NAIST-IS-DT9861010

Doctor's Thesis

Efficient MAP Decoding Algorithms for Linear Block Codes

Ryujiro Shibuya

December 14, 2000

Department of Information Processing
Graduate School of Information Science
Nara Institute of Science and Technology

Doctor's Thesis
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
DOCTOR of ENGINEERING

Ryujiro Shibuya

Thesis committee:	Hiroyuki Seki, Professor
	Minoru Ito, Professor
	Toru Fujiwara, Professor
	Yuichi Kaji, Associate Professor

Efficient MAP Decoding Algorithms for Linear Block Codes[*]

Ryujiro Shibuya

Abstract

Efficient algorithms for the *maximum a posteriori* (*MAP*) decoding of error correcting codes are presented. The MAP decoding plays an essential role in some iterative decoding algorithms including the turbo decoding. Since a MAP decoder is used iteratively in such algorithms, small difference of the efficiency of the MAP decoder may affect the total efficiency of the iterative decoding algorithm much more than we can expect. An efficient algorithm for the MAP decoding contributes for the realization of high-speed and high-reliability communication systems. In this thesis, two algorithms for the MAP decoding are presented. The first algorithm, named *recursive-MAP (rMAP) algorithm*, is devised based on the structural properties of linear block codes. The algorithm performs the MAP decoding in a divide-and-conquer manner. The rMAP algorithm has many implementation advantages compared to the BCJR algorithm, which is the most widely known algorithm for the MAP decoding. For example, the rMAP algorithm is suitable for the parallel and pipeline processing, and the construction of a trellis diagram is not necessary. The latter property contributes to reduce the decoding complexity of the algorithm. Actually, it is shown that

[*]Doctor's Thesis, Department of Information Processing, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DT9861010, December 14, 2000.

the rMAP algorithm is more efficient than the BCJR algorithm if the code rate is low. The second algorithm proposed in this thesis is a “hybrid” of the rMAP and the conventional BCJR algorithm. To perform the MAP decoding, the algorithm uses a section trellis diagram instead of the usual entire trellis diagram. A section trellis diagram has simpler structure than the usual trellis diagram, and the construction of the former is much more efficient and easier than the construction of the latter. A BCJR-like algorithm is executed for the section trellis diagram, where branch metrics of the trellis are computed by the rMAP algorithm. The decoding complexity of the algorithm depends on how the sections of the section trellis are chosen. The relation between the sectionalization of the trellis and the complexity of the algorithm is investigated in detail, and a systematic way to find the optimum sectionalization which minimizes the decoding complexity is presented.

Keywords:

error-correcting codes, MAP decoding, trellis diagram, BCJR algorithm, recursive decoding algorithm

Acknowledgements

I am grateful to Professor Hiroyuki Seki for being my adviser and mentor. His technical and general guidance has supported me in my personal and professional growth.

I would like to thank Professor Minoru Ito for providing me with valuable and beneficial suggestions in this research. The technical guidance and constant encouragement of Professor Toru Fujiwara throughout my study and research on coding theory and its applications are gratefully acknowledged.

I wish to thank Associate Professor Yuichi Kaji for continuous guidance, constructive criticisms and numerous suggestions. In addition, his inspiring thoughts have helped me in conducting my research.

Also I would like to thank Emeritus Professor Tadao Kasami, Visiting Professor Dee Worman, Professor Shu Lin of University of California, Professor Toyoo Takata of Iwate Prefectural University, Doctor Hajime Watanabe of Electrotechnical Laboratory, Associate Professor Ken-ichi Tomita of Shiga University and Associate Professor Marc Fossorier of University of Hawaii, since their supports have been always so helpful and really essential for my work.

Finally, I wish to express my gratitude to all members of Seki Laboratory, all members of Kasami Laboratory and other friends in various positions, for worthwhile discussion and help.

List of Publications

1 Publications Related to the Thesis

1.1 Journal Papers

1. Yuichi Kaji, Ryujiro Shibuya, Toru Fujiwara, Tadao Kasami and Shu Lin: "MAP and LogMAP Decoding Algorithms for Linear Block Codes Using a Code Structure," IEICE Transactions on Fundamentals of Electronics, Communications and Computer Science, vol. E83-A, no.10, pp.1884–1890, October 2000.

2. Ryujiro Shibuya and Yuichi Kaji: "An Efficient MAP Decoding Algorithm which Uses the BCJR and the Recursive Techniques," submitted to IEICE Transactions on Fundamentals of Electronics, Communications and Computer Science.

1.2 Reviewed International Conferences

3. Ryujiro Shibuya, Yuichi Kaji, Toru Fujiwara, Tadao Kasami and Shu Lin: "Recursive Algorithm for Efficient MAP Decoding of Binary Linear Block Codes," Proceedings of the 1998 International Symposium on Information Theory and Its Applications, Mexico City, Mexico, pp.639–642, October

1998.

4. Ryujiro Shibuya, Yuichi Kaji and Tadao Kasami: "An Efficient MAP Decoding Algorithm Using a Section Trellis Diagram," Proceedings of the 2000 IEEE International Symposium on Information Theory, Sorrento, Italy, p.424, June 2000.

1.3 Workshops

5. Ryujiro Shibuya and Yuichi Kaji: "An Efficient MAP Decoding Algorithm Using a Section Trellis Diagram," Technical Report of IEICE, IT99-14, Institute of Electronics, Information and Communication Engineers, May 1999.

6. Ryujiro Shibuya and Yuichi Kaji: "Optimum Sectionalization of a Trellis Diagram for the Hybrid MAP Algorithm," Proceedings of the 23rd Symposium on Information Theory and Its Applications, pp.255–258, October 2000.

2 Other Publications

7. Ryujiro Shibuya, Yuichi Kaji and Tadao Kasami: "Scheme of Visual Secret Sharing as Digital Watermark," Technical Report of IEICE, ISEC96-16, Institute of Electronics, Information and Communication Engineers, June 1996.

8. Ryujiro Shibuya, Yuichi Kaji and Tadao Kasami: "Extension of Digital Watermark Scheme that Uses Visual Secret Sharing," 1997 Symposium on Cryptography and Information Security, 26E, January 1997.

9. Ryujiro Shibuya, Yuichi Kaji and Tadao Kasami: "Digital Watermark for PostScript and PDF Documents," 1998 Symposium on Cryptography and Information Security, 9.2.E, January 1998.

Contents

Chapter 1

Introduction

Coding theory treats *source coding* and *channel coding.* Source coding is important to manage digital data efficiently, and channel coding is essential for digital communication to obtain reliable communication channels. We focus on the channel coding in this thesis.

For several decades, *error control(/correcting) codes* have been investigated. Error control codes made it possible to detect or, moreover, remove the errors occurred on the *codewords* through the communication channel. *Coding* is that transforming messages into codewords, while *decoding* is that transforming codewords into messages. Error detecting or error correcting is done in the decoding phase.

Two general idea of (error correcting) decoding have been established. One is *maximum likelihood decoding* (*MLD*) and the other is *maximum a posteriori (MAP) decoding.* MAP decoding is considered more effective than MLD when the *a priori* probabilities of transmitted codewords are known to the receiver in advance, and the decoding ability of MAP decoding is similar to MLD if those a priori probabilities are unknown to the receiver.

Since the a priori probabilities of transmitted codewords are usually unknown to the receiver, MAP decoding had not been very popular for a long time. However, the MAP decoding is widely noticed nowadays, because it plays an essential role in iterative decoding schemes such as *turbo decoding*[2], which is found experimentally and indicates very good decoding parameters. Actually, for example, turbo codes are decoded by iteratively executing a MAP decoding algorithm many times. Therefore the complexity of a MAP decoding algorithm is significant for realization of efficient turbo decoders.

The first MAP decoding algorithm was independently proposed by Bahl et al.[1] and McAdam et al.[17], which are known as the *BCJR algorithm* nowadays. Unfortunately, the BCJR algorithm is not appropriate for efficient turbo decoders, mainly by two reasons. The first reason of the inappropriateness is that we must construct the entire *trellis diagram* of the code to use the BCJR algorithm. The construction and implementation of the entire trellis diagram are both time and space consuming especially for long practical codes. The second reason is that the BCJR algorithm causes long decoding delay. The BCJR algorithm computes some probabilities (β explained in Chapter 4) by using backward-recursion after the decoder has received entire vector from the matched filter. Therefore the decoding delay can be larger for longer codes. This property is critical for realization of efficient turbo decoders since turbo codes usually use very long constituent codes to improve the error performance.

Efforts have been made to reduce the decoding complexity of the BCJR algorithm. One of such attempts includes a suboptimum realization of the BCJR algorithm. For example, the Log-MAP (MAX-Log-MAP) algorithm and the SOVA (Soft-Output Viterbi Algorithm)[11] use log-likelihood ratios and some approximations to avoid calculating the actual probabilities, and simplify some computations. Franz and Anderson suggest to omit some insignificant computations in the BCJR algorithm to reduce the complexity[8]. These approximation algorithms indeed have smaller complexity than the BCJR algorithm, though, their error performance is not as good as that of the BCJR algorithm. Furthermore, the construction of the entire trellis diagram is still necessary in these method. Another approach for reducing the complexity is to investigate other algorithms for the MAP decoding. Along this approach, the *recursive-MAP (rMAP) algorithm* is proposed in Chapter 5, and another *hybrid algorithm* of the BCJR and the rMAP algorithm is proposed in Chapter6.

Based on the structural properties of linear block codes, the rMAP algorithm executes decoding in the divide-and-conquer manner, which results in many implementation advantages. For example, the decoding complexity (measured by the number of multiplications of probabilities) is significantly reduced compared to the conventional BCJR algorithm especially for low-rate codes. Simulation results show that, the complexity of the rMAP algorithm is less than one-tenth of that of the BCJR algorithm for some well-known codes. Other advantages of the algorithm are also discussed in Chapter 5.

The rMAP algorithm is much more efficient than the BCJR algorithm for low-rate codes, but is also studied to be less efficient than the BCJR algorithm for high-rate codes. Mainly, for improvement of this defect, a hybrid algorithm for the MAP decoding is investigated. The algorithm proposed in Chapter6 is, intuitively, a hybrid algorithm of the rMAP and the BCJR algorithms. The proposed hybrid algorithm uses a *section trellis diagram* of the code instead of the entire trellis diagram. The construction and implementation of the section trellis diagram is much more easier than those of the usual trellis diagram. The (time-)complexity of hybrid algorithm is no wronger than the BCJR and the rMAP algorithms, and it consumes less space for decoding calculation than the BCJR algorithm. It is important problem to prepare the best-sectionalized section trellis diagram to have the least computational complexity. In Chapter6, the algorithm to have such a sectionalization is also proposed.

It should be remarked that, different from the other approximation approaches such as the Log-MAP and SOVA, the proposed algorithms in this thesis is equivalent to the BCJR algorithm in the sense that the output of the proposed algorithm is completely the same as that of the BCJR algorithm. The only difference is that the proposed ones are much more efficient and suitable for implementation than the BCJR algorithm.

Chapter 2

Preliminaries

In this chapter, some preliminaries and ideas of coding and decoding are briefly introduced, to help the easier comprehension of the essence of this thesis.

2.1 Transmission Model

Codes are used for efficient and reliable digital communication or information storage. Information (data) is transmitted into *codewords* by a certain rule and then the codewords will be restored later. The set of codewords and/or the rule of generating codewords is the *code*.

Figure 2.1 shows the simplified transmission model of coding system. A message block represented by k-tuple $\boldsymbol{u} = (u_1, \ldots, u_k)$ from data source is called a *message*. The *encoder* transforms a message $\boldsymbol{u}$ into n-tuple $\boldsymbol{v} = (v_1, \ldots, v_n)$ which is called a codeword. The ration $R = k/n$ is called *code rate*. A codeword $\boldsymbol{v}$ transmitted through the *channel* (or *storage medium*) is called *received sequence* $\boldsymbol{r}$; it would be disturbed by various kinds of noise. The *decoder* transforms the received sequence $\boldsymbol{r}$ into a sequence called *estimate sequence* $\boldsymbol{u}'$. It is required for decoders to decode $\boldsymbol{r}$ into $\boldsymbol{u}$ if possible. Decoding strategy is basically based on

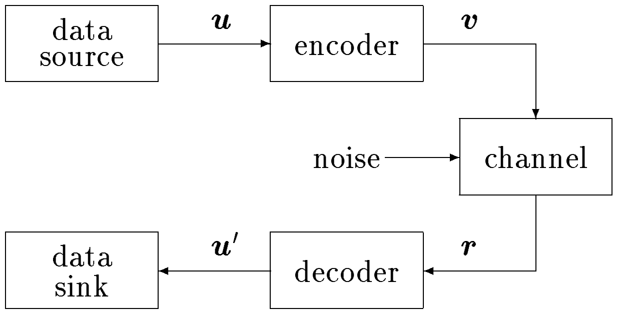


Figure 2.1. Simple transmission model.

the encoding rule.

The channels are usually disturbed from various kind of noise, but for simplicity we assume the *additive white Gaussian noise* (*AWGN*) model as the noise/channel model. Since the AWGN model is simple and easily analyzed in various contexts, and can provide an accurate model of what actually happens in some communication systems, the AWGN model is very popular and often used to discuss about the performance of the codes and decoding strategies.

The AWGN model assumes the chanel be *memoryless*, that means the noise process affecting a given symbol during its transmission is independent of that affecting preceding or succeedung symbols.

2.2 Binary Linear Block Code

We assume the output of the datasource is a sequence of binary digits $b \in GF(2)$, that is "0" or "1". This sequence is segmented into message block $\boldsymbol{u}$ consists of k information bits. The encoder transforms each message $\boldsymbol{u}$ into a codeword $\boldsymbol{v}$, a binary n-tuple, with certain encoding rules. Since there are 2^k distinct messages, there are 2^k codewords, and all of these codewords should be distinct for one-to-one correnspondences with messages. The set of codewords is the block code C.

It is favorable for a code to have the *linearlity*, since the linearity of the code reduces the complexity of encoding and decoding. In many cases, the codeword of length n is divided into two parts, the message part of k-length and the redundant checking part of $(n-k)$-length; this property is called the *systematic structure* and said favorable for a *linear* block code — but not necessary. The bit in message part is called an information bit, and the bit in redundant checking part is called a redundant (checking) bit or a parity (checking) bit. Intuitively, the linear code is a code such that the parity bits are derived by the linear function of k information bits. More formally, a code C of 2^k codewords (of n-length) is linear, if and only if the codewords form a k-dimensional subspace of the vector space of all the n-tuple over $GF(2)$.

Since an (n,k) linear block code C is a k-dimensional subspace of all the n-tuple over $GF(2)$, It is possible to find k linearly independent codewords $\boldsymbol{g}_1, \boldsymbol{g}_2, \ldots, \boldsymbol{g}_k$, which are the basises of C, such that every codewords $\boldsymbol{v}$ is a linear combination of these k codewords as follows.

$$\boldsymbol{v} = u_0\boldsymbol{g}_0 + u_1\boldsymbol{g}_1 + \cdots + u_{k-1}\boldsymbol{g}_{k-1}$$

where the message $\boldsymbol{u} = (u_1, u_2, \ldots, u_k)$ and $u_i = 0$ or 1 for $0 < i \leq k$. Then the following $n \times k$ matrix is called *generator matrix* of a code.

$$\mathbf{G} = \begin{bmatrix} \boldsymbol{g}_1 \\ \boldsymbol{g}_2 \\ \vdots \\ \boldsymbol{g}_k \end{bmatrix} = \begin{bmatrix} g_{1,1} & g_{1,2} & \cdots & g_{1,n} \\ g_{2,1} & g_{2,2} & \cdots & g_{2,n} \\ \vdots & \vdots & & \vdots \\ g_{k,1} & g_{k,2} & \cdots & g_{k,n}, \end{bmatrix}$$

where the message $\boldsymbol{g}_i = (g_{i,1}, g_{i,2}, \ldots, g_{i,k})$ for $0 < i \leq k$ and $g_{i,j} = 0$ or 1 for

$1 < j \leq n$. By using the generator matrix, every codeword $\boldsymbol{v}$ is derived easily from input message $\boldsymbol{u}$ as follows.

$$\boldsymbol{v} = \boldsymbol{u} \cdot \mathbf{G}$$

In the later chapters, some popular linear block codes will appear as to verify the performance of the proposed MAP decoding algorithms. So here those codes are briefly introduced just as examples of linear block codes. The *Reed-Muller* (*RM*) codes are popular linear block codes developed in 1950s. The structure of the RM codes is simple so that the RM codes are easy to construct and decode. The generator matrix of the first-order RM code of length 8, denoted as $\mathrm{RM}_{3,1}(8,4)$ or just RM(8,4), is very simple as follows.

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

The RM codes are known to be used in the *Mariner* spacecraft that explored Mars. In these days, the RM codes are frequently used in the scene that the high-speed decoding, which is easily realized, is needed. The binary *BCH* codes are generally considered as the most powerful block codes, and said to be often used in military and commercial satellite programs. The *extended BCH* (*eBCH*) code is the code extended by an additional redundant bit. Generally, an (n, k) block code becomes an $(n+1, k)$ extended code. Since the structure of original eBCH codes is regarded not very good to be used, so the eBCH codes are, usually, permuted for efficient usages. Permutation on the columns of generator matrices of the codes doesn't change the performance of codes, but only could make them easy to be decoded.

2.3 MLD and MAP Decoding

Every decoding strategy's aim is to transform a received sequence $\boldsymbol{r}$ into $\boldsymbol{u}$, that is $\boldsymbol{u}' = \boldsymbol{u}$. Several decoding methods are well-known as to achieve this aim in highly precise level; one most important is *maximum likelihood decoding* (*MLD*) and the another is *maximum a posteriori* (*MAP*) decoding.

The MLD is a decoding strategy to choose a codeword $\boldsymbol{v} \in C$ in order to maximize $\Pr(\boldsymbol{r}|\boldsymbol{v})$.

$$\Pr(\boldsymbol{r}|\boldsymbol{v}) = \frac{\Pr(\boldsymbol{r};\boldsymbol{v})}{\Pr(\boldsymbol{v})} \tag{2.1}$$

$$= \frac{\Pr(\boldsymbol{r}) \cdot \Pr(\boldsymbol{v}|\boldsymbol{r})}{\Pr(\boldsymbol{v})}. \tag{2.2}$$

MLD is optimum decoding strategy if the *a priori* probabilities $\Pr(\boldsymbol{v})$ are equally likely.

On the other hand, The MAP decoding has better decoding ability than the MLD if a priori probabilities are known in advance. Actually, the MAP decoding was originally defined as a decoding scheme to find a codeword $\boldsymbol{v} \in C$ which maximizes the *a posteriori probability* $\Pr(\boldsymbol{v}|\boldsymbol{r})$ for a received sequence $\boldsymbol{r}$. However, in the context of an iterative decoding, the sum of the a posteriori probabilities is more useful than the maximum a posteriori probability alone. Consequently, the MAP decoding is regarded as a decoding scheme to compute the following sum of probabilities for every bit position i with $1 \le i \le n$ and every symbol $b \in GF(2)$. (Forney proposes to call this decoding as an *APP* decoding to avoid confusion).

$$\sum_{\boldsymbol{v} \in C, \boldsymbol{v}[i]=b} \Pr(\boldsymbol{v}|\boldsymbol{r})$$

Since $\Pr(\boldsymbol{v}|\boldsymbol{r}) = \Pr(\boldsymbol{v};\boldsymbol{r})/\Pr(\boldsymbol{r})$ and $\Pr(\boldsymbol{r})$ can be regarded as a constant, the

Table 2.1. An example of a code.

codeword	$\Pr(\boldsymbol{v};\boldsymbol{r})$
(000)	0.01
(011)	0.05
(101)	0.03
(110)	0.02

essential part of the MAP decoding is the computation of

$$\sum_{\boldsymbol{v}\in C,\boldsymbol{v}[i]=b} \Pr(\boldsymbol{v};\boldsymbol{r}) = \sum_{\boldsymbol{v}\in C,\boldsymbol{v}[i]=b} \Pr(\boldsymbol{v})\cdot\Pr(\boldsymbol{r}|\boldsymbol{v}) \tag{2.3}$$

For example, let C be a (3,2) *single parity check code*[1] and the a priori probabilities $\Pr(\boldsymbol{v};\boldsymbol{r})$ for C are given as Table 2.1 then the MAP decoder will compute as follows.

$$\begin{aligned}
\sum_{\boldsymbol{v}\in C,\boldsymbol{v}[1]=0} \Pr(\boldsymbol{v};\boldsymbol{r}) &= \Pr((\mathbf{0}00);\boldsymbol{r}) + \Pr((\mathbf{0}11);\boldsymbol{r}) = 0.01+0.05 = 0.06,\\
\sum_{\boldsymbol{v}\in C,\boldsymbol{v}[1]=1} \Pr(\boldsymbol{v};\boldsymbol{r}) &= \Pr((\mathbf{1}01);\boldsymbol{r}) + \Pr((\mathbf{1}10);\boldsymbol{r}) = 0.03+0.02 = 0.05,\\
\sum_{\boldsymbol{v}\in C,\boldsymbol{v}[2]=0} \Pr(\boldsymbol{v};\boldsymbol{r}) &= \Pr((0\mathbf{0}0);\boldsymbol{r}) + \Pr((1\mathbf{0}1);\boldsymbol{r}) = 0.01+0.03 = 0.04,\\
\sum_{\boldsymbol{v}\in C,\boldsymbol{v}[2]=1} \Pr(\boldsymbol{v};\boldsymbol{r}) &= \Pr((0\mathbf{1}1);\boldsymbol{r}) + \Pr((1\mathbf{1}0);\boldsymbol{r}) = 0.05+0.02 = 0.07,\\
\sum_{\boldsymbol{v}\in C,\boldsymbol{v}[3]=0} \Pr(\boldsymbol{v};\boldsymbol{r}) &= \Pr((00\mathbf{0});\boldsymbol{r}) + \Pr((11\mathbf{0});\boldsymbol{r}) = 0.01+0.02 = 0.03,\\
\sum_{\boldsymbol{v}\in C,\boldsymbol{v}[3]=1} \Pr(\boldsymbol{v};\boldsymbol{r}) &= \Pr((01\mathbf{1});\boldsymbol{r}) + \Pr((10\mathbf{1});\boldsymbol{r}) = 0.05+0.03 = 0.08.
\end{aligned}$$

And output them as a form of a table (called *MAP table*) as Table 2.2.

[1]The $(k+1,k)$ codes with the codewords $\boldsymbol{v}$ consist of information bits $v_i = u_i$ for $0 < i \le k$ and a parity bit v_{k+1} such that v_{k+1} is the modulo-2 summation of $v_0,\ldots,v_{k-1}$.

Table 2.2. A MAP table.

	1	2	3
0	0.06	0.04	0.03
1	0.05	0.07	0.08

2.4 Iterative Decoding

Recently, the iterative decoding is investigated by many researchers, since it shows very good decoding performance. The decoding strategy of the iterative decoding is decoding the received sequence $\boldsymbol{r}$ once and then re-decoding the (intuitively) same $\boldsymbol{r}$ some times using the result of last decoding.

Turbo code is the code with its exclusive iterative decoding method. Figure 2.2 is the model of turbo encoder. The encoder computes 2 different sets of parity bits A and B from the information part $\boldsymbol{u}$ and sends the concatenation $M \circ A \circ B$ as the codeword. Interleaver permutes the sequence of $\boldsymbol{u}$. Note that $A = E(\boldsymbol{u}), B = E(I(\boldsymbol{u}))$ and $I^{-1}(I(\boldsymbol{u})) = \boldsymbol{u}$, where we denote encoding as $E(\cdot)$ and interleaving as $I(\cdot)$. Figure 2.3 is the model of turbo decoding. The decoder receives $\boldsymbol{r} = \boldsymbol{u}' \circ A' \circ B'$ and devides it into $\boldsymbol{r}_1 = \boldsymbol{u}' \circ A'$ and $\boldsymbol{r}_2 = \boldsymbol{u}' \circ B'$. As the initial step, the MAP decoding is executed for $\boldsymbol{r}_1$ with no a priori probabilities, i.e. equal probabilities, and MAP table for the $\boldsymbol{r}_1$ is computed. Then the decoding results, the MAP table, of $\boldsymbol{r}_1$ is used as the table of a priori probabilities for MAP decoding $\boldsymbol{r}_2$. Similary the decoding results, the MAP table, of $\boldsymbol{r}_2$ is used for MAP decoding $\boldsymbol{r}_1$ again. The MAP decoding (and interleaving) is repeated several times for a received sequence $\boldsymbol{r}$.

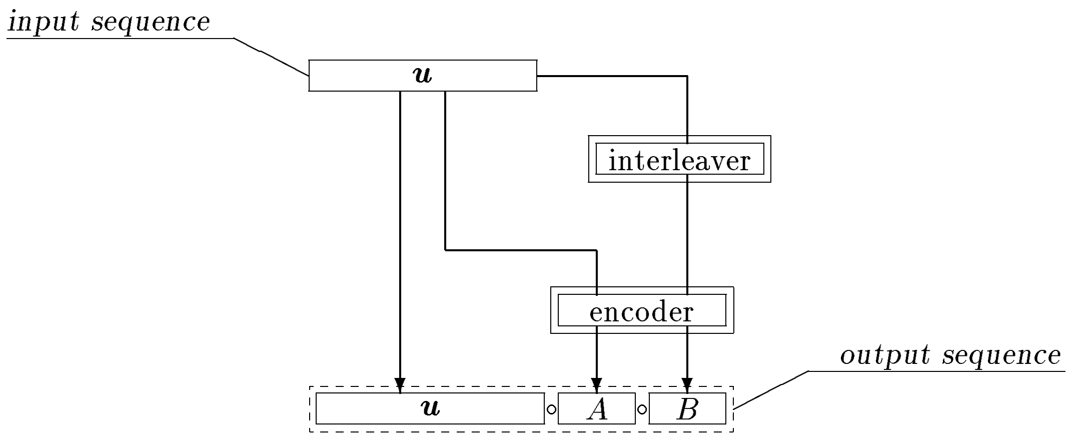


Figure 2.2. Turbo encoder.

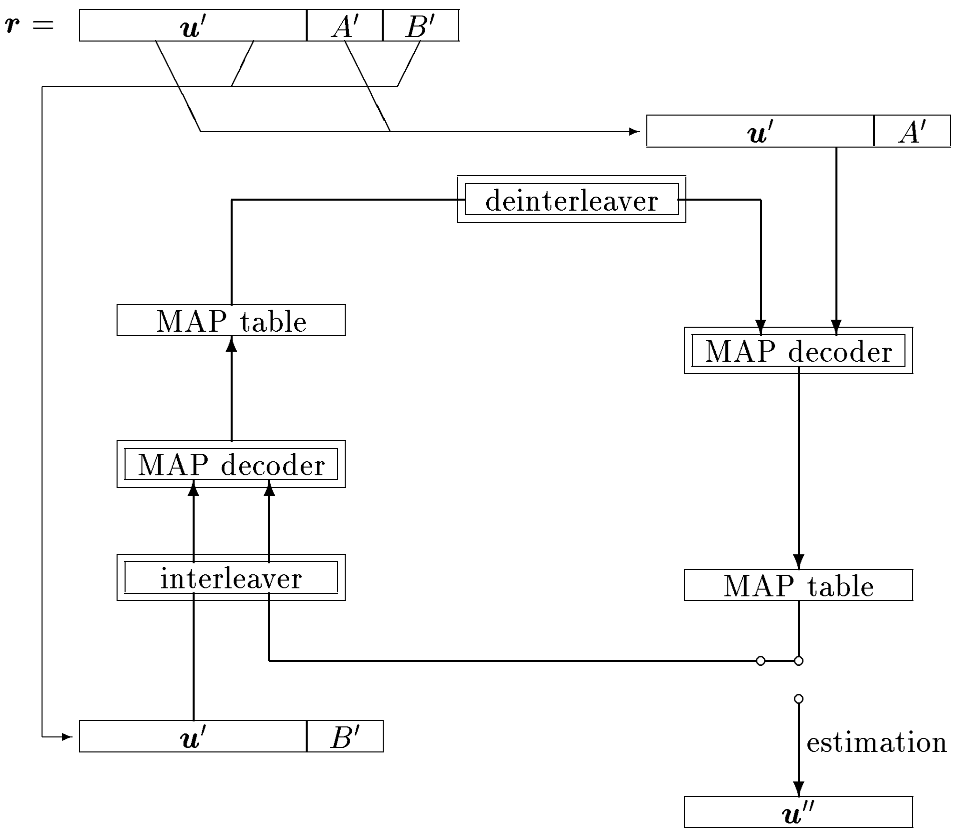


Figure 2.3. Turbo decoder.

Chapter 3

Trellis Diagram

The concept of trellis diagrams are useful to discuss about the structure of code. For that reason, the trellis diagram and some of its properties are introduced in this chapter.

3.1 Trellis Diagram

Let C be an (n,k) linear block code. A *trellis diagram* T of a code C is a transition diagram of the finite state machine with an initial state and a final state, and the branches are labeled by symbol 0 or 1 such that the acceptable sequences are corresponding to the codewords of C by one-to-one. Namely, a trellis diagram represents a code C, therefore, the structure of a trellis diagram is often used for analysis of a code, decoding of a code, and so on.

Figure 3.1 is a trellis diagram T corresponding to the codewords of $\mathrm{RM}_{3,1}(8,4)$

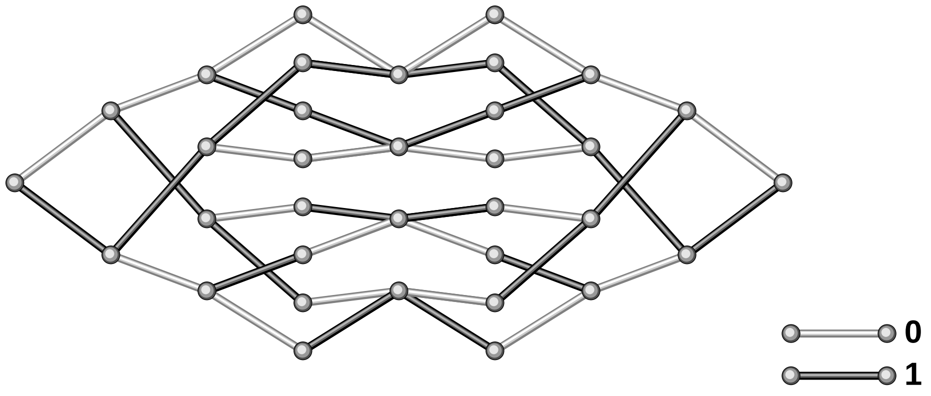


Figure 3.1. The trellis diagram of $\mathrm{RM}_{3,1}(8,4)$.

code such as follows.

$$\left\{\begin{array}{llll}(00000000), & (00001111), & (00110011), & (00111100),\\ (01010101), & (01011010), & (01100110), & (01101001),\\ (10010110), & (10011001), & (10100101), & (10101010),\\ (11000011), & (11001100), & (11110000), & (11111111)\end{array}\right\}.$$

For example, a path which appears with connected bold lines in Figure 3.2 represents a codeword $\boldsymbol{v} = (01010101)$. Obviously, every path in T corresponds to a certain codeword, and there is no path which doesn't correspond to any codeword.

3.2 Section Trellis Diagram

In this section, a *section trellis diagram* is introduced and its properties are discussed briefly. Define a (n,k) linear block code C. A trellis diagram T of the code C is said to be *minimum* if the number of states of T is minimum among all trellis diagrams of C. Figure 3.3 shows the minimization of the trellis for a

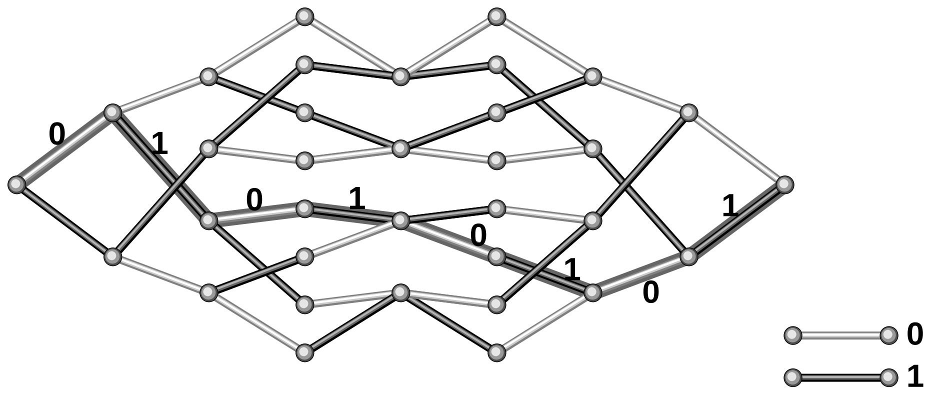


Figure 3.2. A codeword and a path in a trellis diagram.

simple (3,2) parity check code, and the most lower one in the figure is the minimum trellis. Figure 3.1 is also minimum. For an integer t with $0 \leq t \leq n$, let Σ_t be the state space of T at time-t. For two states σ and σ' of a trellis diagram, let $L(\sigma, \sigma')$ be a set of vectors which are associated with the paths from the state σ to σ'. Thus, if σ is the initial state and σ' is the final state of the trellis diagram, then $L(\sigma, \sigma') = C$. Let $B = \{b_0, b_1, \ldots, b_m\}$ be a set of integers satisfying $0 = b_0 < b_1 < \cdots < b_m = n$. A *section trellis diagram of C with boundaries at B* is a state diagram obtained from T by removing all states at time-t with $t \notin B$, and by connecting every state pair $\sigma \in \Sigma_{b_j}$ and $\sigma' \in \Sigma_{b_{j+1}}$ with $L(\sigma, \sigma') \neq \emptyset$ for $0 \leq j \leq m-1$. Furthermore, the path from σ to σ' is associated with the set of vectors $L(\sigma, \sigma')$. Figure 3.1 shows that the minimum trellis diagram of the $\mathrm{RM}_{3,1}(8, 4)$ code, and Figure 3.5 shows the section trellis diagram with boundaries at $\{0, 2, 4, 6, 8\}$.

The minimum trellis diagram and section trellis diagrams have strong relation to a coset structure of a code. Let $x = b_j$ and $y = b_{j+1}$ for $0 \leq j \leq m-1$, and let C_{xy} be a set of codewords of C such that the first x and the last $n - y$ symbols

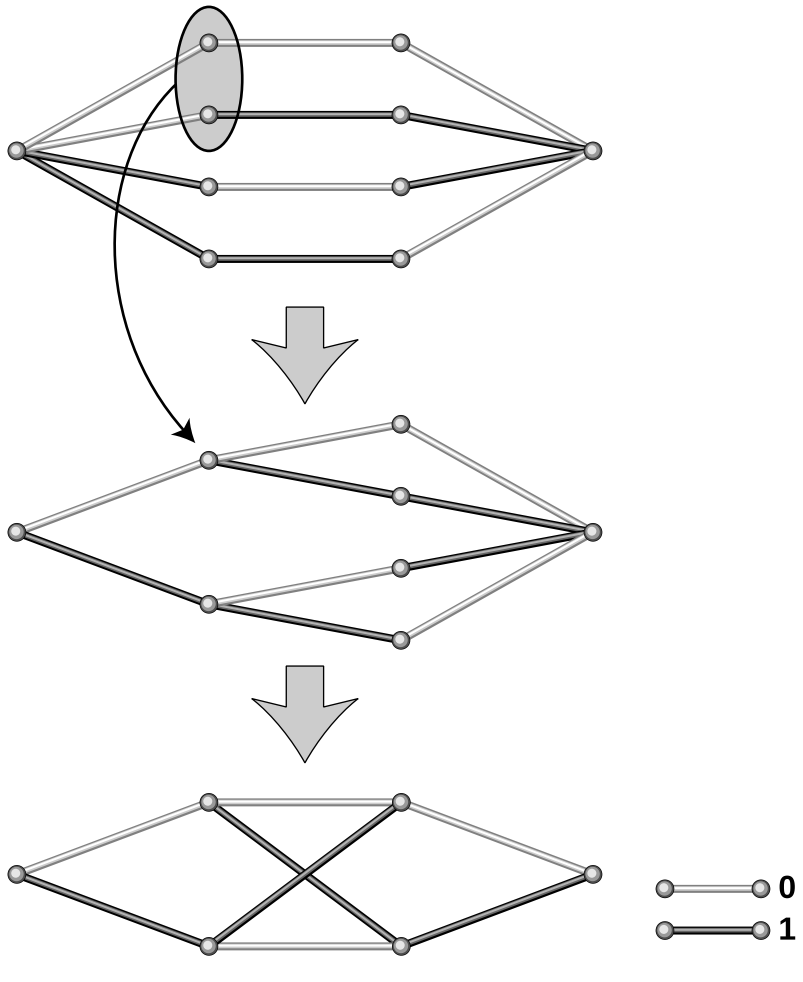


Figure 3.3. Minimization of the trellis for a simple (3,2) parity check code.

are all zero. Also let $p_{xy}(C)$ be a set of vectors which are obtained by puncturing the first x and the last $n-y$ symbols of each codeword of C. $p_{xy}(C)$ is called a *punctured code* of C. Obviously $p_{xy}(C_{xy})$ is a linear subcode of $p_{xy}(C)$ and we can partition the set $p_{xy}(C)$ into cosets of $p_{xy}(C_{xy})$. Define L_{xy} be the set of cosets of $p_{xy}(C_{xy})$ in $p_{xy}(C)$. Figure 3.4 shows the relation of $p_{xy}(C_{xy})$, $p_{xy}(C)$ and the trellis diagram.

It is known that, if there is a branch from a state $\sigma_x \in \Sigma_x$ to $\sigma_y \in \Sigma_y$ in a trellis diagram, then $L(\sigma_x, \sigma_y) \in L_{xy}$. That is, each branch from a state in Σ_x to a state in Σ_y is associated with a coset of C_{xy}. This implies that $|L_{xy}|$ different sets are associated with the branches from states in time-x to states in time-y. If we denote the dimension of a set A as $k(A)$, then some important parameters of a trellis diagram is expressed in a simple way[9]. For example, $|L_{xy}| = 2^{k(p_{xy}(C))-k(C_{xy})}$, $|\Sigma_t| = 2^{k(C)-k(C_{0t})-k(C_{tn})}$ for $t \in B$, and the total number of branches between time-x and time-y is $2^{k(C)-k(C_{0x})-k(C_{xy})-k(C_{yn})}$. Consequently, a coset is associated to $2^{k(C)-k(C_{0x})-k(C_{xy})-k(C_{yn})}/|L_{xy}| = 2^{k(C)-k(p_{xy}(C))-k(C_{0x})-k(C_{yn})}$ different branches.

Figure 3.6 shows that, for any integer z with $x < z < y$, there is a unique integer $m_z(x,y)$, and any coset in L_{xy} can be represented as a union of $m_z(x,y)$ concatenations of cosets in L_{xz} and L_{zy}[9]. That is, for any $D_{xy} \in L_{xy}$, there are $2m_z(x,y)$ cosets $D_{xz}^i \in L_{xz}$ and $D_{zy}^i \in L_{zy}$ with $1 \le i \le m_z(x,y)$ such that

$$D_{xy} = D_{xz}^1 \circ D_{zy}^1 \cup \cdots \cup D_{xz}^{m_z(x,y)} \circ D_{zy}^{m_z(x,y)}.$$

It is also known that $D_{xz}^i \circ D_{zy}^i \cap D_{xz}^j \circ D_{zy}^j = \emptyset$ for different i and j.

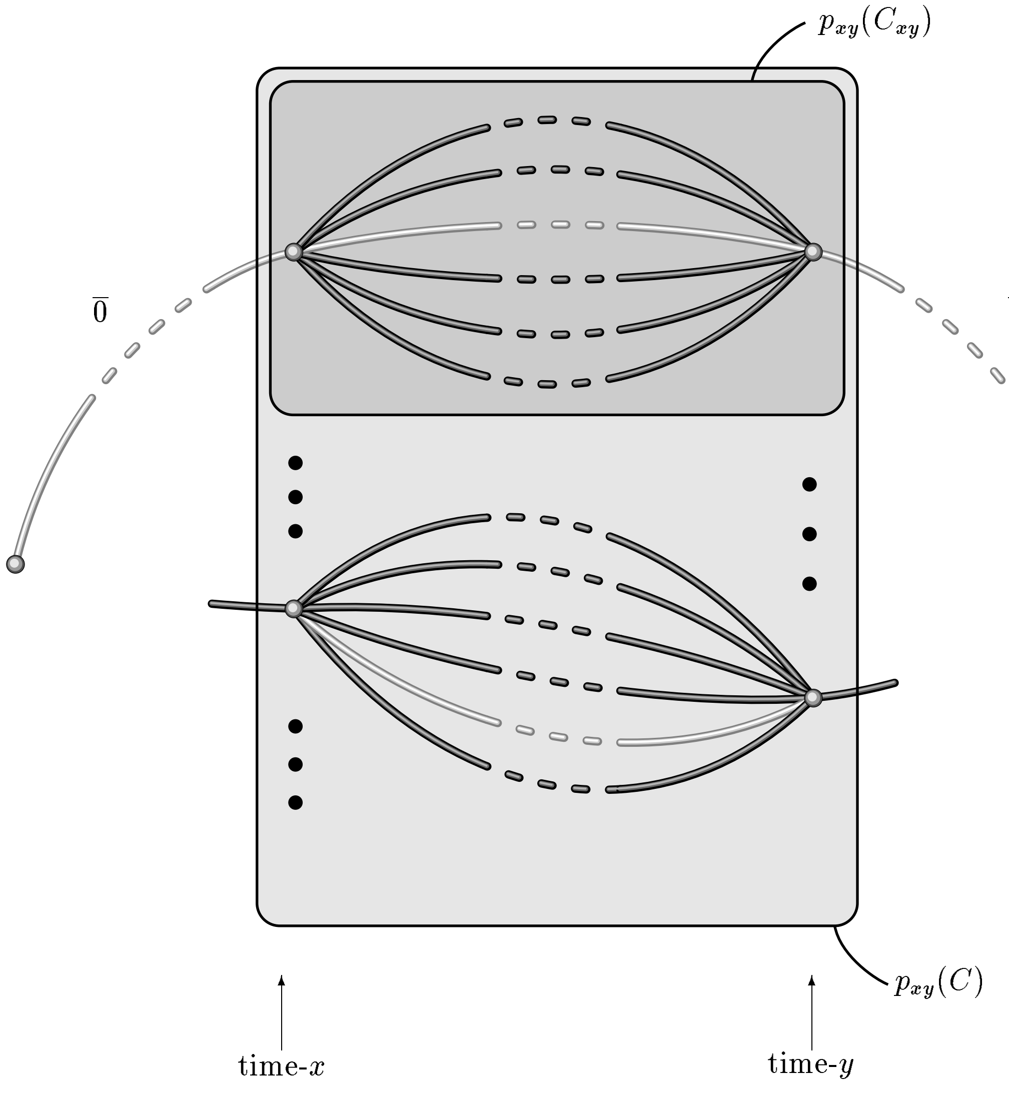


Figure 3.4. $p_{xy}(C_{xy}), p_{xy}(C)$ and the trellis diagram.

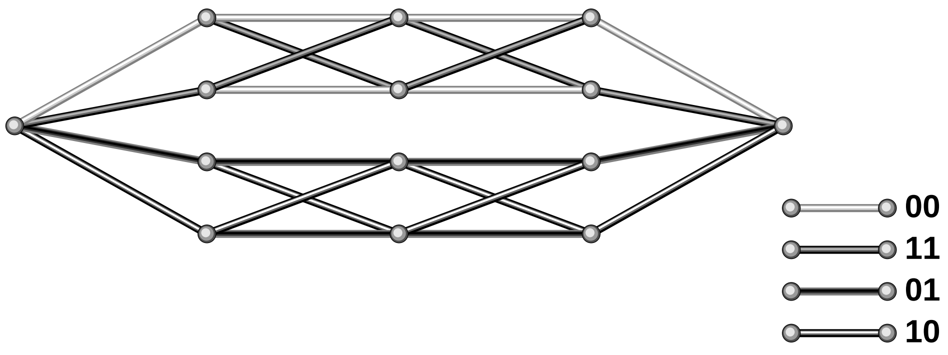


Figure 3.5. The section trellis diagram of $\mathrm{RM}_{3,1}(8,4)$ with $B = \{0, 2, 4, 6, 8\}$.

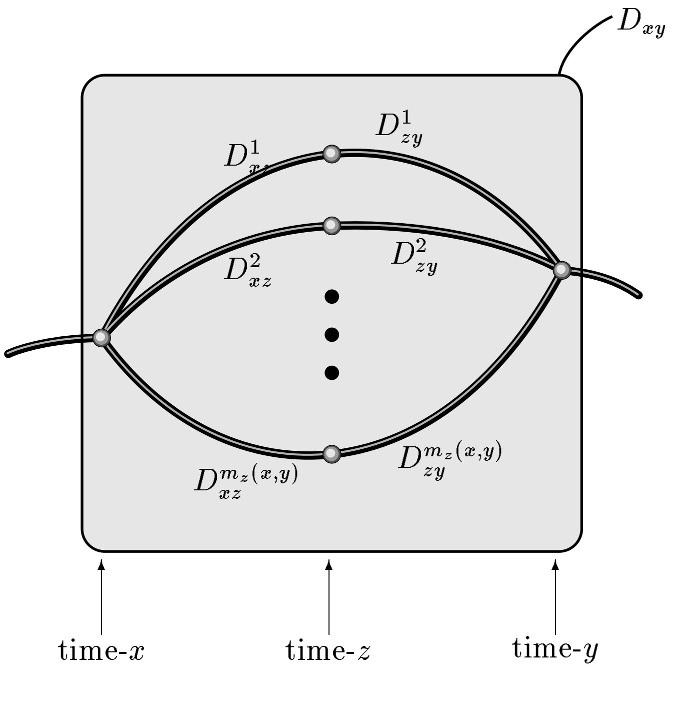


Figure 3.6. Decomposition of a coset D_{xy}.

Chapter 4

MAP Decoding

With the change of the usage of the MAP decoding, the scheme of the MAP decoding itself has changed. We treat only the MAP decoding of modern style, and its essence is introduced in this chapter.

4.1 MAP Decoding

Assume that an (n, k) *binary linear block code* C is used for error control over *AWGN channel*. The set of k *information bit positions* of C is denoted by $\mathcal{I}(C) \subseteq \{1, \ldots, n\}$. It is assumed that the symbols at information bit positions are chosen independently, and it is also assumed that, for any $i \in \mathcal{I}(C)$ and $b \in GF(2)$, we know in advance the *a priori probability* $\Pr_i(b)$ that the symbol b is chosen as the symbol at the i-th bit position. For a vector $\boldsymbol{v} = (v_1, \ldots, v_m)$ of length m, let $\boldsymbol{v}[i]$ with $1 \leq i \leq m$ be the i-th symbol v_i of $\boldsymbol{v}$. For two vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$, the vector obtained by concatenating $\boldsymbol{v}_2$ to $\boldsymbol{v}_1$ is denoted $\boldsymbol{v}_1 \circ \boldsymbol{v}_2$. For two sets of vectors A and B, define $A \circ B \overset{\triangle}{=} \{\boldsymbol{v}_1 \circ \boldsymbol{v}_2 : \boldsymbol{v}_1 \in A, \boldsymbol{v}_2 \in B\}$.

For a received vector $\boldsymbol{r}$, the most essential part of the MAP decoding is the computation of the following sum of probabilities

$$\sum_{\boldsymbol{v}\in C,\boldsymbol{v}[i]=b} \Pr(\boldsymbol{v};\boldsymbol{r}) = \sum_{\boldsymbol{v}\in C,\boldsymbol{v}[i]=b} \Pr(\boldsymbol{v})\cdot\Pr(\boldsymbol{r}|\boldsymbol{v}) \tag{4.1}$$

for every bit position i with $1 \le i \le n$ and every symbol $b \in GF(2)$. Since each symbol at the information bit position is chosen independently,

$$\Pr(\boldsymbol{v}) = \prod_{j\in\mathcal{I}(C)} \Pr_j(\boldsymbol{v}[j]).$$

Since the a priori probabilities for redundant bits have no meaning for MAP decoding, define $\Pr_j(\boldsymbol{v}[j]) = 1$ for $j \notin \mathcal{I}(C)$, then

$$\Pr(\boldsymbol{v}) = \prod_{1\le j\le n} \Pr_j(\boldsymbol{v}[j]). \tag{4.2}$$

On the other hand, since the channel is assumed to be memoryless,

$$\Pr(\boldsymbol{r}|\boldsymbol{v}) = \prod_{1\le j\le n} \Pr(\boldsymbol{r}[j]|\boldsymbol{v}[j]). \tag{4.3}$$

Summarizing (4.2) and (4.3), we can write (4.1) as

$$\sum_{\boldsymbol{v}\in C,\boldsymbol{v}[i]=b} \Pr(\boldsymbol{v};\boldsymbol{r}) = \sum_{\boldsymbol{v}\in C,\boldsymbol{v}[i]=b}\ \prod_{1\le j\le n} \Pr_j(\boldsymbol{v}[j])\cdot\Pr(\boldsymbol{r}[j]|\boldsymbol{v}[j]). \tag{4.4}$$

4.2 BCJR Algorithm

The first MAP decoding algorithm was independently proposed by Bahl et al.[1] and McAdam et al.[17], which is known as the *BCJR algorithm* nowadays. The BCJR algorithm is introduced briefly in this section. Though some expressions in the original papers are conformed in this thesis, to fit the arguments in the later chapters, the general ideas are sufficiently kept.

The BCJR algorithm calculates (4.4) by using the trellis diagram T of the code. First, define and calculate

$$q_j[0] = \begin{cases} \Pr_j(0)\cdot\Pr(\boldsymbol{r}[j]|0) & \text{if } j \in \mathcal{I}(C) \\ \Pr(\boldsymbol{r}[j]|0) & \text{otherwise,} \end{cases}$$

$$q_j[1] \;=\; \begin{cases} \Pr_j(1)\cdot\Pr(\boldsymbol{r}[j]|1) & \text{if } j \in \mathcal{I}(C) \\ \Pr(\boldsymbol{r}[j]|1) & \text{otherwise} \end{cases}$$

for simplicity, and associate those values with T. Then

$$\Pr_j(\boldsymbol{v}[j])\cdot\Pr(\boldsymbol{r}[j]|\boldsymbol{v}[j]) = q_j(\boldsymbol{v}[j]).$$

For an integer t with $0 \le t \le n$, let Σ_t be the state space of T at time-t. Then $\alpha(\sigma)$, called the α-value at state σ, where $\sigma \in \Sigma_t$ is defined as follows.

$$\alpha(\sigma) = \sum_{\boldsymbol{v}=(v_1,v_2,\ldots,v_t)\in L(\sigma_0,\sigma)} \prod_{0<j\le t} q_j(\boldsymbol{v}[j])$$

where σ_0 is the initial state.

Let $\rho_1, \rho_2, \ldots, \rho_m \in \Sigma_{t-1}$ be states which are connected to $\sigma \in \Sigma_t$ in T. Then we can write

$$L(\sigma_0,\sigma) = \bigcup_{s=1}^{m} L(\sigma_0,\rho_s)\circ L(\rho_s,\sigma)$$

and $\alpha(\sigma)$ is expressed as

$$\begin{aligned} \alpha(\sigma) &= \sum_{\boldsymbol{v}=(v_1,v_2,\ldots,v_{t-1})\in L(\sigma_0,\rho_s)} \prod_{0<j\le t-1} q_j(\boldsymbol{v}[j]) \cdot \sum_{\boldsymbol{v}=(v_t)\in L(\rho_s,\sigma)} q_t(\boldsymbol{v}[t]) \\ &= \sum_{s=1}^{m} \alpha(\rho_s)\cdot \sum_{\boldsymbol{v}=(v_t)\in L(\rho_s,\sigma)} q_t(\boldsymbol{v}[t]). \end{aligned}$$

Thus the computation of α-values is done recursively using the trellis diagram T and branch probabilities $q_j(\boldsymbol{v}[j])$, as Figure 4.1 shows.

Similarly, $\beta(\sigma)$, called the β-value at state σ, where $\sigma \in \Sigma_t$ is defined as follows.

$$\beta(\sigma) = \sum_{\boldsymbol{v}=(v_{t+1},\ldots,v_{n-1},v_n)\in L(\sigma,\sigma_n)} \prod_{t<j\le n} q_j(\boldsymbol{v}[j])$$

where σ_n is the final state. Also β-values are computed recursively.

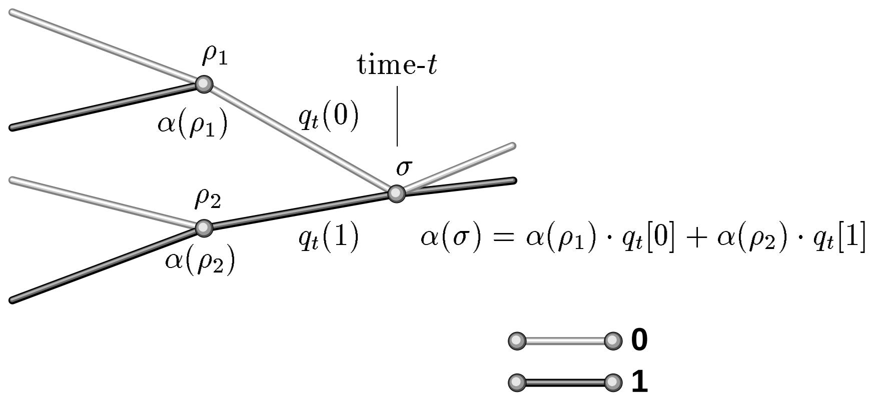


Figure 4.1. The computation of $\alpha(\sigma)$

Finally, γ-value is computed for every branch of T (Figure 4.2), $L(\sigma_{t-1}, \sigma_t) \in L_{t-1,t}$ where $\sigma_{t-1} \in \Sigma_{t-1}$ and $\sigma_t \in \Sigma_t$ for $0 < t \leq n$, as follows.

$$\gamma(L(\sigma_{t-1}, \sigma_t)) = \alpha(\sigma_{t-1}) \cdot q_j[L(\sigma_{t-1}, \sigma_t)] \cdot \beta(\sigma_t).$$

Then

$$\sum_{\boldsymbol{v}[j]=b} \prod_{1 \leq j \leq n} \Pr_j(\boldsymbol{v}[j]) \cdot \Pr(\boldsymbol{r}[j] | \boldsymbol{v}[j]) = \sum_{\{b\} \in L(\sigma_{i-1}, \sigma_i)} \gamma(L(\sigma_{i-1}, \sigma_i))$$

and, thus, the MAP decoding is done.

4.3 Improvement on MAP Decoding

The BCJR algorithm needs to construct the entire trellis diagram of the code C first, and then the computation for α-values, β-values and γ-values is done next. The computational complexity is approximately linear to the number of the branches in the minimum trellis of C. The number of the branches becomes very large in the trellis diagram, even if it is minimum, when the code is large.

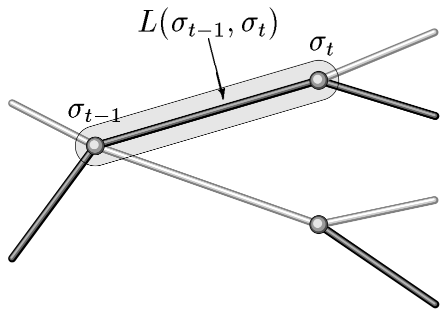


Figure 4.2. The branch $L(\sigma_{t-1}, \sigma_t)$

Therefore often it is almost impossible to construct the entire trellis diagram for a long practical block codes.

Efforts have been made to reduce the decoding complexity of the BCJR algorithm ever. One of such attempts includes a suboptimum realization of the BCJR algorithm. For example, the Log-MAP (MAX-Log-MAP) algorithm and the SOVA (Soft-Output Viterbi Algorithm)[11] use log-likelihood ratios and some approximations to avoid calculating the actual probabilities, and simplify some computations. Since those approaches adopt approximations, the decoding performance have to be worse than the original (optimum) MAP decoding.

Another approach for reducing the complexity is to investigate other algorithms for the MAP decoding. In Chapter5 and 6, the efficient MAP decoding algorithms are presented. The proposed algorithms have the identical decoding performance with the original MAP decoding, that is the BCJR algorithm introduced in this chapter.

Chapter 5

rMAP Algorithm

5.1 Preliminaries

In this section, an important part of the last chapter is reviewed, just for the confirmation.

It is assumed that an (n, k) *binary linear block code* C is used for error control over *AWGN channel.* The set of k *information bit positions* of C is denoted by $\mathcal{I}(C) \subseteq \{1, \ldots, n\}$. It is assumed that the symbols at information bit positions are chosen independently, and it is also assumed that, for any $i \in \mathcal{I}(C)$ and $b \in GF(2)$, we know in advance the *a priori probability* $\Pr_i(b)$ that the symbol b is chosen as the symbol at the i-th bit position. For a vector $\boldsymbol{v} = (v_1, \ldots, v_m)$ of length m, let $\boldsymbol{v}[i]$ with $1 \leq i \leq m$ be the i-th symbol v_i of $\boldsymbol{v}$. For two vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$, the vector obtained by concatenating $\boldsymbol{v}_2$ to $\boldsymbol{v}_1$ is denoted $\boldsymbol{v}_1 \circ \boldsymbol{v}_2$. For two sets of vectors A and B, define $A \circ B \stackrel{\triangle}{=} \{\boldsymbol{v}_1 \circ \boldsymbol{v}_2 : \boldsymbol{v}_1 \in A, \boldsymbol{v}_2 \in B\}$.

For a received vector $\boldsymbol{r}$, the most essential part of the MAP decoding is the computation of the following sum of probabilities

$$\sum_{\boldsymbol{v} \in C, \boldsymbol{v}[i]=b} \Pr(\boldsymbol{v}; \boldsymbol{r}) = \sum_{\boldsymbol{v} \in C, \boldsymbol{v}[i]=b} \Pr(\boldsymbol{v}) \cdot \Pr(\boldsymbol{r}|\boldsymbol{v}) \tag{5.1}$$

for every bit position i with $1 \leq i \leq n$ and every symbol $b \in GF(2)$. Since each symbol at the information bit position is chosen independently,

$$\Pr(\boldsymbol{v}) = \prod_{j \in \mathcal{I}(C)} \Pr{}_j(\boldsymbol{v}[j]).$$

Define $\Pr_j(\boldsymbol{v}[j]) = 1$ for $j \notin \mathcal{I}(C)$, then

$$\Pr(\boldsymbol{v}) = \prod_{1 \leq j \leq n} \Pr{}_j(\boldsymbol{v}[j]). \tag{5.2}$$

On the other hand, since the channel is assumed to be memoryless,

$$\Pr(\boldsymbol{r}|\boldsymbol{v}) = \prod_{1 \leq j \leq n} \Pr(\boldsymbol{r}[j]|\boldsymbol{v}[j]). \tag{5.3}$$

Summarizing (5.2) and (5.3), we can write (5.1) as

$$\sum_{\boldsymbol{v} \in C, \boldsymbol{v}[i]=b} \Pr(\boldsymbol{v};\boldsymbol{r}) = \sum_{\boldsymbol{v} \in C, \boldsymbol{v}[i]=b} \prod_{1 \leq j \leq n} \Pr{}_j(\boldsymbol{v}[j]) \cdot \Pr(\boldsymbol{r}[j]|\boldsymbol{v}[j]).$$

5.2 Recursive MAP Decoding Algorithm "rMAP"

In this chapter, a recursive algorithm for the MAP decoding is proposed, that is called "rMAP" algorithm in subsequent chapters.

5.2.1 MAP decoding as table construction problem

As same as defined in Chapter refchap:trellis, again let C_{xy} be the set of codewords of C such that the first x and the last $n-y$ symbols are all zero, for integers x and y such that $1 \leq x < y \leq n$. Also let $p_{xy}(C)$ be the punctured code of C obtained by removing the first x and the last $n-y$ symbols of each codeword in C. Obviously $p_{xy}(C_{xy})$ is a linear subcode of $p_{xy}(C)$. Define L_{xy} be the set of cosets of $p_{xy}(C_{xy})$ in $p_{xy}(C)$. It is known that, for any integer z with $x < z < y$, there is a unique integer $m_z(x,y)$, and any coset in L_{xy} can be represented as a union of

Table 5.1. A MAP table and $\mathrm{MAP}(D_{xy}, i, b)$.

bit position	$x+1$	$\cdots$	i	$\cdots$	y
0		$\cdots$		$\cdots$	
1		$\cdots$		$\cdots$	

$\mathrm{MAP}(D_{xy}, i, 0)$

$\mathrm{MAP}(D_{xy}, i, 1)$

$m_z(x,y)$ concatenations of cosets in L_{xz} and L_{zy}[9]. That is, for any $D_{xy} \in L_{xy}$, there are $2m_z(x,y)$ cosets $D^i_{xz} \in L_{xz}$ and $D^i_{zy} \in L_{zy}$ with $1 \le i \le m_z(x,y)$ such that

$$D_{xy} = D^1_{xz} \circ D^1_{zy} \cup \cdots \cup D^{m_z(x,y)}_{xz} \circ D^{m_z(x,y)}_{zy}. \tag{5.4}$$

It is also known that $D^i_{xz} \circ D^i_{zy} \cap D^j_{xz} \circ D^j_{zy} = \emptyset$ for different i and j.

For a coset $D_{xy} \in L_{xy}$, an integer i with $x < i \le y$ and a symbol $b \in GF(2)$, define

$$D^{i/b}_{xy} \overset{\triangle}{=} \{(v_{x+1}, \ldots, v_{i-1}, b, v_{i+1}, \ldots, v_y) \in D_{xy}\}$$

and

$$\mathrm{MAP}(D_{xy}, i, b) \overset{\triangle}{=} \sum_{\boldsymbol{v}=(v_{x+1},\ldots,v_y) \in D^{i/b}_{xy}} \prod_{x<j\le y} \mathrm{Pr}_j(\boldsymbol{v}[j]) \cdot \mathrm{Pr}(\boldsymbol{r}[j] | \boldsymbol{v}[j]).$$

A *MAP table for the coset* $D_{xy} \in L_{xy}$ (Table 5.1) is a table which has $y - x$ columns and two rows. The columns are indexed by numbers from $x+1$ to y, and the rows are indexed by 0 and 1. The cell of a table whose column index is i and row index is b is defined to be $\mathrm{MAP}(D_{xy}, i, b)$. Obviously, computing the MAP table for $C \in L_{0,n}$ yields the MAP decoding. Also define the *total* *probability of D_{xy}* as

$$\mathrm{MAP}(D_{xy}) \overset{\triangle}{=} \sum_{\boldsymbol{v} \in D_{xy}} \prod_{x<j\leq y} \mathrm{Pr}_j(\boldsymbol{v}[j]) \cdot \mathrm{Pr}(\boldsymbol{r}[j]|\boldsymbol{v}[j]).$$

It is easily understood that

$$\mathrm{MAP}(D_{xy}) = \mathrm{MAP}(D_{xy}, i, 0) + \mathrm{MAP}(D_{xy}, i, 1),$$

and

$$\mathrm{MAP}(D_{xy}, i, b) = \mathrm{MAP}(D_{xy}^{i/b}). \tag{5.5}$$

for an arbitrary $x < i \leq y$. Furthermore, the following lemmas and corollary can be shown easily.

Lemma 5.2.1: If a set D_{xy} is written $D_{xy} = D_1 \cup D_2$ such that $D_1 \cap D_2 = \emptyset$ with $D_1, D_2 \in L_{xy}$, then

$$\mathrm{MAP}(D_{xy}, i, b) = \mathrm{MAP}(D_1, i, b) + \mathrm{MAP}(D_2, i, b),$$

and

$$\mathrm{MAP}(D_{xy}) = \mathrm{MAP}(D_1) + \mathrm{MAP}(D_2).$$

Proof. Obvious from the definition of $\mathrm{MAP}(D_{xy}, i, b)$. □

Lemma 5.2.2: If a set D_{xy} is written $D_{xy} = D_1 \circ D_2$ with $D_1 \in L_{xz}$ and $D_2 \in L_{zy}$, then

$$\mathrm{MAP}(D_{xy}, i, b) = \begin{cases} \mathrm{MAP}(D_1, i, b) \cdot \mathrm{MAP}(D_2) & (x \leq i < z) \\ \mathrm{MAP}(D_1) \cdot \mathrm{MAP}(D_2, i, b) & (z \leq i \leq y), \end{cases}$$

and

$$\mathrm{MAP}(D) = \mathrm{MAP}(D_1) \cdot \mathrm{MAP}(D_2).$$

Proof. If $x < i \leq z$ then $D_{xy}^{i/b}$ is written as $D_{xy}^{i/b} = D_1^{i/b} \circ D_2$. Consequently the following transformations are possible.

$$\begin{aligned}
\mathrm{MAP}(D_{xy}, i, b) &= \sum_{\boldsymbol{v} \in D_{xy}^{i/b}} \prod_{x<j\leq y} \mathrm{Pr}_j(\boldsymbol{v}[j]) \cdot \mathrm{Pr}(\boldsymbol{r}[j]|\boldsymbol{v}[j]) \\
&= \sum_{\boldsymbol{v}_1 \in D_1^{i/b}} \sum_{\boldsymbol{v}_2 \in D_2} \prod_{x<j\leq z} \mathrm{Pr}_j(\boldsymbol{v}_1[j]) \cdot \mathrm{Pr}(\boldsymbol{r}[j]|\boldsymbol{v}_1[j]) \\
&\qquad \cdot \prod_{z<j\leq y} \mathrm{Pr}_j(\boldsymbol{v}_2[j]) \cdot \mathrm{Pr}(\boldsymbol{r}[j]|\boldsymbol{v}_2[j]) \\
&= \sum_{\boldsymbol{v}_1 \in D_1^{i/b}} \prod_{x<j\leq z} \mathrm{Pr}_j(\boldsymbol{v}_1[j]) \cdot \mathrm{Pr}(\boldsymbol{r}[j]|\boldsymbol{v}_1[j]) \\
&\qquad \cdot \sum_{\boldsymbol{v}_2 \in D_2} \prod_{z<j\leq y} \mathrm{Pr}_j(\boldsymbol{v}_2[j]) \cdot \mathrm{Pr}(\boldsymbol{r}[j]|\boldsymbol{v}_2[j]) \\
&= \mathrm{MAP}(D_1, i, b) \cdot \mathrm{MAP}(D_2).
\end{aligned}$$

Therefore the lemma holds if $x \leq i < z$. The lemma can be shown similarly for the case $z \leq i < y$.

□

Corollary 5.2.3: If D_{xy} is written as (5.4) then

$$\mathrm{MAP}(D_{xy}, i, b) = \sum_{j=1}^{m_z} \mathrm{MAP}(D_{xz}^j, i, b) \cdot \mathrm{MAP}(D_{zy}^j)$$

for $x < i \leq z$, and

$$\mathrm{MAP}(D_{xy}, i, b) = \sum_{j=1}^{m_z} \mathrm{MAP}(D_{xz}^j) \cdot \mathrm{MAP}(D_{zy}^j, i, b)$$

for $z < i \leq y$. □

5.2.2 Recursive algorithm for MAP decoding

Define $\mathcal{M}(x, y) \triangleq \{\mathrm{MAP}(D_{xy}) : D_{xy} \in L_{xy}\}$. The following recursive algorithm rMAP(x, y) computes $\mathcal{M}(x, y)$ for given x and y. In the proposed algorithm rMAP, the computation is carried out in a divide-and-conquer manner. To compute the MAP tables for a long section, say from time-x to time-y, the section is divided into two shorter subsections, say time-x to time-z and time-z to time-y where $x < z < y$. The sets of MAP tables $\mathcal{M}(x,z)$ and $\mathcal{M}(z,y)$ for these subsections are computed first, and then combined to obtain the final MAP tables $\mathcal{M}(x,y)$. In this way, a section is divided until direct computation of the MAP tables is computationally acceptable.

Algorithm 5.2.1: rMAP(x,y)

  input: integers x and y with $0 \le x < y \le n$

 output: $\mathcal{M}(x,y)$

If $y - x = 1$, then execute the following *non-recursive step.* If $y - x > 1$, then execute either the non-recursive step or the *recursive step.*

non-recursive step: compute $\mathcal{M}(x,y)$ directly by the following procedure dMAP(x,y).

recursive step: choose an integer z such that $x < z < y$, execute rMAP(x,z) and rMAP(z,y) to obtain $\mathcal{M}(x,z)$ and $\mathcal{M}(z,y)$, and combine $\mathcal{M}(x,z)$ and $\mathcal{M}(z,y)$ to compute $\mathcal{M}(x,y)$ using Corollary 5.2.3.

The choice of the non-recursive or recursive step, and the choice of z in the recursive step are made so that the total decoding complexity is the smallest. As discussed later, the choice is uniquely determined for the code and the section boundaries x and y. □

Algorithm 5.2.2: dMAP(x,y)

This algorithm is to compute $\mathcal{M}(x,y)$ directly. Let $K = \mathcal{I}(C) \cap \{x+1, \ldots, y\}$ be the set of information bit positions between the time-x and time-y. The algorithm first computes

$$q_j[0] = \begin{cases} \Pr_j(0) \cdot \Pr(\boldsymbol{r}[j]|0) & \text{if } j \in K \\ \Pr(\boldsymbol{r}[j]|0) & \text{otherwise,} \end{cases}$$

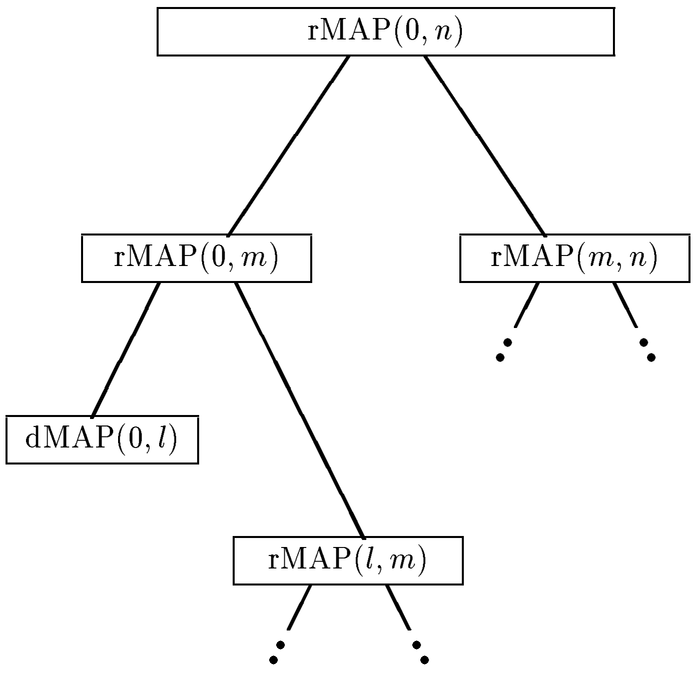


Figure 5.1. A MAP decoding for C by the rMAP algorithm.

$$q_j[1] \;=\; \begin{cases} \Pr_j(1)\cdot\Pr(\boldsymbol{r}[j]|1) & \text{if } j \in K \\ \Pr(\boldsymbol{r}[j]|1) & \text{otherwise} \end{cases}$$

for $x < j \le y$, just as same as the BCJR algorithm (Chapter 4). Then it computes

$$\prod_{x<j\le y} \Pr_j(\boldsymbol{v}[j]) \cdot \Pr(\boldsymbol{r}[j]|\boldsymbol{v}[j]) = \prod_{x<j\le y} q_j(\boldsymbol{v}[j]) \tag{5.6}$$

for all vectors $\boldsymbol{v}$ in $p_{xy}(C)$. Next, MAP(D_{xy}, i, b) is computed by summing the probabilities (5.6) for each coset in $D_{xy} \in L_{xy}$, bit position i with $x < i \le y$ and symbol $b \in GF(2)$. $\square$

A MAP table for C is constructed as Figure 5.1, and thus the MAP decoding for C is done.

5.3 Evaluation of the Algorithm

5.3.1 Complexity of the algorithm

The decoding complexity of the rMAP algorithm is investigated. As the measure of complexity, we take the number of multiplications of probabilities necessary for decoding, since multiplications of probabilities are the most costly operation in the MAP decoding.

First, we consider the complexity of the non-recursive step of rMAP algorithm (i.e. dMAP algorithm). Let $\phi^d(x, y)$ be the number of multiplications necessary for computing $\mathcal{M}(x, y)$ by using dMAP algorithm. It is easily understood that

$$\phi^d(x, y) = 2|K| + (y - x - 1) \cdot |p_{xy}(C)|$$

where the first term is to compute the values of q_j's, and the second term is to compute (5.6).

The complexity of the recursive step of rMAP is little more complicated. Actually, it is the sum of the complexity necessary for constructing $\mathcal{M}(x, z)$ and $\mathcal{M}(z, y)$, and the complexity necessary for combining the MAP tables. Let $\phi^r_z(x, y)$ be the number of multiplications necessary for combining the tables. Note that the function is subscripted with z since the number of multiplications is different for different choice of z.

A value MAP(D_{xy}, i, b) in a cell of a MAP table for $D_{xy} \in L_{xy}$ is determined by using the equations in Corollary 5.2.3. Hence we need $m_z(x, y)$ multiplications of probabilities for computing one cell of the table. Since a table has $2(y - x)$

cells, we need $m_z(x,y) \times 2(y-x)$ multiplications to compute a MAP table in a straightforward way.

However, by using the properties of the MAP table, we can reduce the number of multiplications as follows. First of all, remark that $\mathrm{MAP}(D_{xy}, i, 1)$ with $x < i \leq y$ can be computed by subtracting $\mathrm{MAP}(D_{xy}, i, 0)$ from $|\mathrm{MAP}(D_{xy})|$, and no multiplication is necessary. Since

$$\mathrm{MAP}(D_{xy}) = \sum_{i=1}^{m_z(x,y)} MAP(D^i_{x,z})MAP(D^i_{z,y}),$$

$m_z(x,y)$ multiplications are necessary to compute $\mathrm{MAP}(D_{xy})$. Therefore we need $m_z(x,y) + m_z(x,y) \times (y-x)$ multiplications in total to determine all values in a MAP table.

Moreover, when cosets D_{xz} and D_{zy} contain only small number of vectors, then more efficient combination algorithm is applicable. Table 5.2 shows such special cases. For example, if $|D_{xz}| = |D_{zy}| = 1$ (case 1), then $D_{xz} \circ D_{zy}$ also contains only one vector. In this case only $m_z(x,y)$ multiplications are sufficient for making a MAP table. Similarly, if $|D_{xz}| = 2$ and $|D_{zy}| = 1$ (or $|D_{xz}| = 1$ and $|D_{zy}| = 2$) (case 2), then $D_{xz} \circ D_{zy}$ contains only two vectors. In this case only $2m_z(x,y)$ multiplications are sufficient for making a MAP table. If $|D_{xz}| = 2$ and $|D_{zy}| = 2$ (case 3), then $D_{xz} \circ D_{zy}$ contains four vectors. Using the relation (5.5), probabilities for those four vectors can be computed by only three multiplications. Thus $3m_z(x,y)$ multiplications are sufficient for making a MAP table.

Since there are $|L_{xy}|$ different MAP tables, the number of multiplications

Table 5.2. Special cases.

<u>case 1</u>

$$\begin{tikzcd}
D_{xz} = \{l\}, D_{zy} = \{r\} \arrow[r, Rightarrow] \arrow[d] & \begin{array}{|c|c|c|}\hline 0 & 0 & a \\ \hline a & a & 0 \\ \hline\end{array} \circ \begin{array}{|c|c|c|}\hline b & 0 & b \\ \hline 0 & b & 0 \\ \hline\end{array}, \arrow[d] \\
D_{xz} \circ D_{zy} = \{lr\} \arrow[r, Rightarrow] & \begin{array}{|c|c|c|c|c|c|}\hline 0 & 0 & c & c & 0 & c \\ \hline c & c & 0 & 0 & c & 0 \\ \hline\end{array}
\end{tikzcd}$$

where $\quad c = a \cdot b.$

<u>case 2</u>

$$\begin{tikzcd}
D_{xz} = \{l_1, l_2\}, D_{zy} = \{r\} \arrow[r, Rightarrow] \arrow[d] & \begin{array}{|c|c|c|}\hline a & 0 & a+b \\ \hline b & a+b & 0 \\ \hline\end{array} \circ \begin{array}{|c|c|c|}\hline c & 0 & c \\ \hline 0 & c & 0 \\ \hline\end{array}, \arrow[d] \\
D_{xz} \circ D_{zy} = \{l_1 r, l_2 r\} \arrow[r, Rightarrow] & \begin{array}{|c|c|c|c|c|c|}\hline d & 0 & d+e & d+e & 0 & d+e \\ \hline e & d+e & 0 & 0 & d+e & 0 \\ \hline\end{array}
\end{tikzcd}$$

where $\quad d = a \cdot c,$

$e = b \cdot c.$

<u>case 3</u>

$$\begin{tikzcd}
D_{xz} = \{l_1, l_2\}, D_{zy} = \{r_1, r_2\} \arrow[r, Rightarrow] \arrow[d] & \begin{array}{|c|c|c|}\hline a & 0 & a+b \\ \hline b & a+b & 0 \\ \hline\end{array} \circ \begin{array}{|c|c|c|}\hline c+d & d & c \\ \hline 0 & c & d \\ \hline\end{array}, \arrow[d] \\
D_{xz} \circ D_{zy} = \{l_1 r_1, l_1 r_2, l_2 r_1, l_2 r_2\} \arrow[r, Rightarrow] & \begin{array}{|c|c|c|c|c|c|}\hline e & 0 & e+f & g+h & h & g \\ \hline f & e+f & 0 & 0 & g & h \\ \hline\end{array}
\end{tikzcd}$$

where $\quad e = a \cdot (c+d),$

$f = b \cdot (c+d),$

$g = (a+b) \cdot c,$

$h = e + f - g.$

---

necessary for combining the MAP tables is,

$$\phi_z^r(x,y) = \begin{cases} |L_{xy}| \cdot m_z(x,y) & \text{for case 1} \\ |L_{xy}| \cdot 2m_z(x,y) & \text{for case 2} \\ |L_{xy}| \cdot 3m_z(x,y) & \text{for case 3} \\ |L_{xy}| \cdot (y-x+1)m_z(x,y) & \text{otherwise.} \end{cases}$$

Thus the number of multiplications necessary for executing the recursive step is the sum of $\phi_z^r(x,y)$ and the number of multiplications necessary for constructing $\mathcal{M}(x,z)$ and $\mathcal{M}(z,y)$.

Define

$$\phi(x,y) \overset{\triangle}{=} \begin{cases} \phi^d(x,y) & \text{if } y-x=1, \\ \min\{\phi^d(x,y), & \\ \quad \min\limits_{x<z<y}\{\phi_z^r(x,y)+\phi(x,z)+\phi(z,y)\}\} & \\ & \text{otherwise.} \end{cases}$$

The function $\phi(x,y)$ gives the smallest number of multiplications necessary for constructing $\mathcal{M}(x,y)$, and consequently $\phi(0,n)$ gives the number of multiplications necessary for the MAP decoding. Choosing the optimum dividing points, such as z that gives minimum $\phi_z^r(x,y)+\phi(x,z)+\phi(z,y)$, is done by a dynamic programming. It calculates $\phi(a,b)$'s for shorter sections first, then choose the best dividing point for longer sections by leveraging them. Since this algorithm is equivalent to filling half of the $(n+1)\times n$-size table, each of whose elements is $\phi(x,y)$ for $0 \le x < y \le n$, respectively, and each element is derived by comparing computation at most n-times, the complexity of this dynamic programming is $O(n^3)$. Table 5.3 shows the complexity of the rMAP algorithm and that of the BCJR algorithm for some linear block codes. RM stands for the Reed-Muller codes and eBCH stands for the extended and permuted[9] BCH codes. Results

Table 5.3. The number of multiplications of probabilities necessary for decoding.

code	rMAP	BCJR	ratio
eBCH$(64, 7)_a$	736	10,920	6.7%
eBCH$(64, 10)_c$	3,200	77,224	4.1%
eBCH$(64, 16)_a$	124,288	3,073,000	4.0%
eBCH$(64, 18)_a$	493,952	12,255,208	4.0%
eBCH$(64, 24)_a$	694,656	5,963,752	11.6%
eBCH$(64, 30)_a$	42,338,688	214,433,768	19.7%
eBCH$(64, 36)_a$	105,056,640	108,527,592	96.8%
RM(64,22)	174,464	1,500,136	11.6%
RM(128,29)	10,799,872	185,000,936	5.8%

Table 5.4. The number of probabilities which must be recorded.

code	rMAP	BCJR	ratio
eBCH$(64, 7)_a$	769	5,272	14.6%
eBCH$(64, 10)_c$	3,009	37,080	8.1%
eBCH$(64, 16)_a$	107,969	1,438,200	7.5%
eBCH$(64, 18)_a$	428,481	5,734,392	7.5%
eBCH$(64, 24)_a$	321,985	2,580,472	12.5%
eBCH$(64, 30)_a$	17,500,609	86,704,120	20.2%
eBCH$(64, 36)_a$	4,524,481	33,357,816	13.6%
RM(64,22)	81,345	649,720	12.5%
RM(128,29)	4,672,385	86,143,480	5.4%

for other codes appear in the next chapter 6. For low-rate codes, the proposed rMAP algorithm is much more efficient than the BCJR algorithm. However, for relatively high-rate codes, the BCJR algorithm seems better than the rMAP algorithm. This is because that $m_z(x, y)$ and $|L_{xy}|$ tend to be large in high-rate codes. The next chapter presents the improved algorithm of the rMAP algorithm, which is never inefficient than the BCJR algorithm if you mind just the computational complexity. Or we may use dual codes instead of such high-rate codes, as discussed in [11] to get around this matter.

5.3.2 Other advantages

The rMAP algorithm has a number of advantages for implementing efficient MAP decoders. For example, the trellis diagram is no longer necessary in the rMAP algorithm. This property reduces the space complexity of the decoder. As a measure of the space complexity, consider how many probabilities must be recorded for the MAP decoding. In the case of the BCJR algorithm, the number is approximately double of the number of states of the trellis diagram since we need to record probabilities α and β for each state. In the case of the proposed rMAP algorithm, the total number of cells of MAP table in $\mathcal{M}(x, y)$ is at most $2(y - x)|L_{xy}|$, which can be reduced by using the properties introduced at the derivation of $\phi_z^r(x, y)$. The total number of cells necessary for executing the proposed algorithm is obtained by summing up the number of cells of MAP tables at each divide-and-conquer stage. Table 5.4 shows the number of probabilities which must be recorded for the MAP decoding. We can see that the space complexity is considerably improved in the rMAP algorithm. This improvement enables VLSI implementation of the decoder much easier.

Furthermore, the divide-and-conquer structure of the algorithm makes it suitable for parallel and pipeline processing, which speeds up decoding. Another advantage of using the algorithm is that the decoding delay is much more smaller than that of the BCJR algorithm since the rMAP algorithm does not require time consuming backward recursion. This advantage is significant when the decoder is used iteratively, as in turbo decoding. In addition, the rMAP algorithm is used in another efficient MAP decoding algorithm proposed in the next chapter, and help it to reduce the complexity.

5.4 Discussion

An efficient algorithm for MAP decoding is presented. The algorithm is devised based on the structural properties of linear codes, and has a number of advantages for implementation of efficient MAP decoder. It is also studied that the algorithm is not efficient for high-rate codes. Further investigation had been necessary for improving the results for those high-rate codes, and it has done. The improvement on this problem is seen in the next chapter 6, that is a hybrid of the BCJR and the rMAP algorithms.

Chapter 6

Hybrid Algorithm

6.1 Introduction to Hybrid Algorithm

The rMAP algorithm proposed in the previous chapter has a number of advantages, however, it is not appropriate to decode high-rate codes. In the other words, the original BCJR algorithm is more appropriate for such high-rate codes than the rMAP algorithm, at least in the case we only count the computational complexity. For such a reason, we came to the idea of a hybrid.

The algorithm proposed in this chapter is a hybrid of the BCJR and the rMAP algorithms. This hybrid algorithm is making the most of two algorithms. The weak point of rMAP algorithm is covered by being coupled with a BCJR-like way of calculation. That is making use of the idea of section trellis diagram, which reduces both the space and time complexity.

We can make any section trellis diagram at will with the section boundaries. The complexity of the proposed hybrid algorithm highly depends on given section boundaries, and it is not good idea to find the optimum sectionalization (which minimizes the complexity) by a exhaustive search. Author found that the approach by Lafourcade et al.[13], which is an efficient algorithm to find the

the optimum sectionalization of trellis for maximum likelihood decoding, is even applicable to this problem.

6.2 Hybrid Algorithm

6.2.1 Overview

Roughly speaking, the proposed algorithm is a BCJR-like algorithm on a section trellis diagram. Let $B = \{b_0, b_1, \ldots, b_m\}$ be a section boundaries and consider the trellis diagram T of the code C with boundaries at B. The algorithm consists of two steps. First, it decides branch probabilities of all branches of T. In other words, it constructs MAP tables for all cosets $D \in L_{b_j b_{j+1}}$ with $0 \leq j \leq m-1$. Remark that, if the MAP table has been constructed for $D \in L_{b_j b_{j+1}}$, then we can easily obtain MAP(D) and MAP(D, i, b) for any $b_j < i \leq b_{j+1}$ and $b \in GF(2)$. The construction method of MAP tables is introduced in Section 6.2.2. The second part of the algorithm is the construction of the MAP table for the code C. A BCJR-like algorithm is used for the construction, and its detail is discussed in Section 6.2.3.

6.2.2 rMAP algorithm

A recursive algorithm for the MAP decoding is proposed in previous chapter. The algorithm rMAP(x, y) is used as a part of the proposed hybrid algorithm in this chapter. The behavior of the rMAP algorithm used here is identical to the one proposed in the previous chapter, however, it is used not for the entire code but for cosets.

6.2.3 BCJR-like algorithm

This is not identical to the original BCJR algorithm, however, the general idea of the algorithm is very similar to that so we call this a BCJR-like algorithm. Actually, it is possible to consider this is a BCJR algorithm on the section trellis diagram as a substitute of the entire trellis diagram.

For each state σ of T, define $\alpha(\sigma) = \mathrm{MAP}(L(\sigma_0, \sigma))$ and $\beta(\sigma) = \mathrm{MAP}(L(\sigma, \sigma_n))$ where σ_0 is the initial state and σ_n is the final state. The computation of α's is done recursively using the section trellis diagram and branch probabilities. Assume that $\alpha(\sigma)$ has been computed for every state $\sigma \in \Sigma_{b_j}$ with $j < t$, and now we are going to compute $\alpha(\sigma)$ for a state $\sigma \in \Sigma_{b_t}$. Let $\rho_1, \rho_2, \ldots, \rho_m \in \Sigma_{b_{t-1}}$ be states which are connected to σ in the section trellis diagram. We can write

$$L(\sigma_0, \sigma) = \bigcup_{s=1}^{m} L(\sigma_0, \rho_s) \circ L(\rho_s, \sigma).$$

By using Lemmas 5.2.1 and 5.2.2,

$$\begin{aligned} \alpha(\sigma) &= \mathrm{MAP}(L(\sigma_0, \sigma)) \\ &= \sum_{s=1}^{m} \mathrm{MAP}(L(\sigma_0, \rho_s)) \cdot \mathrm{MAP}(L(\rho_s, \sigma)) \\ &= \sum_{s=1}^{m} \alpha(\rho_s) \cdot \mathrm{MAP}(L(\rho_s, \sigma)). \end{aligned}$$

Since $\mathrm{MAP}(L(\rho_s, \sigma))$ is easily computed from the MAP table for $L(\rho_x, \sigma)$, $\alpha(\sigma)$ is computed recursively using the section trellis diagram and branch probabilities. Similarly, $\beta(\sigma)$ is computed as

$$\beta(\sigma) = \sum_{s=1}^{m} \mathrm{MAP}(L(\sigma, \rho_s)) \cdot \beta(\rho_s)$$

where $\rho_1, \rho_2, \ldots, \rho_m \in \Sigma_{b_{t+1}}$ are states which are connected to σ.

By using α's and β's, $\mathrm{MAP}(C, i, b)$ is computed as follows. Let x be the largest boundary in B which is smaller than i, and let y be the smallest boundary in B

which equals to or more than i. Hence $x < i \leq y$. Remark that

$$C^{i/b} = \bigcup_{\sigma_x \in \Sigma_x} \bigcup_{\sigma_y \in \Sigma_y} L(\sigma_0, \sigma_x) \circ L(\sigma_x, \sigma_y)^{i/b} \circ L(\sigma_y, \sigma_n).$$

By using Lemmas 5.2.1, 5.2.2 and (5.5), we have

$$\begin{aligned} \mathrm{MAP}(C, i, b) &= \sum_{\sigma_x \in \Sigma_x} \sum_{\sigma_y \in \Sigma_y} \{\mathrm{MAP}(L(\sigma_0, \sigma_x)) \\ &\qquad \cdot \mathrm{MAP}(L(\sigma_x, \sigma_y), i, b) \cdot \mathrm{MAP}(L(\sigma_y, \sigma_n))\} \\ &= \sum_{\sigma_x \in \Sigma_x} \sum_{\sigma_y \in \Sigma_y} \alpha(\sigma_x) \cdot \mathrm{MAP}(L(\sigma_x, \sigma_y), i, b) \cdot \beta(\sigma_y). \qquad (6.1) \end{aligned}$$

Figure 6.1 visualizes how the MAP table of C is constructed by (6.1), where $\sigma_x^i \in \Sigma_x, \sigma_y^j \in \Sigma_y$ with $1 \leq i \leq m_l, 1 \leq j \leq m_r$ and m_l, m_r are constants.

6.3 Complexity of the Algorithm

The decoding complexity of the proposed algorithm is investigated in this section. As the measure of complexity, we take the number of multiplications of probabilities necessary for decoding, since multiplications of probabilities are the most costly operation in the MAP decoding.

6.3.1 Computation of branch probabilities

We briefly review the complexity of the rMAP procedure. In Chapter 5, it is shown that the complexity of rMAP(x, y) is given as

$$\phi^r(x, y) \triangleq \begin{cases} \phi^d(x, y) & \text{if } y - x = 1, \\ \min\Big\{\phi^d(x, y), & \\ \qquad \min\limits_{x<z<y} \{\phi_z^c(x, y) + \phi^r(x, z) + \phi^r(z, y)\}\Big\} & \text{otherwise,} \end{cases} \qquad (6.2)$$

where $\phi^d(x, y) = 2|K| + (y - x - 1)|p_{xy}(C)|$ is the complexity for computing MAP tables directly, and $\phi_z^c(x, y)$ is the complexity for combining MAP tables, which

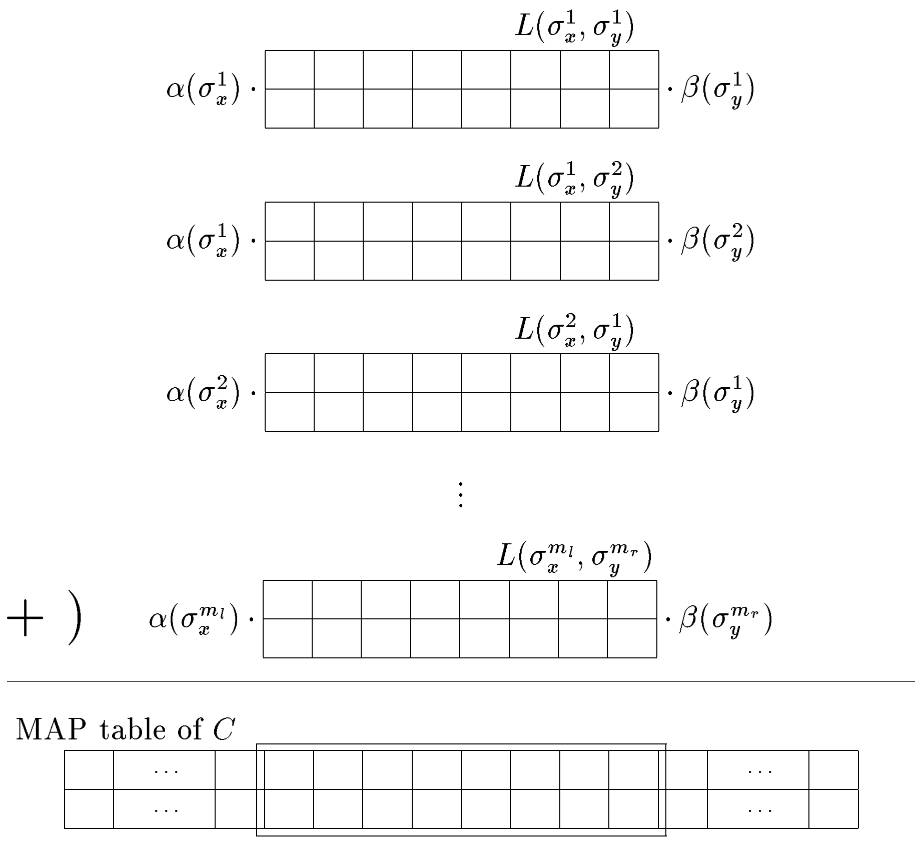

$L(\sigma_x^1, \sigma_y^1)$
$\alpha(\sigma_x^1) \cdot$
$\cdot \beta(\sigma_y^1)$
$L(\sigma_x^1, \sigma_y^2)$
$\alpha(\sigma_x^1) \cdot$
$\cdot \beta(\sigma_y^2)$
$L(\sigma_x^2, \sigma_y^1)$
$\alpha(\sigma_x^2) \cdot$
$\cdot \beta(\sigma_y^1)$
$\vdots$
$L(\sigma_x^{m_l}, \sigma_y^{m_r})$
$+)$
$\alpha(\sigma_x^{m_l}) \cdot$
$\cdot \beta(\sigma_y^{m_r})$
MAP table of C
$\cdots$
$\cdots$
$\cdots$
$\cdots$

Figure 6.1. The construction of MAP table of C

is defined as

$$\phi_z^c(x,y) = \begin{cases} |L_{xy}| \cdot m_z(x,y) & |D_{xy}| = 1 \\ |L_{xy}| \cdot 2m_z(x,y) & |D_{xy}| = 2 \\ |L_{xy}| \cdot 3m_z(x,y) & |D_{xy}| = 4 \\ |L_{xy}| \cdot (y-x+1)m_z(x,y) & \text{otherwise;} \end{cases}$$

The function $\phi^r(x,y)$ gives the smallest number of multiplications necessary for computing branch probabilities between time-x and time-y. Hence, the complexity for computing all branch probabilities is the sum of (6.2) of each section.

$$\phi^R = \sum_{j=0}^{m-1} \phi^r(b_j, b_{j+1}). \tag{6.3}$$

6.3.2 Computation of α's and β's

Note that no multiplication is necessary for computing $\alpha(\sigma)$ for $\sigma \in \Sigma_{b_0} \cup \Sigma_{b_1}$, because those values are obviously defined. Also we do not have to compute $\alpha(\sigma_{b_n})$ since it is not used for constructing the MAP table for C. For computing α-values for all states in $\Sigma_{b_{j+1}}$ with $j \geq 1$, one multiplication is needed for each branch between time-b_j and time-b_{j+1}. Since there are $2^{k(C)-k(C_{0b_j})-k(C_{b_jb_{j+1}})-k(C_{b_jn})}$ branches in this section, $2^{k(C)-k(C_{0b_j})-k(C_{b_jb_{j+1}})-k(C_{b_jn})}$ multiplications are necessary for computing α-values at time-b_{j+1}. Then, the total complexity for computing α-values are

$$\phi^\alpha = \sum_{j=1}^{m-2} 2^{k(C)-k(C_{0b_j})-k(C_{b_jb_{j+1}})-k(C_{b_jn})}.$$

It is easily shown that the complexity for computing β-values equals to ϕ^α, and denoted as ϕ^β. Moreover, if we define

$$\phi^\alpha_{b_j,b_{j+1}} = \phi^\beta_{b_j,b_{j+1}} = \begin{cases} 0 & \text{if } b_j = 0 \text{ or } b_{j+1} = n \\ 2^{k(C)-k(C_{0b_j})-k(C_{b_jb_{j+1}})-k(C_{b_jn})} & \text{otherwise,} \end{cases}$$

then

$$\phi^{\alpha} = \sum_{j=0}^{m-1} \phi^{\alpha}_{b_j,b_{j+1}} \tag{6.4}$$

$$\phi^{\beta} = \sum_{j=0}^{m-1} \phi^{\beta}_{b_j,b_{j+1}}. \tag{6.5}$$

6.3.3 Construction of the MAP table for C

In this section, we discuss the number of multiplications necessary for computing (6.1) for every bit position i and every symbol b. Since the equation (6.1) contains many terms which have a common factor, we can reduce the number of multiplications by using the distributivity property of multiplication operations. Consider a coset $D \in L_{xy}$. It is known that there are

$$m_{xy} \stackrel{\triangle}{=} 2^{k(C)-k(p_{xy}(C))-k(C_{0x})-k(C_{yn})}$$

pairs of states $\sigma_x \in \Sigma_x$ and $\sigma_y \in \Sigma_y$ such that $L(\sigma_x, \sigma_y) = D$. Let $P(D)$ be the set of such pairs $\langle \sigma_x, \sigma_y \rangle$. The equation (6.1) can be transformed as follows.

$$\begin{aligned}
&\sum_{\sigma_x \in \Sigma_x} \sum_{\sigma_y \in \Sigma_y} \alpha(\sigma_x) \cdot \mathrm{MAP}(L(\sigma_x, \sigma_y), i, b) \cdot \beta(\sigma_y) \\
&= \sum_{D \in L_{xy}} \sum_{\langle \sigma_x, \sigma_y \rangle \in P(D)} \alpha(\sigma_x) \cdot \mathrm{MAP}(D, i, b) \cdot \beta(\sigma_y) \\
&= \sum_{D \in L_{xy}} \mathrm{MAP}(D, i, b) \cdot \sum_{\langle \sigma_x, \sigma_y \rangle \in P(D)} \alpha(\sigma_x) \cdot \beta(\sigma_y).
\end{aligned}$$

This means that the number of multiplications can be reduced by computing $\theta \stackrel{\triangle}{=} \sum_{\langle \sigma_x, \sigma_y \rangle \in P(D)} \alpha(\sigma_x) \cdot \beta(\sigma_y)$ first. Furthermore, remark that the sum θ is independent from the bit position i and the symbol b. Therefore, we can use the sum θ for the computation for every position i with $x < i \leq y$ and every symbol b. Actually, to compute $\mathrm{MAP}(C, i, b)$ for every position i with $x < i \leq y$ and every symbol b, we need $m_{xy} + (y - x) + 1$ multiplications for each coset D in L_{xy}, where the first term m_{xy} is to compute the sum θ, the second term $y - x$ is to compute $\theta \cdot \mathrm{MAP}(D, i, 0)$ for every i with $x < i \leq y$, and the third term 1 is to compute $\theta \cdot \mathrm{MAP}(D)$. Remark that we do not have to count the number of multiplications for computing $\theta \cdot \mathrm{MAP}(D, i, 1)$ since it is obtained by subtracting $\theta \cdot \mathrm{MAP}(D, i, 0)$ from $\theta \cdot \mathrm{MAP}(D)$. Since there are $|L_{xy}|$ different cosets, the total number of multiplications necessary for computing $\mathrm{MAP}(C, i, b)$ for every i with $x < i \leq y$ and every symbol b is $|L_{xy}| \cdot (m_{xy} + (y - x) + 1)$. Furthermore, when $x = b_0 = 0$ or $y = b_m = n$, the term m_{xy} is not necessary since we do not have to multiply α and β. Summarizing the above discussions, we need

$$\phi^\gamma = \sum_{j=0}^{m-1} \phi^\gamma_{b_j, b_{j+1}} \qquad (6.6)$$

multiplications for constructing the MAP table for the code, where $\phi^\gamma(b_j, b_{j+1})$ is defined as follows.

$$\phi^\gamma_{b_j, b_{j+1}} = \begin{cases} 0 & \text{if } b_j = 0, b_{j+1} = n \\ |L_{b_0 b_1}| \cdot (b_1 - b_0) & \text{if } b_j = 0, b_{j+1} \neq n \\ |L_{b_{m-1} b_m}| \cdot ((b_m - b_{m-1}) + 1) & \text{if } b_j \neq 0, b_{j+1} = n \\ |L_{b_j b_{j+1}}| \cdot (m_{b_j b_{j+1}} + (b_{j+1} - b_j) + 1) & \text{otherwise.} \end{cases} \qquad (6.7)$$

Furthermore, if $D_{b_j b_{j+1}} \leq 2$, then the term $(b_{j+1} - b_j) + 1$ in (6.7) can be replaced by $|D_{b_j b_{j+1}}|$ since one multiplication for every vector in D_{xy} is sufficient.

Finally, the total complexity is the sum of (6.2), (6.4), (6.5) and (6.6) as follows.

$$\begin{aligned} \phi &= \phi^R + \phi^\alpha + \phi^\beta + \phi^\gamma \\ &= \sum_{j=0}^{m-1} \phi^r(b_j, b_{j+1}) + \sum_{j=0}^{m-1} \phi^\alpha_{b_j, b_{j+1}} + \sum_{j=0}^{m-1} \phi^\beta_{b_j, b_{j+1}} + \sum_{j=0}^{m-1} \phi^\gamma_{b_j, b_{j+1}} \\ &= \sum_{j=0}^{m-1} (\phi^r(b_j, b_{j+1}) + \phi^\alpha_{b_j, b_{j+1}} + \phi^\beta_{b_j, b_{j+1}} + \phi^\gamma_{b_j, b_{j+1}}. \end{aligned}$$

6.3.4 Optimum sectionalization for the hybrid algorithm

The decoding complexity of the hybrid algorithm depends on the sectionalization of the trellis. Since there are 2^{n-1} different sectionalizations for a code with length n, it is not good idea to find the optimum sectionalization (which minimizes the complexity) by a exhaustive search. In this section, we consider more sophisticated and efficient way for finding the optimum sectionalization. Lafourcade et al. showed an efficient algorithm to find the optimum sectionalization of a trellis[13]. By their algorithm, the optimum sectionalization for the maximum likelihood decoding is found in $O(n^2)$. The object function which should be minimized is different between their case and our case, though, our problem is solved in a similar way. The complexity ϕ of the hybrid algorithm is expressed as follows.

$$\phi = \sum_{j=0}^{m-1} \phi_{b_j b_{j+1}},$$

where

$$\phi_{b_j b_{j+1}} = \begin{cases} \phi^r(b_j, b_{j+1}) & \text{if } b_j = 0, b_{j+1} = n \\ \phi^r(b_j, b_{j+1}) + |L_{b_j b_{j+1}}| \cdot (b_{j+1} - b_j) & \text{if } b_j = 0, b_{j+1} \neq n \\ \phi^r(b_j, b_{j+1}) + |L_{b_j b_{j+1}}| \cdot ((b_{j+1} - b_j) + 1) & \text{if } b_j \neq 0, b_{j+1} = n \\ \phi^r(b_j, b_{j+1}) + \phi^\alpha_{b_j b_{j+1}} + \phi^\beta_{b_j b_{j+1}} + \phi^\gamma_{b_j b_{j+1}} & \text{otherwise.} \end{cases}$$

The total complexity is the sum of complexities of all sections, and complexity of a section is independent from complexities of other sections. This property helps us finding the optimum sectionalization. For example, consider two slightly different sets of section boundaries $B = \{b_0, \ldots, b_i, b_{i+1}, \ldots, b_m\}$ and $B' = \{b_0, \ldots, b_i, b, b_{i+1}, \ldots, b_m\}$ where $b_0 = 0, b_m = n$ and b is an integer such that $b_i < b \leq b_{i+1}$. If the complexity for the sectionalization B is expressed as $\phi_{b_i b_{i+1}} + D$ with D a constant, then the complexity for the sectionalization B' is $\phi_{b_i b} + \phi_{b, b_{i+1}} + D$. Therefore, if $\phi_{b_i b} + \phi_{b, b_{i+1}} < \phi_{b_i b_{i+1}}$, then we should put a section boundary at time-b to reduce the complexity.

```
Algorithm 6.3.1: Min-COMP($x, y$)
   input:  integers $x$ and $y$ with $0 \leq x < y \leq n$
   output: $\phi^{\min}_{x,y}$
If $y - x = 1$, then execute the following non-recursive step. If $y - x > 1$, then execute the recursive step.

non-recursive step: $\phi^{\min}_{xy} = \phi_{xy}$.

recursive step: choose the minimum value in $\{\phi_{xy}\} \cup \{\phi_{xz} + \phi^{\min}_{zy} : x < z < y\}$.
```

$\square$

Remark that y should be fixed as n (or x should be fixed as 0), for the essential purpose of this algorithm. This algorithm derives z for each x $(0 \leq x < n)$. Since comparing calculations are done at most n-times for each, this algorithm works in $O(n^2)$.

Execute the algorithm Min-COMP$(0, n)$ to have $\phi^{\min}_{0,n}$, then the optimum sectionalization can be obtained by tracing the computation of $\phi^{\min}_{0,n}$ in a reverse way:

$$\begin{aligned}
\phi^{\min}_{0,n} &= \phi_{0,t_1} + \phi^{\min}_{t_1 n} \\
&= \phi_{0,t_1} + \phi_{t_1 t_2} + \phi^{\min}_{t_2 n} \\
&= \phi_{0,t_1} + \phi_{t_1 t_2} + \phi_{t_2 t_3} + \phi^{\min}_{t_3 n} \\
&\vdots \\
&= \phi_{0,t_1} + \phi_{t_1 t_2} + \phi_{t_2 t_3} + \cdots + \phi_{t_{m-1} n}.
\end{aligned}$$

In this case, the optimum sectionalization is

$$B = \{0, t_1, t_2, \ldots, t_{m-1}, n\}.$$

Table 6.1. The decoding complexity (i).

code*	BCJR	rMAP	hybrid	
RM(16,5)	676	120	120	B={0,16}
RM(16,11)	996	424	352	B={0,4,8,12,16}
RM(16,15)	228	186	136	B={0,2,4,6,8,10,12,14,16}
RM(32,6)	2,724	304	304	B={0,32}
RM(32,16)	25,572	5,888	4,288	B={0,8,16,24,32}
RM(32,26)	4,708	5,840	2,080	B={0,4,8,16,24,28,32}
RM(32,31)	484	506	296	B={0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32}
RM(64,7)	10,916	736	768	B={0,32,64}
RM(64,22)	1,500,132	174,464	116,096	B={0,16,32,48,64}
RM(64,42)	2,197,476	4,492,672	529,792	B={0,8,16,32,48,56,64}
RM(64,57)	20,324	55,200	10,528	B={0,4,8,16,24,32,40,48,56,60,64}

If $\phi_{0,n}^{\min} = \phi_{0,n} = \phi^r(0,n)$, then $B = \{0,n\}$. This means that the code should not be sectionalized. The rMAP algorithm should be used to construct the whole MAP table for C to minimize the complexity.

6.3.5 Results

Tables 6.1–6.3 show the minimum complexity and the optimum sectionalization for some well-known block codes. In this thesis, the number of multiplication of probabilities as the measure of complexity. RM(n,k) and eBCH$(n,k)_p$ stand for the Reed-Muller code and the extended BCH code with p-type permutation, respectively. The results show that the proposed algorithm needs smaller computational complexity than the BCJR algorithm. The proposed algorithm is especially efficient for low-rate codes. And the results also show that the

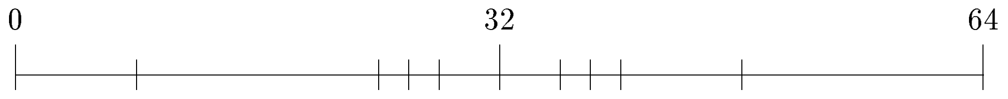
Table 6.2. The decoding complexity (ii).

code*	BCJR	rMAP	hybrid	
eBCH(64,7)$_a$	10,916	736	736	B={0,64}
eBCH(64,7)$_b$	10,916	736	736	B={0,64}
eBCH(64,7)$_c$	10,916	736	736	B={0,64}
eBCH(64,10)$_a$	84,964	4,224	4,224	B={0,64}
eBCH(64,10)$_b$	84,964	4,224	4,224	B={0,64}
eBCH(64,10)$_c$	77,220	3,200	3,200	B={0,64}
eBCH(64,16)$_a$	3,072,996	124,288	124,288	B={0,64}
eBCH(64,16)$_b$	2,860,004	120,192	120,192	B={0,64}
eBCH(64,16)$_c$	3,072,996	124,288	124,288	B={0,64}
eBCH(64,18)$_a$	12,255,204	493,952	493,952	B={0,64}
eBCH(64,18)$_b$	10,675,172	468,352	468,352	B={0,64}
eBCH(64,18)$_c$	12,255,204	493,952	493,952	B={0,64}
eBCH(64,24)$_a$	5,963,748	694,656	461,184	B={0,16,32,48,64}
eBCH(64,24)$_b$	4,531,172	669,056	411,008	B={0,16,32,48,64}
eBCH(64,24)$_c$	5,963,748	694,656	461,184	B={0,16,32,48,64}
eBCH(64,30)$_a$	214,433,764	42,338,688	25,430,400	B={0,16,32,48,64}
eBCH(64,30)$_b$	164,102,116	42,338,688	25,430,400	B={0,16,32,48,64}
eBCH(64,30)$_c$	113,770,468	42,338,688	25,430,400	B={0,16,32,48,64}

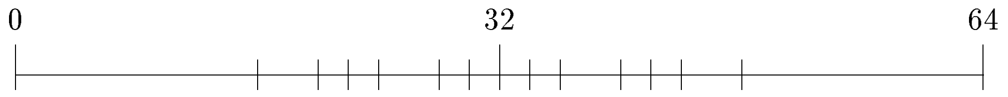
Table 6.3. The decoding complexity (iii).

code*	BCJR	rMAP	hybrid	
eBCH(64,36)$_a$	108,527,588	105,056,640	44,173,712	B={0,8,24,26,28,32,36,38,40,48,64}
eBCH(64,36)$_b$	108,527,588	105,056,640	44,173,712	B={0,8,24,26,28,32,36,38,40,48,64}
eBCH(64,36)$_c$	56,098,788	105,056,705	20,579,792	B={0,16,20,22,24,28,32,36,40,42,44,48,64}
eBCH(64,39)$_a$	394,788,836	831,063,352	201,591,184	B={0,8,24,26,28,32,36,38,40,48,64}
eBCH(64,39)$_b$	461,897,700	839,649,664	251,922,832	B={0,8,24,26,28,32,36,38,40,48,64}
eBCH(64,39)$_c$	107,479,012	444,744,424	58,328,480	B={0,16,20,22,24,28,30,32,34,36,40,42,44,48,64}
eBCH(64,45)$_a$	7,063,524	27,567,392	3,402,128	B={0,4,12,14,16,32,48,50,52,56,64}
eBCH(64,45)$_b$	8,112,100	35,334,528	3,402,128	B={0,4,12,14,16,32,48,50,52,56,64}
eBCH(64,45)$_c$	3,000,292	15,525,680	1,625,440	B={0,8,16,24,26,28,30,32,34,36,38,40,48,56,64}
eBCH(64,51)$_a$	1,132,516	6,082,856	775,504	B={0,4,12,14,16,18,20,22,24,26,28,30,32,34,36,38,40,42,44,46,48,50,52,56,64}
eBCH(64,51)$_b$	923,620	5,459,192	618,832	B={0,8,10,12,14,16,20,22,24,26,28,30,32,34,36,38,40,42,44,48,50,52,54,56,64}
eBCH(64,51)$_c$	1,034,212	5,786,848	701,808	B={0,4,12,14,16,19,21,23,25,27,29,32,35,37,39,41,43,45,48,50,52,56,64}
eBCH(64,57)$_a$	20,324	55,200	10,528	B={0,4,8,16,24,32,40,48,56,60,64}
eBCH(64,57)$_b$	20,324	55,200	10,528	B={0,4,8,16,24,32,40,48,56,60,64}
eBCH(64,57)$_c$	20,324	55,200	10,528	B={0,4,8,16,24,32,40,48,56,60,64}

eBCH(64,36)$_a$, $B = \{0, 8, 24, 26, 28, 32, 36, 38, 40, 48, 64\}$



eBCH(64,39)$_c$, $B = \{0, 16, 20, 22, 24, 28, 30, 32, 34, 36, 40, 42, 44, 48, 64\}$



eBCH(64,45)$_c$, $B = \{0, 8, 16, 24, 26, 28, 30, 32, 34, 36, 38, 40, 48, 56, 64\}$

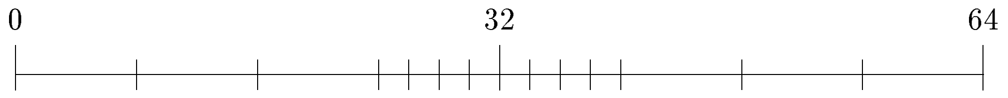


Figure 6.2. Optimum sectionalizations.

Table 6.4. The number of probabilities which must be recorded.

code	BCJR	hybrid	ratio
$\text{eBCH}(64,16)_b$	1,331,765	103,938	7.8%
$\text{eBCH}(64,18)_b$	4,944,437	402,946	8.1%
$\text{eBCH}(64,24)_b$	1,876,533	24,001	1.3%
$\text{eBCH}(64,30)_c$	43,712,565	756,161	1.7%
$\text{eBCH}(64,36)_c$	18,677,813	4,457,889	23.9%
$\text{eBCH}(64,39)_c$	30,605,365	11,666,817	38.1%
$\text{eBCH}(64,45)_c$	836,149	252,801	30.2%
$\text{eBCH}(64,51)_b$	243,253	109,857	45.2%
$\text{eBCH}(64,57)_a$	5,333	1,969	36.9%

equally sectionalized trellis is not necessarily the best structure for the proposed algorithm. Figure 6.2 visualizes the unequally but optimum sectionalization for eBCH(64,36)$_a$, eBCH(64,39)$_c$ and eBCH(64,45)$_c$.

In addition, the space complexity is reduced by using the hybrid algorithm. The space complexity is evaluated with the number of probabilities which must be recorded for the MAP decoding as same as in Chapter 5. Table 6.4 shows the space complexity of the hybrid algorithm for some extended BCH codes, compared with the BCJR algorithm. For low-rate codes, the set of section boundaries with $B = \{0, n\}$ is adopted, that means the pure rMAP algorithm is applied. Remark that the sectionalization for the hybrid algorithm is not optimized with the view of the space complexity but of the number of multiplications.

It will be interesting if we can compare the efficiency of the proposed algorithm to that of the algorithm in [16]. Unfortunately, the measure of the efficiency is little bit different between [16] and ours. In [16], they compute branch metrics by using some exponential operations, which reduces the number of multiplications.

If we substitute the exponential operations by multiplications, then our approach is more efficient than [16].

6.4 Discussion

Another efficient algorithm for the MAP decoding is proposed. The proposed algorithm is a hybrid of the conventional BCJR and the rMAP algorithms, which needs smaller decoding complexity than the BCJR algorithm.

Actually, the BCJR and rMAP algorithms can be regarded as special subcases of the proposed algorithm with the section boundaries $B = \{0, 1, 2, \ldots, n\}$ and $B = \{0, n\}$, respectively. Hence, the efficiency of the proposed algorithm is always better than or equal to that of the rMAP and BCJR algorithms for any code.

The proposed algorithm highly depends on the sectionalization of a trellis, and finding the optimum sectionalization is essential. A systematic way to find the optimum sectionalization for the hybrid MAP algorithm is investigated — the approach by Lafourcade et al. [13] is applicable to our problem.

The decoding complexity for some well-known linear block codes are calculated and showed; the effeciency of the proposed algorithm is confirmed. In addition, the proposed algorithm uses a section trellis diagram instead of the entire trellis diagram, which is easier to construct and implement.

Chapter 7

Conclusion

In this thesis, efficient algorithms for MAP decoding is presented.

Since the conventional algorithm for MAP decoding, which is called the BCJR algorithm, is considered incapable of applying to long practical codes, some kinds of modification has been requested.

The rMAP algorithm, presented in Chapter 5, needs no trellis diagram for decoding; constructing of the trellis diagram is both time and space consuming calculation and it is necessary for the BCJR algorithm. The number of calculation is reduced for the low-rate codes but rather increased for the high-rate codes.

The algorithm presented in Chapter 6 is the hybrid algorithm of the BCJR and the rMAP algorithm. This algorithm uses the section trellis diagram in place of the entire trellis diagram of the BCJR algorithm. The table of probabilities, called MAP table, is constructed on every branch of the section trellis diagram by using the rMAP algorithm and a BCJR-like algorithm is executed to the section trellis diagram. This hybrid algorithm has mainly two advantages. The first advantage is that the number of calculation is no wronger than both of the BCJR and the rMAP algorithms. The second one is the adoption of the section trellis diagram, that is more easy to construct than the entire trellis diagram.

The hybrid algorithm is suitable for all codes, and recommended to use instead of the conventional BCJR algorithm. And the rMAP algorithm is still recommendable for decoding the low-rate codes because it doesn't need even a section trellis diagram.

References

[1] L.R. Bahl, J. Cocke, F. Jelinek and J. Raviv: "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate," IEEE Transactions on Information Theory, **20**, 2, pp.284–287 (1974), and presented at the 1972 IEEE International Symposium on Information Theory, Asilomar, CA (January 1972).

[2] C. Berrow, A. Glavieux and P. Thitimajshima: "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo Codes," Proceedings of IEEE International Conference on Communications, Geneva, Switzerland, pp.1064–1070 (May 1993).

[3] J.A. Briffa and V. Buttigieg: "Interleavers for Unpunctured Symmetric Turbo Codes," Proceedings of the 2000 IEEE International Symposium on Information Theory, Sorrento, Italy, p.450 (June 2000).

[4] O.M. Collins, O.Y. Takeshita and D.J. Costello, Jr.: "Iterative Decoding of Non-Systematic Turbo-Codes," Proceedings of the 2000 IEEE International Symposium on Information Theory, Sorrento, Italy, p.172 (June 2000).

[5] D.J. Costello, Jr., J. Hagenauer, H. Imai and S.B. Wicker: "Applications of Error-Control Coding," IEEE Transactions on Information Theory, **44**, 6, pp.2351–2560 (October 1998).

[6] M. Elia, L. Rigazio and L. Blessent: "The Hamming(7,4) - Golay(23,12) Product Code and Its Iterative Decoding," Proceedings of the 1998 International Symposium on Information Theory and its Applications, Mexico City, Mexico, pp.636-638 (October 1998).

[7] M.P.C. Fossorier, M. Mihaljevic and H. Imai: "Reduced Complexity Iterative Decoding of Low Density Parity Check Codes Based on Belief Propagation," Proceedings of the 1998 International Symposium on Information Theory and its Applications, Mexico City, Mexico, pp.655-658 (October 1998).

[8] V. Franz and J.B. Anderson: "Reduced-Search BCJR Algorithm," Proceedings of 1997 International Symposium on Information Theory, Ulm, Germany, p.230 (June 1997).

[9] T. Fujiwara, H. Yamamoto, T. Kasami and S. Lin: "A Trellis-Based Recursive Maximum-Likelihood Decoding Algorithm for Binary Linear Block Codes," IEEE Transactions on Information Theory, **44**, 2, pp.714–729 (March 1998), also presented at the 33rd Allerton Conference on Communication Control and Computing, pp.700–709 (October 1995).

[10] R. Garello, G. Montorsi, S. Benedetto and G. Cancellieri: "Analysis of the Trellis Complexity of Interleavers and Turbo Codes," Proceedings of the 2000 IEEE International Symposium on Information Theory, Sorrento, Italy, p.65 (June 2000).

[11] J. Hagenauer, E. Offer and L. Papke: "Iterative Decoding of Binary Block and Convolutional Codes," IEEE Transactions on Information Theory, **42**, 2, pp.429–445 (March 1996).

[12] M. Isaka and H. Imai: "A Tutorial on 'Parallel Concatenated (Turbo) Coding,' 'Turbo (iterative) Decoding' and Related Topics," Technical Report of IEICE, IT98-51 (December 1998).

[13] A. Lafourcade and A. Vardy: "Optimal Sectionalization of a Trellis," IEEE Transactions on Information Theory, **42**, 3, pp. 689–703 (May 1996).

[14] J. Li and H. Imai: "Iterative Decoding for Majority Decodable Code," Proceedings of the 1998 International Symposium on Information Theory and its Applications, Mexico City, Mexico, pp.647-650 (October 1998).

[15] S. Lin and D.J. Costello, Jr.: *Error Control Coding: Fundamentals and Applications,* New Jersey: Prentice-Hall (1983).

[16] Y. Liu, M. Fossorier and S. Lin: "MAP Algorithms for Decoding Linear Block Codes Based on Sectionalized Trellis Diagram," IEEE Transactions on Communications, **48**, 4, pp.577–587 (April 2000), also presented at Globecom '98, Sydney, Australia, pp.562–566 (November 1998)

[17] P.L. McAdam, L. Welch and C. Weber: "M.A.P. Bit Decoding of Convolutional Codes," Proceedings of the 1972 International Symposium on Information Theory, Asilomar, CA (January 1972).

[18] V.S. Pless and W.C. Huffman, Editors: *Handbook of Coding Theory,* North-Holland, Netherlands, pp.1989–2117. (1998)

[19] H. Sadjadpour, N. Sloane, G. Nebe and M. Salehi: Interleaver Design for Turbo Codes Proceedings of the 2000 IEEE International Symposium on Information Theory, Sorrento, Italy, p.453 (June 2000).

[20] A. Vardy, J. Feigenbaum, G.D. Forney, B.H. Marcus and R.J. McEliece: "Introduction to the Special Issue on Codes and Complexity," IEEE Transactions on Information Theory, **42**, 6, pp.1649–1657 (November 1996).

[21] C.H. Wang and C.C. Chao: "Modified SOVA Decoding for Turbo Codes," Proceedings of the 1998 International Symposium on Information Theory and its Applications, Mexico City, Mexico, pp.643-646 (October 1998).

[22] S.B. Wicker: *Error Control Systems for Digital Communication and Storage,* New Jersey: Prentice-Hall (1995).