TRELLISES AND TRELLIS-BASED DECODING ALGORITHMS FOR LINEAR BLOCK CODES

Technical Report

to

NASA

Goddard Space Flight Center
Greenbelt, Maryland 20771

Grant Numbers: NAG 5-931
NAG 5-2938

Report Number: 98-003

Principal Investigator: Shu Lin

Department of Electrical Engineering
University of Hawaii at Manoa
2540 Dole Street, Holmes Hall 483
Honolulu, Hawaii 96822

April 20, 1998

TRELLISES AND TRELLIS-BASED DECODING ALGORITHMS FOR LINEAR BLOCK CODES

Part 3

Shu Lin and Marc Fossorier

April 20, 1998

11 A RECURSIVE MAXIMUM LIKELIHOOD DECODING

The Viterbi algorithm is indeed a very simple and efficient method of implementing the maximum likelihood decoding. However, if we take advantage of the structural properties in a trellis section, other efficient trellis-based decoding algorithms can be devised. Recently, an efficient trellis-based **recursive maximum likelihood decoding (RMLD)** algorithm for linear block codes has been proposed [37]. This algorithm is more efficient than the conventional Viterbi algorithm in both computation and hardware requirements. Most importantly, the implementation of this algorithm does not require the construction of the entire code trellis, only some special one-section trellises of relatively small state and branch complexities are needed for constructing path (or branch) metric tables recursively. At the end, there is only one table which contains only the most likely codeword and its metric for a given received sequence $\mathbf{r} = (r_1, r_2, \ldots, r_N)$. This algorithm basically uses the **divide and conquer strategy**. Furthermore, it allows **parallel/pipeline** processing of received sequences to speed up decoding.

11.1 BASIC CONCEPTS

Consider a binary (N, K) linear block code C. Suppose a codeword is transmitted and $r = (r_1, r_2, \ldots, r_N)$ is the received vector at the output of the matched filter of the receiver.

Let T be the minimal trellis diagram for C. Consider the trellis section from time-x to time-y. As shown in Section 6.2, a composite branch between two adjacent states in this trellis section is a coset in $p_{x,y}(C)/C^{\mathrm{tr}}_{x,y}$, and a composite branch may appear many times as shown in (6.13). Using this fact, we can reduce the decoding complexity by just processing the distinct composite branches in each trellis section. To achieve this, we form a table for the metrics of composite branches, which for each coset D in $p_{x,y}(C)/C^{\mathrm{tr}}_{x,y}$, stores the largest metric D, denoted $m(D)$, and the label for the branch with the largest metric, denoted $l(D)$. This table is called the **composite branch metric table**, denoted $\mathrm{CBT}_{x,y}$, for the trellis section between time-x and time-y. Since the set of cosets $p_{0,N}(C)/C^{\mathrm{tr}}_{0,N} = C/C$ consists of C only, the table $\mathrm{CBT}_{0,N}$ contains only the codeword in C that has the largest metric. This is the most likely codeword. The RMLD algorithm is simply an algorithm to construct a composite branch metric table **recursively** from tables for trellis sections of shorter lengths to reduce computational complexity. When the table $\mathrm{CBT}_{0,N}$ is constructed, the decoding is completed and $\mathrm{CBT}_{0,N}$ contains the decoded codeword.

A straightforward method to construct the table $\mathrm{CBT}_{x,y}$ is to compute the metrics of all the vectors in the punctured code $p_{x,y}(C)$, and then find the vector with the largest metric for every coset in $p_{x,y}(C)/C^{\mathrm{tr}}_{x,y}$ by comparing the metrics of vectors in the coset. This method is efficient only when $y - x$ is small and should only be used at the **bottom** (or the **beginning**) of the recursive construction procedure. When $y - x$ is large, $\mathrm{CBT}_{x,y}$ is constructed from $\mathrm{CBT}_{x,z}$ and $\mathrm{CBT}_{z,y}$ for a properly chosen integer z with $x < z < y$.

Therefore, the key part of the RMLD algorithm is to construct the metric table $\mathrm{CBT}_{x,y}$ from tables $\mathrm{CBT}_{x,z}$ and $\mathrm{CBT}_{z,y}$. First we must show that this can be done. For two adjacent states, σ_x and σ_y, with $r_x \in \Sigma_x(C)$ and $\sigma_y \in \Sigma_y(C)$, let

$$\Sigma_z(\sigma_x, \sigma_y) = \{\sigma_z^{(1)}, \sigma_z^{(2)}, \ldots, \sigma_z^{(\mu)}\} \tag{11.1}$$

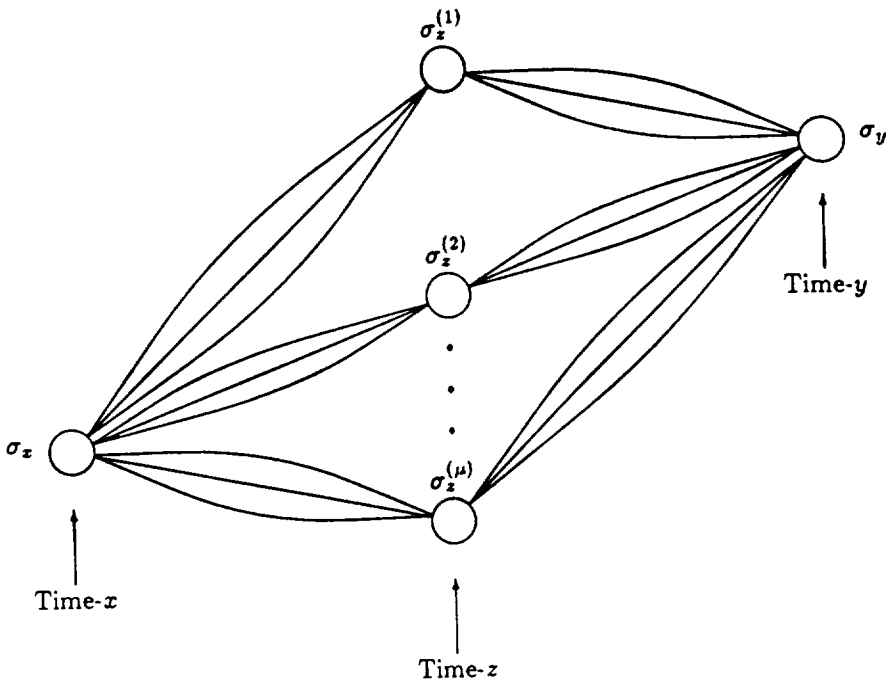

Figure 11.1. Connection between two states.

denote the subset of states in $\Sigma_z(C)$ through which the paths in $L(\sigma_x, \sigma_y)$ connect σ_x to σ_y as shown in Figure 11.1. Then

$$L(\sigma_x, \sigma_y) = \bigcup_{\sigma_z^{(i)} \in \Sigma_z(\sigma_x, \sigma_y)} L(\sigma_x, \sigma_z^{(i)}) \circ L(\sigma_z^{(i)}, \sigma_y). \tag{11.2}$$

It follows from (11.2) and the definitions of metric and label of a coset (or a composite branch) that we have

$$m(L(\sigma_x, \sigma_y)) \triangleq \max_{\sigma_z^{(i)} \in \Sigma_z(\sigma_x, \sigma_y)} \{m(L(\sigma_x, \sigma_z^{(i)})) + m(L(\sigma_z^{(i)}, \sigma_y))\} \tag{11.3}$$

and

$$l(L(\sigma_x, \sigma_y)) = l(L(\sigma_x, \sigma_z^{(i_{\max})})) \circ l(L(\sigma_z^{(i_{\max})}, \sigma_y)), \tag{11.4}$$

where $i_{\max}$ is the index for which the sum in (11.3) takes its maximum.

Note that the metrics, $m(L(\sigma_x, \sigma_z^{(i)}))$ and $m(L(\sigma_z^{(i)}, \sigma_z))$, and labels, $l(L(\sigma_x, \sigma_z^{(i)}))$ and $l(L(\sigma_z^{(i)}, \sigma_y))$, are stored in the composite branch metric tables $\mathrm{CBT}_{x,y}$ and $\mathrm{CBT}_{z,y}$. The state set $\Sigma_z(\sigma_x, \sigma_y)$ can be determined from the code trellis. Therefore, (11.3) and (11.4) show that the composite branch metric table $\mathrm{CBT}_{x,y}$ can be constructed from $\mathrm{CBT}_{z,z}$ and $\mathrm{CBT}_{z,y}$.

Based on the structural properties of a sectionalized trellis, we can readily show that

$$\mu = |\Sigma_z(\sigma_x, \sigma_y)| = 2^{k(C_{x,y}) - k(C_{x,z}) - k(C_{z,y})}. \tag{11.5}$$

This says that if we compute the metric $m(L(\sigma_x, \sigma_y))$ from (11.3) using tables, $\mathrm{CBT}_{x,z}$ and $\mathrm{CBT}_{z,y}$, we need to perform $|\Sigma_z(\sigma_x, \sigma_y)|$ additions and $|\Sigma_z(\sigma_x, \sigma_y)| - 1$ comparisons. However, if we compute the metric $m(L(\sigma_x, \sigma_y))$ directly from the parallel branches in $L(\sigma_x, \sigma_y)$, we need to compute $|C_{x,y}^{\mathrm{tr}}| = 2^{k(C_{x,y})}$ branch metrics and perform $2^{k(C_{x,y})} - 1$ comparisons. For large $y - x$, $|C_{x,y}^{\mathrm{tr}}|$ is much larger than $|\Sigma_z(\sigma_x, \sigma_y)|$ and hence constructing the metric table $\mathrm{CBT}_{x,y}$ from tables, $\mathrm{CBT}_{x,z}$ and $\mathrm{CBT}_{z,y}$, requires much less additions and comparisons than the direct construction of $\mathrm{CBT}_{x,y}$ from vectors in $p_{x,y}(C)$ and cosets in $p_{x,y}(C)/C_{x,y}^{\mathrm{tr}}$. Therefore, recursive construction of composite branch metric tables for trellis sections of longer lengths from tables for trellis sections of shorter lengths reduces decoding computational complexity.

11.2 THE GENERAL ALGORITHM

Now we describe the general framework of the RMLD algorithm for constructing the composite branch metric table $\mathrm{CBT}_{x,y}$ for decoding a received sequence r. We denote this algorithm with $\mathbf{RMLD}(x, y)$. This algorithm uses two procedures, denoted $\mathbf{MakeCBT}(x, y)$ and $\mathbf{CombCBT}(x, y; z)$, which are defined as follows:

- MakeCBT(x, y): construct the table $\mathrm{CBT}_{x,y}$ directly as described later.
- CombCBT$(x, y; z)$: Given tables $\mathrm{CBT}_{x,z}$ and $\mathrm{CBT}_{z,y}$ as inputs, where $x < z < y$, combine these tables to form $\mathrm{CBT}_{x,y}$ as shown in (11.3) and (11.4).

The procedure CombCBT$(x, y; z)$ can be expressed as

$$\mathrm{CombCBT}(\mathrm{RMLD}(x, z), \mathrm{RMLD}(z, y))$$

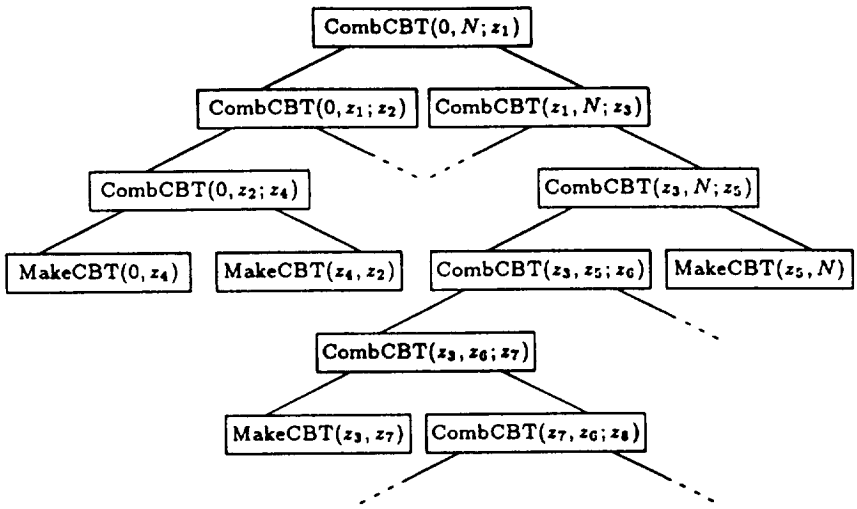

Figure 11.2. Illustration of the recursion process of the RMLD algorithm.

to show its recursive nature.

[Algorithm RMLD(x, y)]
Construct $\mathrm{CBT}_{x,y}$ using the least complex of the following two options:

(1) Execute MakeCBT(x, y), or

(2) Execute CombCBT(RMLD(x, z), RMLD(z, y)), where z with $x < z < y$ is selected to minimize computational complexity.

Decoding is accomplished by executing RMLD$(0, N)$. The recursion process is depicted in Figure 11.2. We see that the RMLD algorithm allows parallel/pipeline processing of received words. This speeds up the decoding process.

The MakeCBT(x, y) procedure is efficient only when $y-x$ is small and should only be used at the **bottom** (or the **beginning**) of the recursive construction procedure. When $y-x$ is large, $\mathrm{CBT}_{x,y}$ is constructed from $\mathrm{CBT}_{x,z}$ and $\mathrm{CBT}_{z,y}$ for a properly chosen z with $x < z < y$. At the bottom of the recursion process, $y - x$ is small and the computation done by the MakeCBT procedure during the entire decoding process is also small. Therefore, the major computation is

carried out by the CombCBT procedure. Hence, the CombCBT procedure is the major procedure in the RMLD algorithm and should be devised to reduce either the total number of computations for software implementation or the circuit requirement and chip size for IC implementation.

In a soft-decision decoding algorithm, addition and comparison operations for metrics are considered as the basic operations. An addition operation and a comparison operation are in general assumed to have equal weight (or cost).

Let $\psi_M(x,y)$ and $\psi_C(x,y;z)$ denote the number of basic operations required to execute the procedure MakeCBT(x,y) and the procedure CombCBT$(x,y;z)$, respectively. The values of $\psi_M(x,y)$ and $\psi_C(x,y;z)$ depend on the implementation of the RMLD algorithm. Assume that the formulas for $\psi_M(x,y)$ and $\psi_C(x,y;z)$ are given. To minimize the overall decoding complexity of the RMLD algorithm, sectionalization of a trellis (choices of z) must be done properly. A sectionalization which gives the smallest overall decoding complexity for given $\psi_M(x,y)$ and $\psi_C(x,y;z)$ is called the **optimum sectionalization** for the code.

Let $\psi_{\min}(x,y)$ denote the smallest number of operations required to construct the table $\mathrm{CBT}_{x,y}$. Then it follows from the algorithm RMLD(x,y) given above that

$$\psi_{\min}(x,y) \triangleq \begin{cases} \psi_M(x,y), & \text{if } x+1=y, \\ \min\left\{\psi_M(x,y), \min\limits_{x<z<y}\{\psi_{R,\min}(x,y;z)\}\right\}, & \text{otherwise,} \end{cases} \tag{11.6}$$

where

$$\psi_{R,\min}(x,y;z) \triangleq \psi_{\min}(x,z) + \psi_{\min}(z,y) + \psi_C(x,y;z). \tag{11.7}$$

The total number of operations required to decode a received word is given by $\psi_{\min}(0,N)$.

By using (11.6) and (11.7) together with formulas for $\psi_M(x,y)$ and $\psi_C(x,y;z)$, we can compute $\psi_{\min}(x,y)$ for every (x,y) with $0 \le x < y \le N$ efficiently in the following way: The values of $\psi_{\min}(x,x+1)$ for $0 \le x < N$ are computed using the given formula for $\psi_M(x,y)$. For an integer i with $0 \le x < x+i \le N$, $\psi_{\min}(x,x+i)$ can be computed from $\psi_{\min}(x',y')$ with $y'-x' < i$ and the given formulas for $\psi_M(x,y)$ and $\psi_C(x,y;z)$. By keeping track of the values of z selected in the above procedure, it is easy to find an optimum sectionalization.

If $\psi_M(x,y)$ and $\psi_C(x,y;z)$ are independent of the received sequence $\boldsymbol{r}$ for any $0 \le x < y \le N$ and $x < z < y$, then the optimum sectionalization can be fixed.

11.3 DIRECT METHODS FOR CONSTRUCTING COMPOSITE BRANCH METRIC TABLES

For two integers x and y such that $0 \le x < y \le N$, a straightforward way to construct the composite branch metric table $\mathrm{CBT}_{x,y}$ directly is to compute the metrics of all the vectors in the punctured code $p_{x,y}(C)$ independently, and then find the vector (branch) with the largest metric for every coset in $p_{x,y}(C)/C^{\mathrm{tr}}_{x,y}$ by comparing the metrics of vectors in the coset. Each surviving vector and its metric are stored in the table $\mathrm{CBT}_{x,y}$. Let **MakeCBT-I**(x,y) denote this procedure.

The number of addition-equivalent operations required to construct the table $\mathrm{CBT}_{x,y}$ by executing MakeCBT-I(x,y), denoted $\psi^{(I)}_M(x,y)$, is given as follows:

$$\psi^{(I)}_M(x,y) = (y-x-1)2^{k(p_{x,y}(C))} + 2^{k(p_{x,y}(C))-k(C_{x,y})}(2^{k(C_{x,y})}-1). \quad (11.8)$$

The first term is the number of additions to compute all the metrics for the vectors in $p_{x,y}(C)$, and the second term is the number of comparisons for finding the vectors with the largest metrics by comparing the metrics of vectors in each coset in $p_{x,y}(C)/C^{\mathrm{tr}}_{x,y}$.

A more efficient method for constructing the table $\mathrm{CBT}_{x,y}$ is to compute the metrics of the 2^{y-x} branch labels following the order of the Gray code as proposed in [60, 102]. Let **MakeCBT-G**(x,y) denote this procedure, where G stands for Gray code. Assume that the bit metric satisfies the following condition: $M(r,0) = -M(r,1)$, where r is a received symbol. This condition holds for the AWGN channel with BPSK transmission and $M(r,1) \triangleq r$. We also assume that the all-one vector of length $y-x$, denoted $\mathbf{1}_{y-x}$, is in $p_{x,y}(C)$ for any x and y with $0 \le x < y \le N$. In this case, the metrics of 2^{y-x-1} labels are computed first in the order of the Gray code, and then the remaining metrics are computed by negating the first 2^{y-x-1} metrics. If $\mathbf{1}_{y-x} \in C^{\mathrm{tr}}_{x,y}$, for any vector in a coset of $C^{\mathrm{tr}}_{x,y}$, the complementary vector is in the same coset. In this case, we can simply discard the branches with negative metrics [60, 102] for finding the largest metric in each coset.

Let $\psi_M^{(G)}(x,y)$ denote the values of $\psi_M(x,y)$ using the MakeCBT-G(x,y). Assume that the negation is costless. Then,

$$\psi_M^{(G)}(x,y) = \begin{cases} 2^{y-x-1} + y - x - 2 + 2^{k(p_{x,y}(C))-k(C_{x,y})}(2^{k(C_{x,y})-1} - 1), \\ \qquad\qquad \text{if } \mathbf{1}_{y-x} \in C_{x,y}^{\text{tr}}, \\ 2^{y-x-1} + y - x - 2 + 2^{k(p_{x,y}(C))-k(C_{x,y})}(2^{k(C_{x,y})} - 1), \\ \qquad\qquad \text{otherwise.} \end{cases} \tag{11.9}$$

For small $y-x$, the dimension of $p_{x,y}(C)$ is close to $y-x$. The computational complexities of both MakeCBT-I and MakeCBT-G procedures are small for small $y-x$. The RMLD algorithm with the MakeCBT-G procedure requires slightly less computational complexity than that with the MakeCBT-I procedure; however, the MakeCBT-I procedure is simpler for IC implementation. Using the MakeCBT-G procedure, the metrics of the first 2^{y-x-1} labels in $p_{x,y}(C)$ must be computed serially, however with the MakeCBT-I procedure, the metrics for all the labels in $p_{x,y}(C)$ can be computed independently in parallel.

11.4 THE COMBCBT PROCEDURE

The CombCBT$(x,y;z)$ procedure simply performs the computation of (11.3) and finds the label of (11.4). It is important to note that in the construction of the metric table $\text{CBT}_{x,y}$, we do not need to compute the metric

$$m(\sigma_x,\sigma_y) \triangleq m(L(\sigma_x,\sigma_y))$$

for every adjacent state pair (σ_x,σ_y). We only need to compute $m(\sigma_x,\sigma_y)$ for those adjacent state pairs for which the paths between each state pair form a distinct coset in $p_{x,y}(C)/C_{x,y}^{\text{tr}}$. Therefore, we only compute the metrics for $2^{k(p_{x,y}(C))-k(C_{x,y})}$ distinct adjacent state pairs between time-x and time-y. This is the key to reduce computational complexity.

In principle we can construct the metric table $\text{CBT}_{x,y}$ using the section of the code trellis T from time-x to time-y as follows:

(i) For each coset $D \in p_{x,y}(C)/C_{x,y}^{\text{tr}}$, identify a state pair (σ_x,σ_y) such that $L(\sigma_x,\sigma_y) = D$;

(ii) Determine the state set $\Sigma_x(\sigma_x,\sigma_y)$; and

(iii) Compute the metric $m(\sigma_x, \sigma_y)$ and the label $l(\sigma_x, \sigma_y)$ from (11.3) and (11.4), respectively.

However for long codes, it is a big effort to construct the large trellis section from time-x to time-y and execute the above steps (i) to (iii). The total number of composite branches in the trellis section between time-x and time-y can be very large and the number of distinct composite branches is only a small fraction. Examining this trellis section can be very time consuming and effort wasting. Consequently, implementation will be complex and costly.

To overcome the complexity problem and facilitate the computation of (11.3), we construct a much simpler special two-section trellis for the punctured code $p_{x,y}(C)$ with section boundary locations in $\{x, z, y\}$ and multiple "final" states at time-y, one for each coset in $p_{x,y}(C)/C^{\text{tr}}_{x,y}$. This special two-section trellis contains only the needed information for constructing the metric table $\text{CBT}_{x,y}$ from $\text{CBT}_{x,z}$ and $\text{CBT}_{z,y}$ (no redundancy). For a coset $D_y \in p_{x,y}(C)/C^{\text{tr}}_{x,y}$, define

$$S_z(D_y) \triangleq \{D_z \in p_{x,z}(C)/C^{\text{tr}}_{x,z} : D_z \subseteq p_{x,z}(D_y)\}, \tag{11.10}$$

where $p_{x,z}(D_y)$ is the truncation of the coset D_y from time-x to time-z. For each $D_z \in S_z(D_y)$, there is exactly one coset in $p_{z,y}(C)/C^{\text{tr}}_{z,y}$, denoted $\text{adj}(D_z, D_y)$, such that $D_z \circ \text{adj}(D_z, D_y) \subseteq D_y$ (see Figure 11.1). Then,

$$D_y = \bigcup_{D_z \in S_z(D_y)} D_z \circ \text{adj}(D_z, D_y). \tag{11.11}$$

From (11.11) we see that the metric of D_y can be computed from metrics of cosets in $p_{x,z}(C)/C^{\text{tr}}_{x,z}$ and cosets in $p_{z,y}(C)/C^{\text{tr}}_{z,y}$ (or from tables $\text{CBT}_{x,z}$ and $\text{CBT}_{z,y}$) once the set $S_z(D_y)$ and $\text{adj}(D_z, D_y)$ for each $D_z \in S_z(D_y)$ are identified. The special two-section trellis to be constructed is simply to display the relationship given by (11.11) and identify the set $S_z(D_y)$ for each coset $D_y \in p_{x,y}(C)/C^{\text{tr}}_{x,y}$.

Let Σ_z and Σ_y denote the state spaces of the special two-section trellis for $p_{x,y}(C)$ at time-z and time-y, respectively. To achieve the purpose as described above, the special two-section trellis for $p_{x,y}(C)$ must have the following structural properties:

(1) There is an initial state, denoted $\sigma_{x,0}$ at time-x.

(2) There is a one-to-one correspondence between the states in the state space Σ_z and the cosets in $p_{x,z}(C)/C^{\text{tr}}_{x,z}$. Let D_z denote a coset in $p_{x,z}(C)/C^{\text{tr}}_{x,z}$ and $\sigma(D_z)$ denote its corresponding state at time-z. Then the composite branch label between $\sigma_{x,0}$ and $\sigma(D_z)$ is $L(\sigma_{x,0}, \sigma(D_z)) = D_z$.

(3) There is a one-to-one correspondence between the states in the state space Σ_y and the cosets $p_{x,y}(C)/C^{\text{tr}}_{x,y}$. Let D_y denote a coset in $p_{x,y}(C)/C^{\text{tr}}_{x,y}$ and $\sigma(D_y)$ denote its corresponding state at time-y. For any state $\sigma(D_y) \in \Sigma_y$, $L(\sigma(D_z), \sigma(D_y)) = \text{adj}(D_z, D_y)$ if $D_z \in S_z(D_y)$. Otherwise, $L(\sigma(D_z), \sigma(D_y)) = \emptyset$.

From the structural properties of the above special two-section trellis, we see that: (1) For every state $\sigma(D_z) \in \Sigma_z$, its (state) metric $m(D_z)$ is given in the table $\text{CBT}_{x,z}$; and (2) For each composite branch between a state $\sigma(D_z)$ at time-z and an adjacent state $\sigma(D_y)$ at time-y, its composite branch metric, $m(\sigma(D_z), \sigma(D_y))$, is given in the table $\text{CBT}_{z,y}$.

It follows from (11.11) and the structural properties of the above special two-section trellis for $p_{x,y}(C)$ that for each coset $D_y \in p_{x,y}(C)/C^{\text{tr}}_{x,y}$, the metric $m(D_y)$ is given by

$$m(D_y) = \max_{D_z \in S_z(D_y)} \{m(D_z) + m(\sigma(D_z), \sigma(D_y))\}, \tag{11.12}$$

where the set of states correspond to $S_z(D_y)$ and the state pairs $(\sigma(D_z), \sigma(D_y))$ can be easily identified from the special two-section trellis. Eq.(11.12) is simply equivalent to (11.3). Therefore, CombCBT$(x, y; z)$ will be designed to compute the metrics for the table $\text{CBT}_{x,y}$ based on (11.12) using the special two-section trellis. In general, this special trellis is much simpler than the section of the entire code trellis T from time-x to time-y except for the cases where $x = 0$ or $y = N$, and is much easier to construct. As a result, the construction of the metric table $\text{CBT}_{x,y}$ is much simpler.

The construction of the above special two-section trellis for $p_{x,y}(C)$ is done as follows: Choose a basis $\{v_1, v_2, \ldots, v_{k(p_{x,y}(C))}\}$ of $p_{x,y}(C)$ such that the first $k(C^{\text{tr}}_{x,y}) = k(C_{x,y})$ vectors form a basis of $C^{\text{tr}}_{x,y}$. Define

$$n_{x,y} \triangleq y - x + k(p_{x,y}(C)) - k(C_{x,y}). \tag{11.13}$$

Let $G(x,y)$ be the following $k(p_{x,y}(C)) \times n_{x,y}$ matrix:

$$G(x,y) = \begin{bmatrix} v_1 & \\ \vdots & 0 \\ v_{k(C_{x,y})} & \\ & \\ v_{k(C_{x,y})+1} & \\ \vdots & I \\ v_{k(p_{x,y}(C))} & \end{bmatrix},$$

where 0 denotes the $k(C_{x,y}) \times (k(p_{x,y}(C)) - k(C_{x,y}))$ all-zero matrix, and I denotes the identity matrix of dimension $(k(p_{x,y}(C)) - k(C_{x,y}))$. Let $C(x,y)$ be the binary linear code of length $n_{x,y}$ generated by $G(x,y)$. Construct a 3-section trellis diagram $T(\{x,z,y,x+n_{x,y}\})$ for $C(x,y)$ with section boundaries at times x, z, y and $x + n_{x,y}$ as shown in Figure 11.3. Then the first two sections of $T(\{x,z,y,x+n_{x,y}\})$ give the desired special two-section trellis for computing (11.12).

In fact from (11.12) and the properties of the special two-section trellis for $p_{x,y}(C)$, we only need the second section of $T(\{x,z,y,x+n_{x,y}\})$ to construct the table $\mathrm{CBT}_{x,y}$. For convenience, we denote this special one-section trellis with $T_x(z,y)$. Table $\mathrm{CBT}_{x,x}$ gives the state metrics of $T_x(z,y)$ at time-z and Table $\mathrm{CBT}_{x,y}$ gives the composite branch metrics of $T_x(z,y)$ between time-z and time-y. Therefore, the implementation of the RMLD algorithm does not require the construction of the code trellis T for the entire code C, it only requires the construction of the special one-section trellises, one for each recursion step. Each of these special one-section trellises has the minimum (state and branch) complexity for constructing a composite branch metric table using the CombCBT procedure. This reduces decoding complexity considerably.

Example 11.1 Consider the RM code $C = \mathrm{RM}_{2,4}$ given in Example 6.3. Let $x = 4$, $y = 12$, and $z = 8$. Then, $p_{x,y}(C) = \mathrm{RM}_{2,3}$, $C^{\mathrm{tr}}_{x,y} = \mathrm{RM}_{1,3}$, $C^{\mathrm{tr}}_{x,z} =$

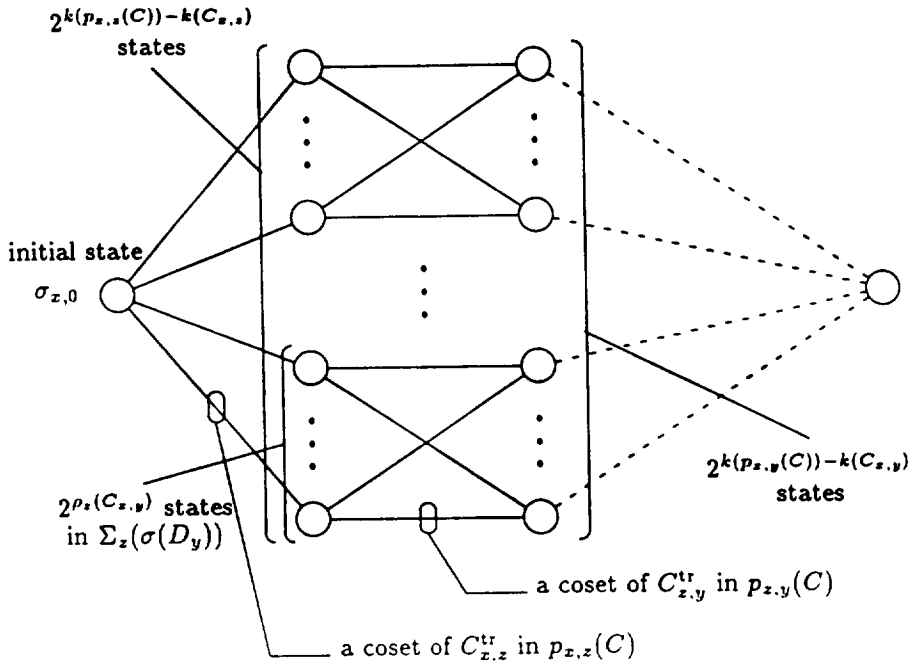

Figure 11.3. Structure of the trellis diagram $T(\{x, z, y, x + n_{x,y}\})$.

$C^{\mathrm{tr}}_{x,y} = \mathrm{RM}_{0,2}$, and $n_{x,y} = 11$. $C(x, y)$ is the (11,9) code generated by

$$G(x,y) = \left[\begin{array}{cccccccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
 & & & & & & & & & & \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1
\end{array}\right].$$

It can be put in trellis oriented form by simple row operations. The one-section minimal trellis diagram, $T_4(8, 12)$, consists of two 4-state parallel components. One of the components is depicted in Figure 11.4. The other can be obtained

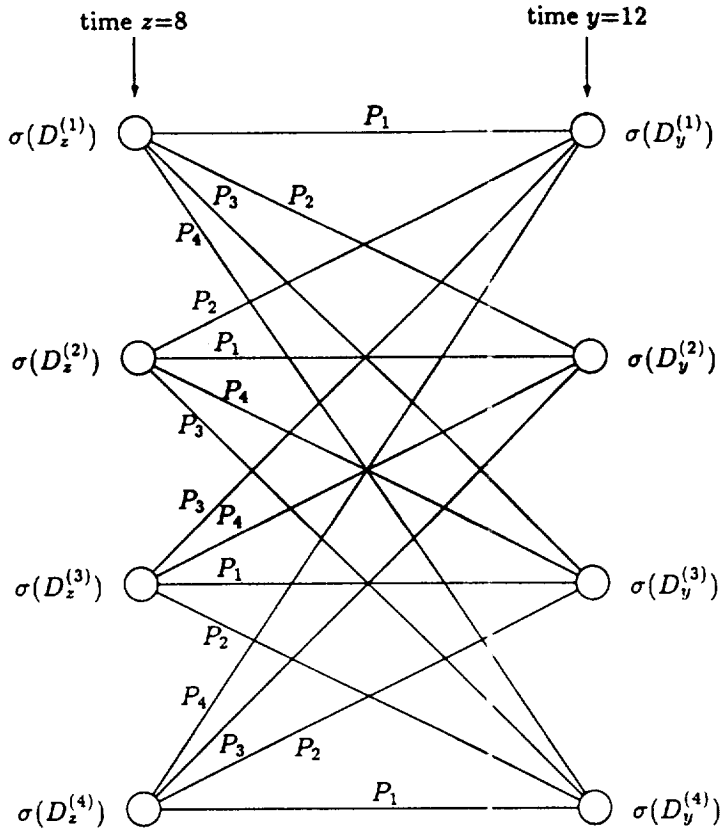

Figure 11.4. A parallel component of $T_4(8, 12)$ for the $RM_{2,4}$ code.

by adding $(0, 0, 0, 1)$ to each branch label.

△△

From (11.12), we see that the computation of the composite branch metric $m(D_y)$ depends on the size of the set $S_z(D_y)$. Since for a coset $D_y \in$

$p_{x,y}(C)/C^{tr}_{x,y}$, the truncation $p_{x,z}(D_y)$ is a union of cosets in $p_{x,z}(C)/C^{tr}_{x,z}$, it follows from property (3) of the special two-section trellis that for every state $\sigma(D_y) \in \Sigma_y$, the number of composite branches merging into the state $\sigma(D_y)$ is

$$\begin{aligned} |S_z(D_y)| &= \frac{|D_y|}{|D_z| \cdot |L(\sigma(D_z), \sigma(D_y))|} = \frac{|C^{tr}_{x,y}|}{|C^{tr}_{x,z}| \cdot |C^{tr}_{z,y}|} \\ &= 2^{k(C_{x,y}) - k(C_{x,z}) - k(C_{z,y})}, \end{aligned} \tag{11.14}$$

which is exactly the same as (11.5). From (11.14), we can readily determine the number of computation operations required to compute $m(D_y)$.

Next we need to devise efficient methods to solve (11.12) using the one-section trellis $T_x(x, y)$ so that either the computational complexity of the CombCBT procedure is reduced or the circuit requirement and chip size of IC implementation of the CombCBT procedure are reduced. Two methods for solving (11.12) will be presented in the next two sections and they result in two specific CombCBT procedures, named the **CombCBT-V** and the **CombCBT-U** procedures.

11.5 COMBCBT-V$(X, Y; Z)$ PROCEDURE

A straightforward procedure to solve (11.12) based on the one-section trellis $T_x(z, y)$ is to apply the conventional add-compare-select (**ACS**) procedure that is used in the conventional Viterbi algorithm. For each coset D_y in $p_{x,y}(C)/C^{tr}_{x,y}$, the metric sum, $m(D_z) + m(\sigma(D_z), \sigma(D_y))$, is computed for every state $\sigma(D_z)$ with $D_z \in S_z(D_y)$, and $m(D_y)$ is found by comparing all the computed metric sums. This procedure is called the CombCBT-V$(x, y; z)$ procedure, where V stands for Viterbi algorithm.

Since the Viterbi algorithm is applied to a one-section trellis diagram to construct a composite branch metric table from two smaller tables, the IC implementation of the CombCBT-V procedure is quite simple and straightforward.

Let $\psi^{(V)}_C(x, y; z)$ denote the value of $\psi_C(x, y; z)$ for the CombCBT-V$(x, y; z)$ procedure. Note that the number of states at time-y in $T_x(z, y)$ is $2^{k(p_{x,y}(C)) - k(C_{x,y})}$, and for each state $\sigma(D_y)$ at time-y, the number of states $\sigma(D_z)$ at time-z in $T_x(z, y)$ which are adjacent to $\sigma(D_y)$ is given by (11.14).

Let

$$\rho_x(C_{x,y}) = k(C_{x,y}) - k(C_{x,x}) - k(C_{x,y}). \tag{11.15}$$

Then, the total number of additions and that of comparisons executed by the CombCBT-V$(x, y; z)$ procedure are

$$2^{k(p_{x,y}(C))-k(C_{x,y})+\rho_x(C_{x,y})} \quad \text{and} \quad 2^{k(p_{x,y}(C))-k(C_{x,y})}(2^{\rho_x(C_{x,y})} - 1),$$

respectively. Consequently, the computational complexity of the CombCBT-V$(x, y; z)$ procedure is given by

$$\psi_C^{(V)}(x, y; z) = 2^{k(p_{x,y}(C))-k(C_{x,y})}(2^{\rho_x(C_{x,y})+1} - 1). \tag{11.16}$$

11.6 RMLD-(I,V) AND RMLD-(G,V) ALGORITHMS

Combining the CombCBT-V procedure with either the MakeCBT-I procedure or the MakeCBT-G procedure, we obtain two specific RMLD algorithms, denoted **RMLD-(I,V)** and **RMLD-(G,V)**. From (11.6), (11.7), (11.8) and (11.16), we can compute the total number of addition-equivalent operations required by the RMLD-(I,V) algorithm for decoding a received word. The computational complexity of the RMLD-(G,V) algorithm can be computed from (11.6), (11.7), (11.9) and (11.16).

For either the RMLD-(I,V) algorithm or the RMLD-(G,V) algorithm, we need to know for what value of $y - x$ that the CombCBT-V procedure should be executed to construct the table $\text{CBT}_{x,y}$. This is answered by the following two theorems. We simply state the theorems here without the proofs which can be found in [111].

Theorem 11.1 Consider a binary linear code C of length N such that the minimum Hamming distances of C and its dual code are both greater than one.

(i) If $y - x > 2$, then for any z with $x < z < y$, the CombCBT-V$(x, y; z)$ procedure requires less computation to form the metric table $\text{CBT}_{x,y}$ than the MakeCBT-I(x, y) procedure. If $y - x = 2$, the complexities of CombCBT-V$(x, y; x + 1)$ and that of MakeCBT-I(x, y) are the same.

(ii) If $y-x > 2$, $k(p_{x,y}(C)) = y-x$ and $C^{\mathrm{tr}}_{x,y} = \{0_{y-x}, (1,*,\ldots,*,1)\}$, where 0_{y-x} denotes the all-zero vector of length $y-x$ and $(1,*,\ldots,*,1)$ denotes a vector of length $y-x$ such that the first and the last components are 1, then the right-hand side of (11.6) takes its minimum for both $z = \lfloor (x+y)/2 \rfloor$ and $z = \lceil (x+y)/2 \rceil$. △△

Theorem 11.1 simply says that for $y - x > 2$, procedure CombCBT-V$(x, y; z)$ should be used to construct the metric table $\mathrm{CBT}_{x,y}$ in the RMLD-(I,V) algorithm.

Theorem 11.2 Consider a binary linear code C of length N such that the minimum Hamming distance of C and its dual code are both greater than one. For the RMLD-(G,V) algorithm,

(i) If $k(p_{x,y}(C)) = y - x$ and $C^{\mathrm{tr}}_{x,y} = \{0\}$ or $\{0_{y-x}, (1,*,\ldots,*,1)\}$, then the MakeCBT-G$(x, y)$ procedure requires less computation than the CBT$(x, y; z)$ procedure for any z with $x < z < y$ to form the metric table $\mathrm{CBT}_{x,y}$ for $y - x > 2$. When $y - x = 2$, they are the same.

(ii) If the conditions of (i) do not hold, then the CombCBT-V$(x, y; z)$ procedure with some z is more efficient than the MakeCBT-G(x, y) for constructing the metric table $\mathrm{CBT}_{x,y}$ for $y - x > 2$. Moreover, if $k(p_{x,y}(C)) < y - x$ and $C^{\mathrm{tr}}_{x,y} = \{0\}$ or $\{0_{y-x}, (1,*,\ldots,*,1)\}$, then the right-hand side of (11.6) takes its minimum for both $z = \lfloor (x+y)/2 \rfloor$ and $z = \lceil (x+y)/2 \rceil$. △△

Since the Viterbi algorithm is applied to a one-section trellis diagram to construct a composite branch metric table from two smaller tables, the IC implementations of both the RMLD-(I,V) and RMLD-(G,V) algorithms are quite simple and straightforward. For high speed decoders, the MakeCBT-I procedure is more suitable than the MakeCBT-G procedure, since branch metrics can be computed in parallel. As shown in Theorem 11.1, in the optimum sectionalization, the value of $y - x$ for the MakeCBT-I procedure to be executed can be kept equal to 2, but this is not necessarily the case for the MakeCBT-G procedure (see Theorem 11.2). Hence, IC implementation of the MakeCBT-I procedure is easier. Furthermore, with the MakeCBT-G procedure, the metrics must be computed serially.

11.7 COMBCBT-U$(X, Y; Z)$ PROCEDURE

This procedure is based on the decomposition of the one-section trellis $T_x(z, y)$ into simple uniform subtrellises as described in Section 6.4. The one-section trellis $T_x(z, y)$ may consist of parallel isomorphic components. These parallel components can be partitioned into groups of the same size in such a way that: (1) two parallel components in the same group are identical up to path labeling; and (2) two parallel components in two different groups do not have any path label in common [44]. Each group consists of 2^λ identical parallel components, where λ can be computed from (6.36) with $C(x, y)$ as the code.

Furthermore, each parallel component of $T_x(z, y)$ can be decomposed into subtrellises with simple uniform structures as shown in Figure 11.5 by applying Theorem 3 of [44] (also see Section 6.3) to the code $C(x, y)$ that was used for constructing the one-section trellis $T_x(z, y)$. Consider a parallel component Λ. The state spaces at the two ends of the parallel component can be partitioned into blocks of the same size 2^ν, called **left U-blocks** and **right U-blocks**, respectively, where ν can be computed from (6.41) with C replaced by $C(x, y)$.

A pair of a left U-block and a right U-block is called a U-block pair, and each U-block pair (B_z, B_y) has the following **uniform structure**, denoted U: For any two states σ_y and σ'_y in B_y,

$$\{L(\sigma_z, \sigma_y) : \sigma_z \in B_z\} = \{L(\sigma_z, \sigma'_y) : \sigma_z \in B_z\}. \tag{11.17}$$

The above property simply says that for a U-block pair (B_z, B_y), the set of composite branches from the states in the left U-block B_z to any state in the right U-block B_y is the same. This property can be used in solving (11.12) to reduce the computational complexity. The label set of composite branches defined by (11.17) is called the **composite branch label set** of the U-block pair (B_z, B_y). Two different U-block pairs have mutually disjoint composite branch label sets.

The CombCBT-U$(x, y; z)$ procedure is devised based on the uniform structure of a U-block pair. In contrast to the CombCBT-V$(x, y; z)$ which solves (11.12) independently for every state $\sigma(D_y)$ with $D_y \in p_{x,y}(C)/C^{\mathrm{tr}}_{x,y}$, CombCBT-U$(x, y; z)$ solves (11.12) simultaneously for each U-block pair (B_z, B_y) of a parallel component of $T_x(z, y)$ by taking into account of the uniform

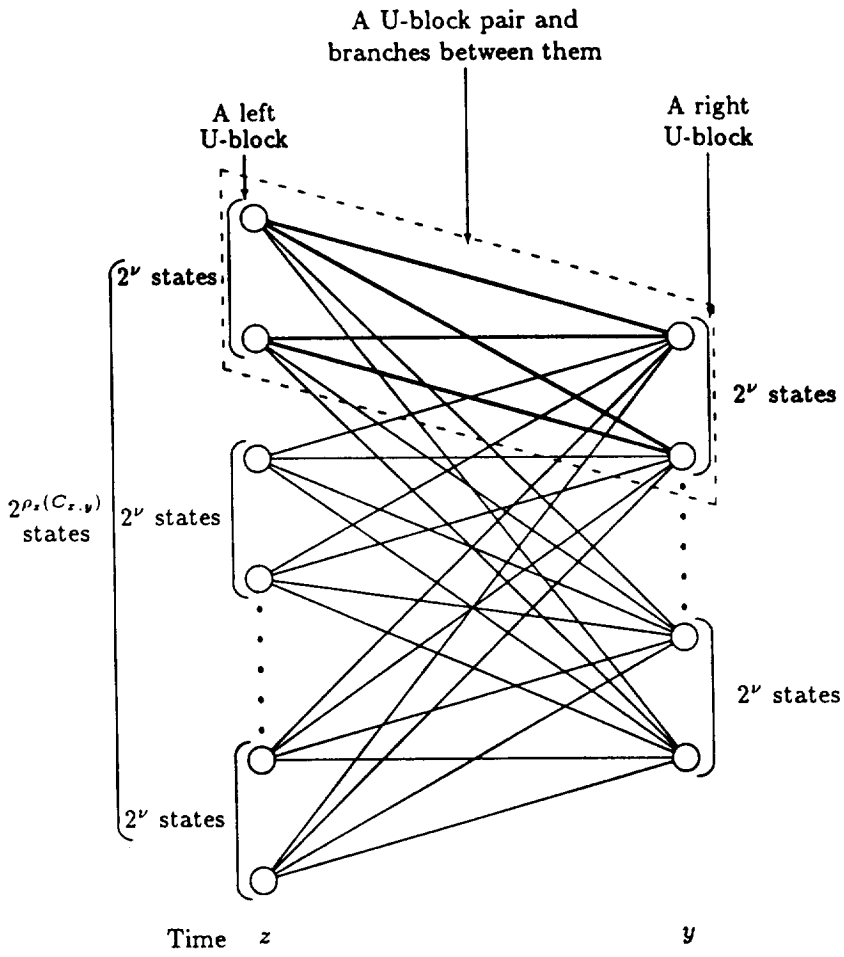

Figure 11.5. The left U-blocks and right U-blocks of a parallel component in $T_x(z, y)$.

property U given by (11.17). For a parallel component Λ of $T_x(z,y)$, let $\mathrm{LU}(\Lambda)$ denote the set of left U-blocks in Λ. Based on (11.17), (11.12) can be put in the following form: For a state $\sigma(D_y)$ in a right U-block B_y,

$$m_{B_z}(D_y) \triangleq \max_{D_z \in \{D_z : \sigma(D_z) \in B_z\}} \{m(D_z) + m(\sigma(D_z), \sigma(D_y))\}, \text{ for } B_z \in \mathrm{LU}(\Lambda), \tag{11.18}$$

$$m(D_y) = \max_{B_z \in LU(\Lambda)} m_{B_z}(D_y). \tag{11.19}$$

Equations (11.18) and (11.19) show that (11.12) can be solved simultaneously for each U-block pair. This allows parallel processing to speed up the computation. In fact the computations of (11.18) and (11.19) can be carried out for all the parallel components of $T_x(z,y)$ in parallel.

For easy understanding, an example is used to explain how to solve (11.18) for each U-block pair (B_z, B_y).

Example 11.2 Again consider the RM code $C = \mathrm{RM}_{2,4}$. As shown in Example 11.1, the one-section trellis diagram $T_x(z,y)$ with $x = 4$, $y = 12$ and $z = 8$ consists of two four-state parallel components. From (6.36) and (6.41), we find that $\lambda = 0$ and $\nu = 2$. Therefore, the two parallel components are not identical, and each consists of only one left U-block and one right U-block. As shown in Figure 11.4, the four end states of one parallel component at time-8, denoted $\sigma(D_z^{(1)})$, $\sigma(D_z^{(2)})$, $\sigma(D_z^{(3)})$, $\sigma(D_z^{(4)})$, form a single left U-block, and the 4 end states at time-12, denoted $\sigma(D_y^{(1)})$, $\sigma(D_y^{(2)})$, $\sigma(D_y^{(3)})$, $\sigma(D_y^{(4)})$, form a single right U-block. There are four different composite branch labels between them, denoted

$$P_1 \triangleq \{0000, 1111\}, \quad P_2 \triangleq \{0011, 1100\},$$
$$P_3 \triangleq \{0101, 1010\}, \quad P_4 \triangleq \{0110, 1001\}.$$

The set of the composite branch labels merging into any state $\sigma(D_y^{(j)})$ at time-12 is $\{P_1, P_2, P_3, P_4\}$.

From (11.18) and (11.19), the largest metric, denoted $m(D_y^{(j)})$, for the coset $D_y^{(j)}$ with $1 \le j \le 4$ is given by

$$m(D_y^{(j)}) = \max_{1 \le i \le 4} \{m(D_z^{(i)}) + m(P_{b(i,j)})\}, \tag{11.20}$$

where $b(i,j)$ is the unique integer such that $L(\sigma(D_z^{(i)}),\sigma(D_y^{(j)})) = P_{b(i,j)}$.

To compute the metrics $m(D_y^{(j)})$ for $1 \le j \le 4$, form the following set of metric sums:

$$M \triangleq \{m(D_z^{(i)}) + m(P_b) : 1 \le i \le 4, 1 \le b \le 4\}.$$

Each sum in M is associated with a path in the trellis of Figure 11.4. Clearly the largest sum in M corresponds to the survivor path for the associated state $\sigma(D_y^{(q)})$ at time-y., i.e., $m(D_y^{(q)})$ is equal to the largest sum in M. Thus this value for the coset $D_y^{(q)}$ is entered in Table $\mathrm{CBT}_{z,y}$. This can be proceeded by examining the second, third, ... largest sum in M. If the j-th largest sum M_j corresponds to state $\sigma(D_y^{(q')})$, and $\mathrm{CBT}_{z,y}$ contains no entry for the coset $D_y^{(q')}$, then M_j is entered in $\mathrm{CBT}_{z,y}$. This process continues until $\mathrm{CBT}_{z,y}$ contains entries for each $D_y^{(j)}$.

Similarly, the metrics of cosets that correspond to the four states at time-12 in the other parallel component in the one-section trellis $T_z(z,y)$ can be computed. This completes the construction of table $\mathrm{CBT}_{z,y}$.

We can find the j-th largest sum of M more efficiently by pre-sorting

$$\{m(D_z^{(i)}) : 1 \le i \le 4\} \quad \text{and} \quad \{m(P_b) : 1 \le b \le 4\}.$$

$\triangle\triangle$

In general, for a U-block pair (B_z, B_y) with $B_z = \{\sigma(D_z^{(1)}),\sigma(D_z^{(2)}),\ldots,\sigma(D_z^{(2^\nu)})\}$, $B_y = \{\sigma(D_y^{(1)}),\sigma(D_y^{(2)}),\ldots,\sigma(D_y^{(2^\nu)})\}$, and the composite branch label set of (B_z, B_y), $\{P_1, P_2, \ldots, P_{2^\nu}\}$, (11.18) is solved in the following way:

(S1) Sort $m(D_z^{(1)})$, $m(D_z^{(2)})$, ..., $m(D_z^{(2^\nu)})$ in the decreasing order.

(S2) Sort $m(P_1)$, $m(P_2)$, ..., $m(P_{2^\nu})$ in the decreasing order.

(S3) Form $M \triangleq \{m(D_z^{(i)}) + m(P_b) : 1 \le i \le 2^\nu, 1 \le b \le 2^\nu\}$. Determine $m_{B_z}(D_y^{(j)})$ with $1 \le j \le 2^\nu$ as described in Example 11.2 by using the following partial ordering on M:

$$m(D_z^{(i)}) + m(P_b) \;\ge\; m(D_z^{(i')}) + m(P_{b'}),$$
$$\text{if } m(D_z^{(i)}) \ge m(D_z^{(i')}) \text{ and } m(P_b) \ge m(P_{b'}).$$

Clearly the above procedure for a U-block pair can be executed for all the U-block pairs in all the distinct parallel components in the one-section trellis $T_x(z, y)$ simultaneously.

Note that the CombCBT-U$(x, y; z)$ procedure is identical to the CombCBT-V$(x, y; z)$ procedure only for the case of $\nu = 0$ (the trivial case in which each left U-block and right U-block consist of a single state).

Let $\psi_C^{(U)}(x, y; z)$ denote the number of addition-equivalent operations of CombCBT-U$(x, y; z)$. The computational complexity for solving (11.18) depends on the received sequence. In the following, an upper bound on $\psi_C^{(U)}(x, y; z)$ for the worst case is given, which is independent of the received sequence. Without derivation, the bound is given below [37]:

$$\begin{aligned}\psi_C^{(U)}(x, y; z) \leq\ & (2^{k(p_{x,z}(C))-k(C_{x,z})-\nu} + 2^{k(p_{x,y}(C))-k(C_{x,y})-\nu})\eta(2^\nu) \\ & + 2^{k(p_{x,y}(C))-k(C_{x,y})}((1+\frac{1}{2^\nu})2^{\rho_z(C_{x,y})} - 1) \qquad (11.21)\end{aligned}$$

where

$$\eta(2^\nu) = \begin{cases} \nu, & \text{for } \nu = 0, 1, \\ 2(\nu-1)(2^\nu - 1) - 1, & \text{otherwise.} \end{cases} \qquad (11.22)$$

Let $\bar{\psi}_C^{(U)}(x, y; z)$ denote the upper bound given by the right-hand side of (11.21). It can be shown that [37]

$$\bar{\psi}_C^{(U)}(x, y; z) \leq \psi_C^{(V)}(x, y; z). \qquad (11.23)$$

The inequality of (11.23) holds for $\nu \geq 2$. As ν becomes large, the ratio

$$\bar{\psi}_C^{(U)}(x, y; z)/\psi_C^{(V)}(x, y; z)$$

decreases rapidly. This says that the CombCBT-U procedure is more efficient than the CombCBT-V procedure computation-wise.

11.8 RMLD-(G,U) ALGORITHM

The CombCBT-U procedure can be combined with either the MakeCBT-I procedure or the MakeCBT-G procedure to form specific RMLD algorithms. Since the MakeCBT-G procedure requires less computational complexity than the MakeCBT-I procedure. The MakeCBT-G procedure and CombCBT-U procedure are combined to form an RMLD algorithm, called the RMLD-(G,U) algorithm.

Table 11.1. Numbers of addition-equivalent operations with various maximum likelihood decoding algorithms for some RM and extended BCH codes of length 64.

Code, (Basis)	64-section	RMLD-(I,V)	RMLD-(G,V)	RMLD-(G,U) upper bound $\psi_{\min}^{(G,U)}$	Lafourcade & Vardy [60]
$RM_{2,6}(64,22)$	425,209	78,209	77,896	66,824	101,786
$RM_{3,6}(64,42)$	773,881	326,017	323,759	210,671	538,799
$RM_{4,6}(64,57)$	7,529	5,281	4,999	4,087	6,507
EBCH(64,10), (C)	20,073	3,201	3,108	3,108	4,074
EBCH(64,16), (B)	764,153	120,193	119,880	96,840	148,566
EBCH(64,18), (B)	2,865,401	468,040	468,040	372,808	509,120
EBCH(64,24), (B)	1,327,353	271,745	271,432	171,823	316,608
EBCH(64,30), (C)	35,028,985	16,091,009	16,056,668	9,408,567	16,598,063
EBCH(64,36), (C)	18,710,521	9,995,617	9,961,580	7,684,276	12,829,263
EBCH(64,39), (C)	38,436,857	24,741,161	24,707,149	19,841,161	30,982,731
EBCH(64,45), (C)	1,082,105	893,489	891,695	665,713	891,819
EBCH(64,51), (A)	418,553	312,721	312,382	257,300	393,528

From (11.6), (11.7), (11.9) and (11.21), we can compute an upper bound, denoted $\psi_{\min}^{(G,U)}$, on the worst-case computational complexity of the RMLD-(G,U) algorithm for decoding a received word.

11.9 COMPARISONS

Among the three specific RMLD algorithms, RMLD-(I,V), RMLD-(G,V) and RMLD-(G,U), the RMLD-(G,U) algorithm is the most efficient one computation-wise, while the RMLD-(I,V) algorithm is the simplest for IC implementation.

In the following, the three specific RMLD algorithms are applied to some well known codes of length 64 to show their effectiveness in terms of computational complexity.

Let EBCH$(64,k)$ denote the extended code obtained from the binary primitive $(63,k)$ BCH code by adding an overall parity bit. The computational complexities of decoding the $RM_{r,6}$ codes with $2 \leq r \leq 4$ and the permuted EBCH$(64,k)$ codes with $10 \leq k \leq 51$ are computed based on certain symbol

position permutations and given in Table 11.1. Hereafter, $\mathrm{RM}_{r,m}$ is denoted $\mathrm{RM}_{r,m}(2^m, \sum_{i=0}^{r}\binom{m}{i})$ to show the number of information bits explicitly.

To reduce the state complexities of the trellis diagrams for EBCH$(64, k)$ codes, the order of symbol positions must be permuted. For the RM codes, the natural symbol ordering is optimal for the state complexity [45]. For the EBCH codes, consider the following permutations π [46]. Let α be a primitive element of GF(2^6) and $\{\beta_1, \ldots, \beta_6\}$ a basis of GF(2^6) over GF(2). For a positive integer i less than 2^6, let α^{i-1} be expressed as $\alpha^{i-1} = \sum_{j=1}^{6} b_{i,j}\beta_j$, with $b_{i,j} \in$ GF(2). For $i = 0$, let $b_{0,j} \triangleq 0$ for $1 \leq j \leq 6$. Then π is the following permutation on $\{1, 2, \ldots, 2^6\}$, $\pi(i) \triangleq 1 + \sum_{j=1}^{6} b_{i-1,j}2^{6-j}$, for $1 \leq i \leq 2^6$. Consider the following three bases for codes of length 64: (1) Basis A is the polynomial basis, $\{1, \alpha, \alpha^2, \ldots, \alpha^5\}$; (2) Basis B is $\{1, \alpha, \alpha^2, \alpha^{21}, \alpha^{22}, \alpha^{23}\}$, which is obtained by combining a basis of GF(2^6) over GF(2^2), $\{1, \alpha, \alpha^2\}$, and a basis of GF(2^2) over GF(2), $\{1, \alpha^{21}\}$; (3) Basis C is $\{1, \alpha, \alpha^9, \alpha^{10}, \alpha^{18}, \alpha^{19}\}$, which is obtained by combining a basis of GF(2^6) over GF(2^3), $\{1, \alpha\}$, and a basis of GF(2^3) over GF(2), $\{1, \alpha^9, \alpha^{18}\}$.

Table 11.1 gives the total numbers of addition-equivalent operations required by the three specific RMLD algorithms, RMLD-(I,V), RMLD-(G,V) and RMLD-(G,U), for decoding the above codes. For the RMLD-(G,U) algorithm, only the values of the upper bound $\psi_{\min}^{(G,U)}$ on the worst-case computational complexity are given. For comparison purpose, the numbers of addition-equivalent operations required for decoding the above codes with the Viterbi decoding algorithm based on optimum sectionalization presented in Section 10.2 (Lafourcade and Vardy algorithm) are also included. The column labeled 64-section gives the numbers of operations required in the conventional Viterbi decoding based on the bit-level 64-section minimal trellis diagram.

For all EBCH codes, other than the EBCH(64, 51) code, the symbol permutations indicated in Table 11.1 give the smallest optimum values for each column among the three symbol permutations given above. For EBCH(64,51), Basis B gives the smallest number of operations required in the conventional Viterbi decoding based on the 64-section trellis diagram among the three permutations, but Basis A gives the smallest values for the other columns. This shows that a good bit ordering for the N-section trellis diagram is not always good for the

proposed RMLD decoding procedures. The last column in Table 11.1 shows the numbers of addition-equivalent operations given by Lafourcade and Vardy [60].

Table 11.1 shows that the RMLD-(G,U) algorithm is the most efficient trellis-based decoding algorithm even in terms of the worst case computational complexity, and the difference between the computational complexities of the RMLD-(I,V) and RMLD-(G,V) algorithms is very small. All three RMLD algorithms are more efficient than the Viterbi decoding algorithm based on optimum sectionalization [60], except only for the RMLD-(I,V) algorithm for the EBCH(64, 45) code. For each algorithm, the number of basic operations executed by the MakeCBT procedure is relatively small compared with that executed by the CombCBT procedure. Consider the EBCH(64, 45) code. Decoding this code with the RMLD-(I,V) algorithm, the number of basic operations executed by the MakeCBT-I procedure is 108 out of a total of 893,489 basic operations. Using the RMLD-(G,U) algorithm, the MakeCBT-G procedure executes 818 basic operations out of a total 665, 713 basic operations.

Let $< x, y >$ denote the MakeCBT-G(x, y) operation, and let $\cdot$ denote the CombCBT-U operation. The optimum trellis sectionalizations for $\mathrm{RM}_{3,6}(64, 42)$ and EBCH(64, 24) with Basis B for the complexity measure $\bar{\psi}_{\min}^{(G,U)}$ of the RMLD-(G,U) algorithm are identical, and represented as

$$((< 0, 8 > \cdot < 8, 16 >) \cdot (< 16, 24 > \cdot < 24, 32 >)) \cdot ((< 32, 40 > \cdot < 40, 48 >) \cdot (< 48, 56 > \cdot < 56, 64 >)).$$

The optimum trellis sectionalization for $\mathrm{RM}_{2,6}(64, 22)$ for $\bar{\psi}_{\min}^{(G,U)}$ is

$$((((((< 0, 8 > \cdot < 8, 16 >) \cdot (< 16, 24 > \cdot < 24, 32 >)) \cdot (< 32, 40 > \cdot < 40, 48 >)) \cdot < 48, 56 >) \cdot < 56, 61 >) \cdot < 61, 63 >) \cdot < 63, 64 >,$$

and that for EBCH(64, 30) with Basis C is

$$(< 0, 16 > \cdot < 16, 32 >) \cdot (< 32, 48 > \cdot < 48, 64 >).$$

The optimum trellis sectionalization for a code using the algorithm RMLD is generally not unique. The above optimum trellis sectionalizations are chosen in the following manner: If $\psi_{\min}(x, y) = \psi_M^{(G)}(x, y)$, then execute the

Table 11.2. Average numbers of addition-equivalent operations using the RMLD-(G,U) algorithm for the $RM_{3,6}(64,42)$ and EBCH(64, 24) codes.

$RM_{3,6}(64,42)$	RMLD-(G,U)	Upper bound $\psi_{\min}^{(G,U)}$ on the worst case complexity		210,671
		The average number of operations $\psi_{\min}^{(G,U)}$	1dB	66,722
			4dB	66,016
			7dB	63,573
			10dB	61,724
EBCH(64,24), Basis B	RMLD-(G,U)	Upper bound $\psi_{\min}^{(G,U)}$ on the worst case complexity		171,823
		The average number of operations $\psi_{\min}^{(G,U)}$	1dB	70,676
			4dB	70,420
			7dB	69,325
			10dB	68,158

MakeCBT-G(x,y) procedure. Otherwise, the CombCBT-U$(x,y;z)$ procedure is executed for an integer z such that $\psi_{\min}(x,y) = \psi_{\min}(x,z) + \psi_{\min}(z,y) + \psi_C^{(U)}(x,y;z)$ and $|z-(x+y)/2|$ are the smallest.

Using the trellis sectionalizations which are optimum with respect to the measure $\bar{\psi}_{\min}^{(G,U)}$, we evaluate the average values of $\psi_{\min}(0,N)$, denoted $\psi_{\min}^{(G,U)}$, for the RMLD-(G,U) algorithm which are given in Table 11.2. It is assumed that BPSK modulation is used on an AWGN channel. The average values at the SNRs per information bit, 1, 4, 7 and 10 (dB), are listed in the rows labeled $\psi_{\min}^{(G,U)}$ for $RM_{3,6}(64,42)$ and EBCH(64, 24). We see that these values vary only slightly and are much smaller than the worst-case upper bound $\bar{\psi}_{\min}^{(G,U)}$.