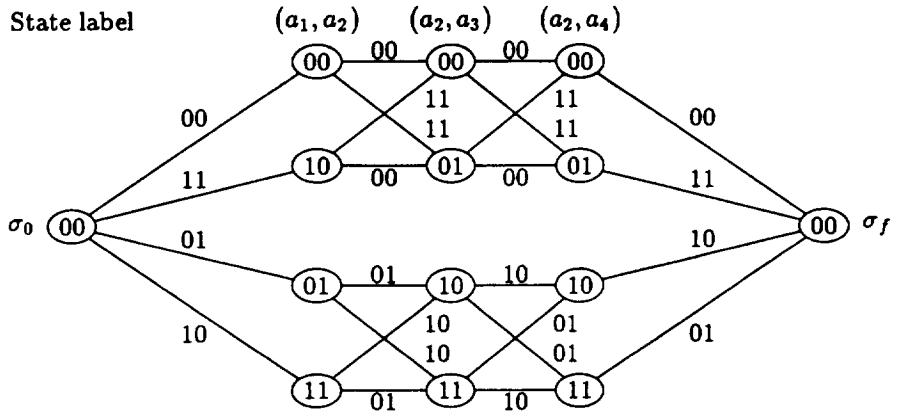

Figure 6.4. A minimal 4-section trellis diagram for the $(8, 4, 4)$ RM code with 2-bit state labels.

Construction of $T(U)$ can be achieved by using the state labeling based on a parity-check matrix H for C [101]. Let

$$H_{h_j,h_{j+1}} = [h_{h_j+1}, h_{h_j+2}, \cdots, h_{h_{j+1}}] \tag{6.25}$$

denote the submatrix of the parity-check matrix H of C that consists of columns from h_{h_j+1} to $h_{h_{j+1}}$. Let $l(\sigma_{h_j})$ be the label for the state σ_{h_j}. Then the composite branch given by (6.19) connects the state σ_{h_j} to the state $\sigma_{h_{j+1}} \in \Sigma_{h_{j+1}}(C)$ at time-h_{j+1} that is labeled by

$$l(\sigma_{h_{j+1}}) = l(\sigma_{h_j}) + (p_{h_j,h_{j+1}}(u) + b) \cdot H^T_{h_j,h_{j+1}}. \tag{6.26}$$

Eq.(6.26) gives the connection from a starting state σ_{h_j} at time-h_j to a destination state $\sigma_{h_{j+1}}$ at time-h_{j+1}.

To facilitate the construction of the $(j+1)$-th trellis section between time-h_j and time-h_{j+1}, we form a table, denoted Q_{h_j}, at the completion of the construction of the j-th section. Each entry in Q_{h_j} is a triplet,

$$(\boldsymbol{f}, l(\sigma), \boldsymbol{c}).$$

The first component $\boldsymbol{f}$ is a binary ρ_{h_j}-tuple formed by a specific combination of the ρ_{h_j} information bits in $A^s_{h_j}$. This ρ_{h_j}-tuple defines a specific state σ in $\Sigma_{h_j}(C)$. The second component $l(\sigma)$ is simply the label of state σ. The third component is given by

$$\begin{aligned} \boldsymbol{c} &= p_{h_j,h_{j+1}}(\boldsymbol{u}) \\ &= p_{h_j,h_{j+1}}(\boldsymbol{f} \cdot G^s_{h_j}), \end{aligned} \tag{6.27}$$

where $\boldsymbol{u} = \boldsymbol{f} \cdot G^s_{h_j}$ is given by (6.18).

Construction of the $(j+1)$-th section of $T(U)$ is carried out as follows:

(1) Form $C^{\mathrm{tr}}_{h_j,h_{j+1}}$ and $B_{h_j,h_{j+1}}$.

(2) For every entry $(\boldsymbol{f}, l(\sigma), \boldsymbol{c}) \in Q_{h_j}$ and every $\boldsymbol{b} \in B_{h_j,h_{j+1}}$, form the composite branch,

$$\{\boldsymbol{b} + \boldsymbol{c} + \boldsymbol{x} : \boldsymbol{x} \in C^{\mathrm{tr}}_{h_j,h_{j+1}}\}, \tag{6.28}$$

(3) For every starting state $\sigma \in \Sigma_{h_j}(C)$, and $\boldsymbol{b} \in B_{h_j,h_{j+1}}$, the destination state $\sigma' \in \Sigma_{h_{j+1}}(C)$ is labeled with

$$l(\sigma') = l(\sigma) + (\boldsymbol{b} + \boldsymbol{c}) \cdot H^T_{h_j,h_{j+1}}. \tag{6.29}$$

Repeat the above process until the L-section trellis $T(U)$ is completed.

The trellis construction procedures presented in Section 4.1, Section 4.2, and this section only provide the general steps of construction. A detail and efficient trellis construction procedure is given in Appendix A.

6.4 PARALLEL STRUCTURE

Consider the trellis section from time-h_j to time-h_{j+1}. For a given $\rho_{h_j,h_{j+1}}$-tuple a_{h_j}, define the following two sets of states at time-h_j and time-h_{j+1}, respectively:

$$S_L(a_{h_j}) \triangleq \left\{\sigma(a^0_{h_j}, a_{h_j}) : a^0_{h_j} \in \{0,1\}^{\rho_{h_j} - \rho_{h_j,h_{j+1}}}\right\} \tag{6.30}$$

and

$$S_R(a_{h_j}) \triangleq \left\{\sigma(a_{h_j}, a^*_{h_j}) : a^*_{h_j} \in \{0,1\}^{\rho_{h_{j+1}} - \rho_{h_j,h_{j+1}}}\right\}. \tag{6.31}$$

Then $S_L(a_{h_j})$ and $S_R(a_{h_j})$ are subspaces of the state spaces, $\Sigma_{h_j}(C)$ and $\Sigma_{h_{j+1}}(C)$, respectively. From the state transition analysis given in the previous section, we observe the following:

(1) Every state in $S_L(a_{h_j})$ is adjacent to all the $2^{\rho_{h_{j+1}} - \rho_{h_j,h_{j+1}}}$ states in $S_R(a_{h_j})$ and is not adjacent to any other state in $\Sigma_{h_{j+1}}(C)$.

(2) Every state in $S_R(a_{h_j})$ is adjacent from all the $2^{\rho_{h_j} - \rho_{h_j,h_{j+1}}}$ states in $S_L(a_{h_j})$ and is not adjacent from any other state in $\Sigma_{h_j}(C)$.

Therefore, the states in $S_L(a_{h_j})$, the states in $S_R(a_{h_j})$, and the composite branches connecting them form a **completely connected subtrellis** (known as a **complete bipartite graph**). Since there are $2^{\rho_{h_j,h_{j+1}}}$ possible $\rho_{h_j,h_{j+1}}$-tuple a_{h_j}, there are $2^{\rho_{h_j,h_{j+1}}}$ such completely connected subtrellises in the trellis section time-h_j and time-h_{j+1}. All these subtrellises are **structurally identical (isomorphic)**, and there are no cross connections between them. These subtrellises are called **parallel components**. The parallel structure of a trellis section is shown in Figure 6.5.

It follows from the definition of $\rho_{h_j,h_{j+1}}$ given by (6.21) and the partition of the TOGM G shown in Figure 6.3 that

$$\rho_{h_j,h_{j+1}} = K - k(C_{h_j,N}) - k(C_{0,h_{j+1}}) + k(C_{h_j,h_{j+1}}). \tag{6.32}$$

Therefore, the total number of parallel components in the trellis section from time-h_j to time-h_{j+1} is

$$2^{K - k(C_{h_j,N}) - k(C_{0,h_{j+1}}) + k(C_{h_j,h_{j+1}})}. \tag{6.33}$$

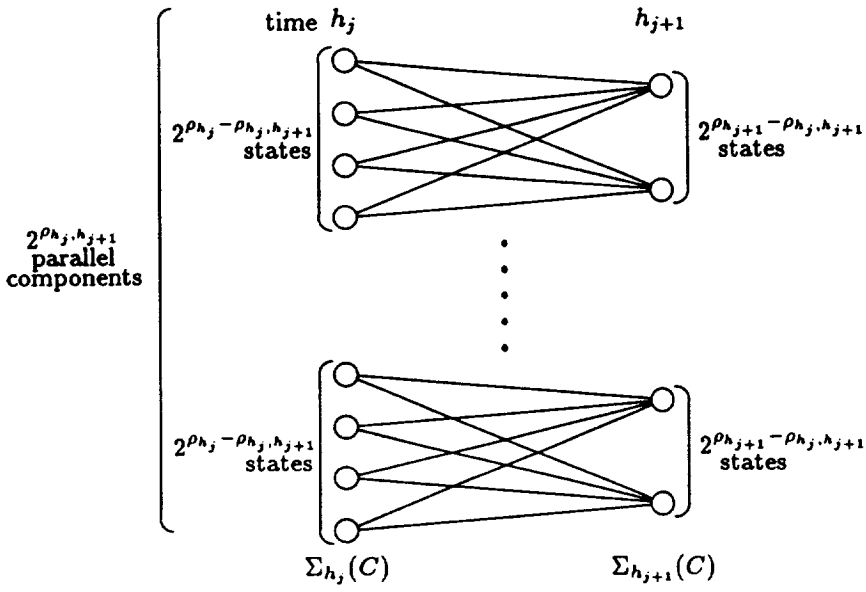

Figure 6.5. Parallel structure in a trellis section.

From (6.2) and (6.33), we find that the numbers of states in $S_L(a_{h_j})$ and $S_R(a_{h_j})$ are:

$$2^{k(C_{0,h_{j+1}})-k(C_{0,h_j})-k(C_{h_j,h_{j+1}})} \tag{6.34}$$

and

$$2^{k(C_{h_j,N})-k(C_{h_{j+1},N})-k(C_{h_j,h_{j+1}})}, \tag{6.35}$$

respectively. Equations (6.30) to (6.35) completely characterize the parallel components in the $(j+1)$-th trellis section of $T(U)$.

Consider the 4-section trellis diagram for the $(8,4)$ RM code shown in Figure 6.4. There are two parallel components in both the second and third sections of the trellis. Each component has two states at each end.

Analysis of the parallel structure of a sectionalized code trellis is presented in the Appendix A.

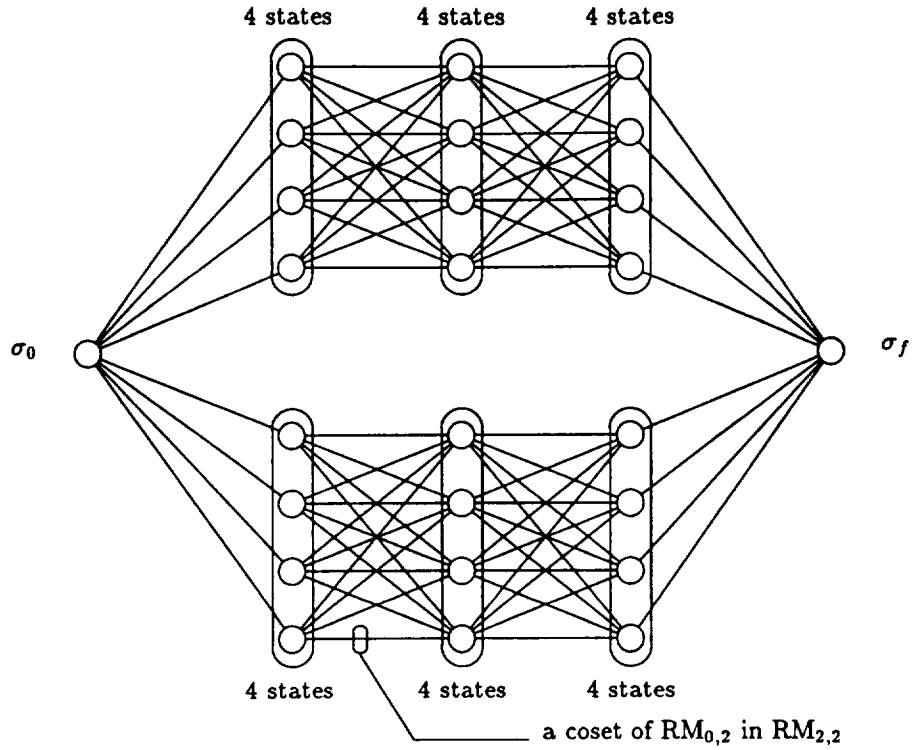

Figure 6.6. The 4-section minimal trellis diagram $T(\{0, 4, 8, 12, 16\})$ for $\mathrm{RM}_{2,4}$.

Example 6.3 Consider the $\mathrm{RM}_{2,4}$ code which is a (16,11) code. The 4-section minimal trellis diagram $T(U)$ with section boundary locations in $U = \{0, 4, 8, 12, 16\}$ is depicted in Figure 6.6. There are two parallel and structural identical components in both the second and third sections of the trellis, and each component is a complete bipartite graph. Each component has 4 states at each end. Each state at boundary location 8 has 4 composite branches diverging from it. For $1 \leq j \leq 4$, $p_{4(j-1),4j}(\mathrm{RM}_{2,4}) = \mathrm{RM}_{2,2}$ and $C^{\mathrm{tr}}_{4(j-1),4j} = \mathrm{RM}_{0,2}$. Therefore, there are 2 parallel branches between any two adjacent states whose 4-bit label sequences form a coset of $\mathrm{RM}_{0,2}$ in $\mathrm{RM}_{2,2}$. In both the second and third sections of the trellis, each coset in $\mathrm{RM}_{2,2}/\mathrm{RM}_{0,2}$ appears 4 times as the

composite branch label. In fact, the entire trellis consists of two 4-section parallel and structurally identical subtrellises without cross connections between them. The maximum state complexity is 8. Therefore it is possible to devise two identical trellis-based decoders, say Viterbi decoders, to process the entire trellis in parallel. This not only simplifies the decoding complexity but also speeds up the decoding process.

△△

The parallel components in a trellis section can be partitioned into groups of the same size in such a way that [44]: (1) two parallel components in the same group are **identical up to path labeling**, and (2) if there is a common label sequence in two parallel components, then they are in the same group. Since all the parallel components in a group have the same label set, in a trellis-based decoding algorithm, only the metrics of the branches in one of the parallel components need to be computed. This results in a reduction of branch metric computation.

Let $\bar{C}_{h_j,h_{j+1}}$ denote the subcode of C that consists of those codewords whose components from the (h_j+1)-th bit to the h_{j+1}-th bit positions are all zero. Then each group consists of $2^{\lambda_{h_j,h_{j+1}}(C)}$ identical parallel components where [44]

$$\lambda_{h_j,h_{j+1}}(C) = k(\bar{C}_{h_j,h_{j+1}}) - k(p_{0,h_j}(C_{0,h_{j+1}}) \cap p_{0,h_j}(\bar{C}_{h_j,h_{j+1}})) - k(C_{h_{j+1},N}). \tag{6.36}$$

Each parallel component can be decomposed into subtrellises with simple **uniform structure** [44] as shown in Figure 6.7. Consider a parallel component, denoted Λ. Let $\Sigma_{h_j}(\Lambda)$ and $\Sigma_{h_{j+1}}(\Lambda)$ denote the state spaces at two ends of Λ. We first partition $\Sigma_{h_j}(\Lambda)$ into blocks, called **left U-blocks**, which satisfy the following condition:

(B1) If two states σ_{h_j} and σ'_{h_j} in $\Sigma_{h_j}(\Lambda)$ are in the same left U-block, then they have the same set of diverging composite branches, i.e.,

$$\begin{aligned} &\{L(\sigma_{h_j},\sigma_{h_{j+1}}) : \sigma_{h_{j+1}} \in \Sigma_{h_{j+1}}(\Lambda)\} \\ &= \{L(\sigma'_{h_j},\sigma_{h_{j+1}}) : \sigma_{h_{j+1}} \in \Sigma_{h_{j+1}}(\Lambda)\} \end{aligned} \tag{6.37}$$

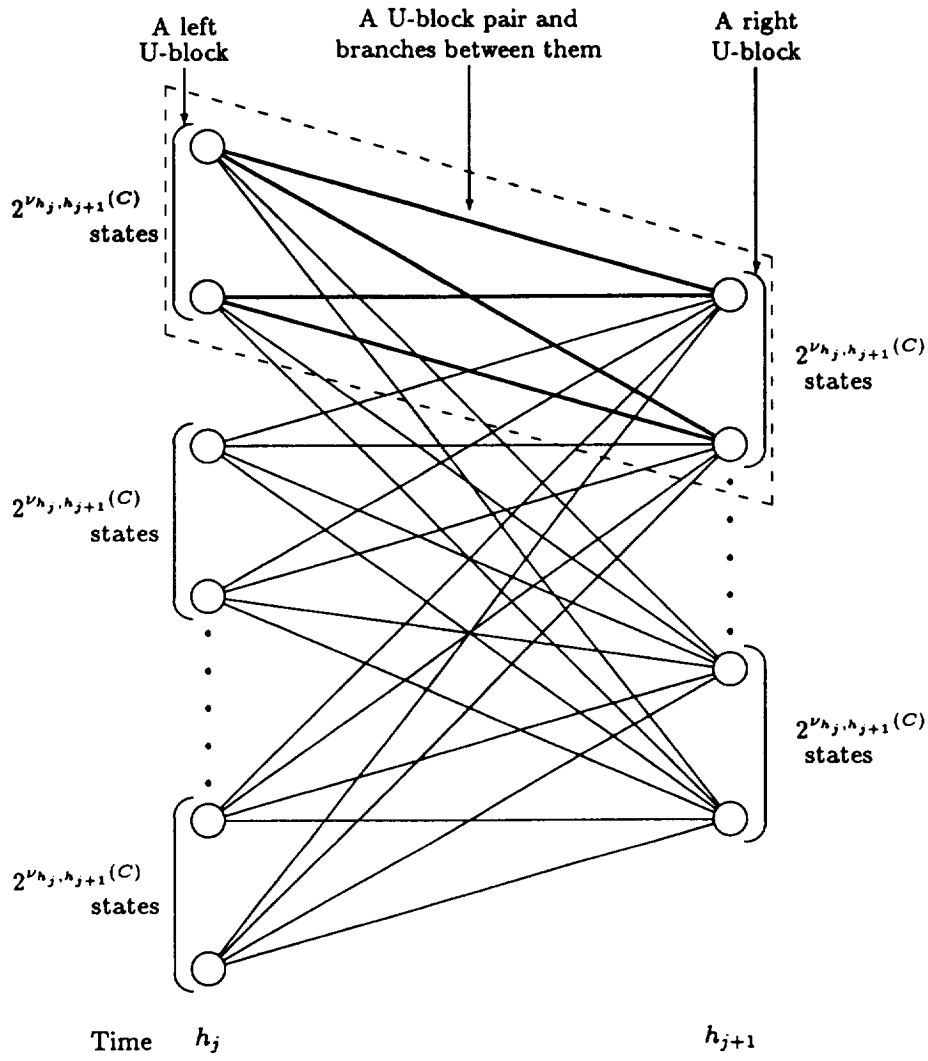

Figure 6.7. Partition of a parallel component.

and otherwise

$$\{L(\sigma_{h_j},\sigma_{h_{j+1}}) : \sigma_{h_{j+1}} \in \Sigma_{h_{j+1}}(\Lambda)\} \cap \{L(\sigma'_{h_j},\sigma_{h_{j+1}}) : \sigma_{h_{j+1}} \in \Sigma_{h_{j+1}}(\Lambda)\} = \emptyset. \quad (6.38)$$

We next partition $\Sigma_{h_{j+1}}(\Lambda)$ into blocks, called **right U-blocks**, which satisfy the following conditions:

(B2) If two states $\sigma_{h_{j+1}}$ and $\sigma'_{h_{j+1}}$ in $\Sigma_{h_{j+1}}(\Lambda)$ are in the same right U-block, then they have the same set of converging composite branches, i.e.,

$$\{L(\sigma_{h_j},\sigma_{h_{j+1}}) : \sigma_{h_j} \in \Sigma_{h_j}(\Lambda)\} = \{L(\sigma_{h_j},\sigma'_{h_{j+1}}) : \sigma_{h_j} \in \Sigma_{h_j}(\Lambda)\} \quad (6.39)$$

and otherwise

$$\{L(\sigma_{h_j},\sigma_{h_{j+1}}) : \sigma_{h_j} \in \Sigma_{h_j}(\Lambda)\} \cap \{L(\sigma_{h_j},\sigma'_{h_{j+1}}) : \sigma_{h_j} \in \Sigma_{h_j}(\Lambda)\} = \emptyset. \quad (6.40)$$

Each left U-block (or right U-block) consists of $2^{\nu_{h_j,h_{j+1}}(C)}$ states [44], where

$$\nu_{h_j,h_{j+1}}(C) \triangleq k(p_{0,h_{j+1}}(\tilde{C}_{h_j,h_{j+1}} \cap (C_{0,h_{j+1}} \oplus C_{h_j,N}))) - k(C_{0,h_j}). \quad (6.41)$$

A pair of a left U-block and a right U-block is called a **U-block pair**. It follows from the conditions (B1) and (B2), that each U-block pair $(B_h, B_{h_{j+1}})$ has the following **uniform** properties:

(1) For any two states $\sigma_{h_{j+1}}$ and $\sigma'_{h_{j+1}}$ in $B_{h_{j+1}}$,

$$\{L(\sigma_{h_j},\sigma_{h_{j+1}}) : \sigma_{h_j} \in B_{h_j}\} = \{L(\sigma_{h_j},\sigma'_{h_{j+1}}) : \sigma_{h_j} \in B_{h_j}\}. \quad (6.42)$$

(2) For any two states σ_{h_j} and σ'_{h_j} in B_{h_j},

$$\{L(\sigma_{h_j},\sigma_{h_{j+1}}) : \sigma_{h_{j+1}} \in B_{h_{j+1}}\} = \{L(\sigma'_{h_j},\sigma_{h_{j+1}}) : \sigma_{h_{j+1}} \in B_{h_{j+1}}\}. \quad (6.43)$$

The first property simply says that for a U-block pair $(B_{h_j}, B_{h_{j+1}})$, the set of composite branches from states in the left U-block B_{h_j} converging to any state in the right U-block $B_{h_{j+1}}$ is the same. The second property simply says that the set of composite branches diverging from any state in B_{h_j} to states in $B_{h_{j+1}}$ is the same. Two different U-block pairs have mutually disjoint composite branch sets.