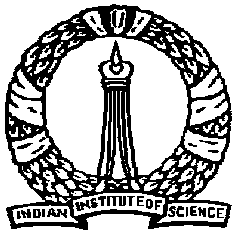
Unifying Views of Tail-Biting Trellises for Linear Block Codes

A Thesis
Submitted For the Degree of
Doctor of Philosophy
in the Faculty of Engineering

by

Aditya Vithal Nori



Department of Computer Science and Automation
Indian Institute of Science
Bangalore – 560 012

SEPTEMBER 2005

Abstract

This thesis presents new techniques for the construction and specification of linear tail-biting trellises. Tail-biting trellises for linear block codes are combinatorial descriptions in the form of layered graphs, that are somewhat more compact than the corresponding conventional trellis descriptions. Conventional trellises for block codes have a well understood underlying theory. On the other hand, the theory of tail-biting trellises appears to be somewhat more involved, though several advances in the understanding of the structure and properties of such trellises have been made in recent years. Of fundamental importance is the observation that a linear tail-biting trellis for a block code corresponds to a coset decomposition of the code with respect to a subcode. All constructions seem to use this property in some way; in other words, this is the unifying factor in all our constructions. The constructions yield the conventional trellis when the subcode is the whole code. We list the main contributions of this thesis.

(i) We generalize the well known Bahl-Cocke-Jelinek-Raviv construction for conventional trellises to obtain tail-biting trellises. We show that a linear tail-biting trellis for a linear block code $\mathcal{C}$ with block length n and dimension k over a finite field $\mathbb{F}_q$, can be constructed by specifying an arbitrary parity check matrix $\mathcal{H}$ for $\mathcal{C}$ along with a *displacement matrix* $\mathcal{D} \in \mathbb{F}_q^{(n-k)\times k}$.

(ii) We show that the displacement matrix $\mathcal{D}$ yields a coset decomposition of the code. We present a dynamic algorithm that starts with a generator matrix for the code $\mathcal{C}$ in trellis-oriented form, and computes the displacement matrix $\mathcal{D}$ for a minimal trellis in time $\mathcal{O}(n^k)$.

(iii) Forney has given an elegant algebraic characterization of a conventional trellis in terms of a quotient group of the code with respect to a special subgroup. Given a coset decomposition of the code, we propose a natural extension for tail-biting trellises, which considers a circular time axis instead of a linear one. The quotient group of interest is formed by judiciously using the properties of coset leaders.

(iv) There is an interesting connection between algebraic and combinatorial duality. For conventional trellises, it is known that the primal and dual trellises have the same

state-complexity profile. We give a simple and direct construction for a dual trellis in terms of the generator matrix $\mathcal{G}$, the parity check matrix $\mathcal{H}$ and the displacement matrix $\mathcal{D}$, and prove that the same property is true for tail-biting trellises.

(v) We give a new characterization of linear tail-biting trellises in terms of of an equivalence relation defined on a language derived from the code under consideration. The equivalence relation is intimately tied up with the coset decomposition of the code with respect to a subcode. In the context of formal language theory, this interpretation is an adaptation of the Myhill-Nerode theorem for regular languages.

Papers based on this Thesis

(i) Aditya Nori and Priti Shankar, Insights into compact graph descriptions of block codes, *SIAM Conference on Discrete Mathematics, San Diego, USA*, July 2002.

(ii) Aditya Nori and Priti Shankar, A BCJR-like labeling algorithm for tail-biting trellises, *IEEE International Symposium on Information Theory (ISIT), Yokohama, Japan*, July 2003.

(iii) Aditya Nori and Priti Shankar, Tail-biting trellises for linear codes and their duals, *Forty-First Annual Allerton Conference on Communication, Control and Computing, Allerton, IL, USA*, October 2003.

(iv) Aditya Nori and Priti Shankar, A coset construction for tail-biting trellises, *International Symposium on Information Theory and its Applications (ISITA), Parma, Italy*, October 2004.

(v) Aditya Nori and Priti Shankar, Unifying views of tail-biting trellis constructions for linear block codes, *IEEE Transactions on Information Theory* (accepted pending revision), July 2005.

To my wife, Sarada

my daughter, Madhavi

my parents, Govind and Asha

Contents

List of Figures

Chapter 1

Introduction

Our difficulty is not in the proofs, but in learning what to prove.

Emil Artin

Trellis decoding is the most pragmatic and well researched method of performing soft-decision decoding. The trellis was first introduced by Forney [For67] in 1967 as a conceptual device to explain the Viterbi algorithm for decoding convolutional codes. While the trellis has been studied in the context of convolutional codes for the first two decades since its inception, recent times have seen a significant amount of research devoted to the study of the trellis structure of linear block codes [Var98]. In 1974, Bahl, Cocke, Jelinek and Raviv [BCJR74] worked on a maximum aposteriori decoding algorithm for convolutional codes, and showed that linear block codes have combinatorial descriptions in the form of trellises. They proposed a labeling scheme that directly yielded the minimal conventional trellis for a linear block code. Solomon and van Tilborg [SvT79] introduced tail-biting trellises to construct block codes from convolutional codes.

Tail-biting trellises for linear block codes [CFV99] are combinatorial descriptions that are somewhat more compact than the corresponding conventional trellis descriptions. While conventional trellises for block codes have a well understood underlying theory [BCJR74, For88, KS95, Ksc96, Mas78, Mud88, Var98], the theory of tail-biting trellises appears to be somewhat more involved, though several advances in the understanding of the structure and properties of such trellises have been made in recent years [CFV99, KV02, KV03, RB99,

SKSR01, SDDR03, SB00].

Given a block code, it is known that there exists a unique minimal conventional trellis representing the code [McE96, Mud88], and there are several different algorithms for the construction of such a trellis [BCJR74, For88, KS95, Mas78]. The trellis simultaneously minimizes all measures of minimality. However, it is known that tail-biting trellises do not have such a property. Kötter and Vardy [KV02, KV03] have made a detailed study of the structure of linear tail-biting trellises and have also defined several measures of minimality. For tail-biting trellises, it is shown that different measures of minimality correspond to different partial orders on the set of linear trellises for a block code. An interesting property that is known for conventional trellises is that the minimal conventional trellis for a linear block code and its dual have identical state-complexity profiles [For88]. Kötter and Vardy have suggested a dual trellis construction using an intersection product operation [KV03]. They prove that the resulting dual trellis has a state-complexity profile identical to its primal counterpart only if the primal trellis is $\prec_\Theta$-minimal. We will now review the general organization of the thesis, and provide a road map to our main results.

1.1 Overview of the Thesis

In Chapter 2, we review the basic notions of error-correcting codes starting with the description of linear codes. In Section 2.2, we introduce the notion of a conventional trellis and then describe in chronological order the various constructions for minimal conventional trellises. We also define tail-biting trellises in this chapter.

In Chapter 3, we give a detailed introduction to the theory of linear tail-biting trellises. We first review various notions of trellis minimality and then describe Kötter and Vardy's *characteristic matrix* formulation for the minimal trellis problem. We present the overlayed structure of linear tail-biting trellises, where a linear tail-biting trellis for a linear block code $\mathcal{C}$ may be constructed by overlaying subtrellises obtained from a coset decomposition of $\mathcal{C}$. The coset decomposition in fact is the unifying property in all the constructions.

In Chapter 4, we generalize the BCJR construction for conventional trellises to obtain linear tail-biting trellises. We show that a linear tail-biting trellis for a linear block code $\mathcal{C}$ of length n and dimension k, can be constructed by specifying an arbitrary parity check

matrix $\mathcal{H}$ for $\mathcal{C}$ along with an $(n-k) \times k$ *displacement matrix* $\mathcal{D}$. We also present a dynamic algorithm that starts with a generator matrix for $\mathcal{C}$ in trellis-oriented form, and computes $\mathcal{D}$ for a minimal trellis in time $\mathcal{O}(n^k)$.

In Chapter 5, we generalize the Forney construction for conventional trellises to obtain tail-biting trellises. Specifically, we show that a linear tail-biting trellis for a linear block code can be computed from a certain coset decomposition of the code with respect to a subcode of the code.

In Chapter 6, we give a new construction for computing dual tail-biting trellises. We begin by describing Kötter and Vardy's *intersection product* to construct a linear tail-biting dual trellis directly from a generator matrix for the primal code. This results in a linear tail-biting trellis $T^{\perp}$ for the dual code that has the same state-complexity profile as the primal trellis T, only if T is $\prec_{\Theta}$-minimal. We next describe our construction of dual trellises that is based on the BCJR specification. This results in a linear tail-biting trellis for the dual code that has the same state-complexity profile as the primal trellis. Using this construction, we prove that given any minimal trellis T for a primal code, there exists a dual trellis $T^{\perp}$ for the dual code, such that T and $T^{\perp}$ have identical state-complexity profiles.

Finally, in Chapter 7, we describe an abstract characterization for linear tail-biting trellises in terms of an equivalence relation defined on a certain language derived from the code. In the context of formal language theory, this characterization is similar to the Myhill-Nerode theorem for regular languages.

We conclude our work in Chapter 8 with a summarization of our results. We also propose avenues of study for future research in this area.

Chapter 2

Preliminaries

In this chapter the basic notions of error-correcting codes will be reviewed, starting with the concept of linear block codes. We will then introduce the theory of trellises and constructions of trellises that will be used in this thesis.

2.1 An Introduction to Error-Correcting Codes

The purpose of this section is to provide the reader with a convenient reference for basic definitions and concepts from coding theory that we will use in this thesis. These definitions may be found in any textbook on coding theory [McE04, vL99].

The fundamental problem in communication is to deal with distortions that occur in messages that are sent from one place to another. Error-correcting codes provide a systematic means of *padding* a message carrying information with extra symbols, so that a receiver can retrieve the sent message even if part of the message is corrupted in transmission. In his seminal 1948 paper *A Mathematical Theory of Communication*, Shannon [Sha48] proved tight lower bounds on the amount of redundancy required to tolerate a prescribed amount of distortion introduced by discrete communication channels. Shannon modeled the communication problem as a situation in which a sender is trying to send information from a source to a destination over a channel that is *noisy*. He also described a coding scheme in which the original message is divided into blocks, and each block is padded with redundant information to form a *codeword* that is transmitted over the noisy channel. A receiver collects the possibly distorted transmitted word and if this is not too different from the sent codeword, it will

be able to pass on the correct message to the destination. This is illustrated in Figure 2.1. Remarkably, Shannon demonstrated that there exists a coding scheme that enables one to reliably communicate across a channel at rates lower than a quantity which he termed the *capacity* of the channel.

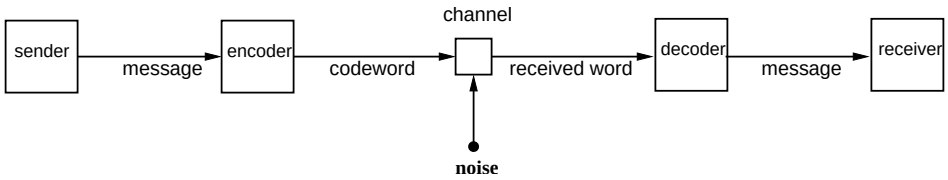


Figure 2.1: Shannon's schematic of a communication system.

The important components of a coding scheme are as follows.

- The information word $\boldsymbol{m}$ is a block of symbols that the sender wishes to transmit. For our purposes, $\boldsymbol{m}$ is an arbitrary word of k symbols from the finite field $\mathbb{F}_q$ of size q.
- The encoder is an algorithm that the sender uses to pad the information word $\boldsymbol{m}$ with redundant information. It can be looked upon as a function $enc : \mathbb{F}_q^k \rightarrow \mathbb{F}_q^n$ that transforms an information word of k symbols to a codeword of n symbols.
- The code $\mathcal{C}$ is the set of all possible codewords. That is, $\mathcal{C} = \{enc(\boldsymbol{m}) : \boldsymbol{m} \in \mathbb{F}_q^k\}$.
- Three important parameters of a code are

  (i) The block length of the code which is the length n of the codewords output by the encoder.

  (ii) The dimension k of the code which is the length of an information word. This term is typically used for *linear codes* when the code forms a vector space. These codes are described in Section 2.1.1.

  (iii) The rate R of a code is the ratio of its dimension and block length, or $R = k/n$.
- The channel is the communication medium over which the codeword is transmitted. There are many channel models in the literature, and the reader is referred to [CT91] for descriptions of some well known models.

- The received word $\boldsymbol{y}$ is the output of the channel and possibly corrupted version of the codeword $\boldsymbol{c}$.
- The decoder is an algorithm that the receiver uses to recover the original information word $\boldsymbol{m}$ from the received word $\boldsymbol{y}$. It can be looked upon as a function $dec : \mathbb{F}_q^n \rightarrow \mathbb{F}_q^k$ that computes an information word of k symbols from the received word $\boldsymbol{y}$ of n symbols.

The primary objective in coding theory is to design coding schemes with the best possible parameters and efficient encoding/decoding algorithms. It is often convenient to restrict codes to a certain structure and this is the topic of the next section.

2.1.1 Linear Block Codes

A code with block length n is any set of M vectors over a finite field $\mathbb{F}_q$. A code $\mathcal{C}$ with block length n over an alphabet $\mathbb{F}_q$ is a linear block code if it is a linear subspace of $\mathbb{F}_q^n$. If $\mathcal{C}$ has dimension k, we will call such a code an $(n,k)_q$ linear block code. There are two natural ways of specifying an $(n,k)_q$ code $\mathcal{C}$.

The first consists of specifying a basis for $\mathcal{C}$, in the form of a generator matrix. A generator matrix $\mathcal{G}$ for $\mathcal{C}$ is a $k \times n$ matrix whose rows are the basis vectors for the subspace $\mathcal{C}$. The encoder for $\mathcal{C}$ simply multiplies the information word $\boldsymbol{m}$ with the generator matrix $\mathcal{G}$. In other words, $enc(\boldsymbol{m}) = \boldsymbol{m}\mathcal{G}$, and therefore

$$\mathcal{C} = \{\boldsymbol{m}\mathcal{G} : \boldsymbol{m} \in \mathbb{F}_q^k\} \tag{2.1}$$

The second way of specifying $\mathcal{C}$ is by listing out the linear constraints that each codeword in $\mathcal{C}$ must obey. This is done by specifying a parity check matrix for $\mathcal{C}$. A parity check matrix $\mathcal{H}$ for $\mathcal{C}$ is an $(n-k) \times n$ matrix whose rows are the basis vectors for the dual code $\mathcal{C}^\perp$. The dual code $\mathcal{C}^\perp$ is the set of all words in $\mathbb{F}_q^n$ that are orthogonal to words in $\mathcal{C}$. Therefore $\mathcal{C}$ may also be defined as

$$\mathcal{C} = \{\boldsymbol{x} \in \mathbb{F}_q^n : \mathcal{H}\boldsymbol{x}^t = \boldsymbol{0}\} \tag{2.2}$$

Example 2.1 *As an example of a linear block code, consider a $(7,4)_2$ Hamming code $\mathcal{C}$, with block length $n = 7$, dimension $k = 4$ and rate $R = 4/7$. A generator matrix and parity*

check matrix for this code are

$$\mathcal{G} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad \mathcal{H} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

A matrix over a field $\mathbb{F}$ is said to be in row reduced echelon (RRE) form if it satisfies the following properties:

(a) The leftmost nonzero entry in each row is 1.

(b) Every column containing such a leftmost 1 has all its other entries 0.

(c) If the leftmost entry in row i occurs in column t_i, then $t_1 < t_2 < \cdots t_r$.

Definition 2.2 *An $(n,k)_q$ code is said to be a* systematic code *if every codeword has the information symbols in its first k components. In other words, its generator matrix can be written as $\mathcal{G} = [I_k \ A]$.*

Example 2.3 *The following is a generator matrix for a $(7,4)_2$ Hamming code in systematic form.*

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

Definition 2.4 *Let $\mathcal{C}$ be an $(n,k)_q$ code with parity check matrix $\mathcal{H} = [\boldsymbol{h}_1 \boldsymbol{h}_2 \cdots \boldsymbol{h}_n]$, where $\boldsymbol{h}_i \in \mathbb{F}_q^{(n-k)\times n}$, $1 \le i \le n$ are the columns of $\mathcal{H}$. Given a codeword $\boldsymbol{c} = (c_1, c_2, \ldots, c_n) \in \mathcal{C}$, the i^{th}* partial syndrome *of $\boldsymbol{c}$ is defined as $\sum_{j=1}^{i} c_j \boldsymbol{h}_j$.*

2.1.2 Coset Decomposition of a Linear Block Code

Let $\mathcal{C}$ be an $(n,k)_q$ code with generator matrix $\mathcal{G}$. A subset of q^{k_1} codewords $\mathcal{C}_0$ in $\mathcal{C}$, $0 \le k_1 \le k$, is called a linear subcode of $\mathcal{C}$ if this subset $\mathcal{C}_0$ is a k_1-dimensional subspace of

$\mathcal{C}$. Any k_1 rows of the generator matrix $\mathcal{G}$ span an $(n, k_1)_q$ linear subcode $\mathcal{C}$, and they form a generator matrix for the subcode.

Let $\mathcal{C}_0$ be an $(n, k_1)_q$ subcode of $\mathcal{C}$. Then $\mathcal{C}$ can be partitioned into q^{k-k_1} cosets $\{\mathcal{C}_i\}_{i=0}^{q^{k-k_1}-1}$ of $\mathcal{C}_0$, where

$$\mathcal{C}_i \stackrel{\text{def}}{=} \boldsymbol{v}_i + \mathcal{C}_0 = \{\boldsymbol{v}_i + \boldsymbol{y} : \boldsymbol{y} \in \mathcal{C}_0\},\ 1 \leq i \leq q^{k-k_1} - 1 \tag{2.3}$$

with $\boldsymbol{v}_i \in \mathcal{C} \backslash \mathcal{C}_0$. This partition of $\mathcal{C}$ with respect to $\mathcal{C}_0$ is denoted by $\mathcal{C}/\mathcal{C}_0$, and the codewords $\boldsymbol{v}_i,\ 1 \leq i \leq q^{k-k_1} - 1$, are called the coset representatives. Any codeword in a coset can be used as the coset representative without changing the coset.

Example 2.5 *For the $(7, 4)_2$ Hamming code from Example 2.3, choosing $k_1 = 2$, the subcode $\mathcal{C}_0$ as the space spanned by the first two rows of $\mathcal{G}$, $\boldsymbol{v}_1 = 0010111$, $\boldsymbol{v}_2 = 0001101$ and $\boldsymbol{v}_3 = 0011010$, we obtain the following coset decomposition $\mathcal{C}/\mathcal{C}_0$.*

$$\begin{aligned}
\mathcal{C}_0 &= \{0000000, 1000110, 0100011, 1100101\} \\
\mathcal{C}_1 &= \{0010111, 1010001, 0110100, 1110010\} \\
\mathcal{C}_2 &= \{0001101, 1001011, 0101110, 1101000\} \\
\mathcal{C}_3 &= \{0011010, 1011101, 0111001, 1111111\}
\end{aligned}$$

2.1.3 Decoding an Error-Correcting Code

Suppose an $(n, k)_2$ linear block code $\mathcal{C}$ is used for error-correction over an Additive White Gaussian Noise (AWGN) channel [CT91].

Definition 2.6 *The Additive While Gaussian Noise (AWGN) channel has real input $x \in \mathbb{R}$ and real output $y \in \mathbb{R}$. The conditional distribution of y given x is a Gaussian distribution.*

$$\Pr(y|x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(y-x)^2}{2\sigma^2}\right] \tag{2.4}$$

This channel has inputs and outputs that are from a continuous space, but is discrete in time.

Let $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ be the transmitted codeword. Before transmission, a modulator maps each codeword component into an elementary signal waveform. The resulting signal sequence is then transmitted over the channel and is possibly distorted by noise. At the

receiver, the received signal sequence is processed by a demodulator and this results in a *received sequence* of real numbers $\boldsymbol{r} = (r_1, r_2, \ldots, r_n)$. Each component of $\boldsymbol{r}$ is the sum of a fixed real number c and a Gaussian random variable of zero mean and variance σ^2, where c corresponds to a transmitted codeword bit. In one scheme, the demodulator can be used to make *hard-decisions* on whether each transmitted codeword bit is a '0' or a '1'. This results in a binary *received sequence* $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)$, which may contain transmission errors, that is, for some i, $y_i \neq x_i$. This sequence $\boldsymbol{y}$ is then fed to the decoder which attempts recover the original transmitted codeword $\boldsymbol{x}$. Since the decoder operates on hard-decisions made by the demodulator, the decoding process is called hard-decision decoding.

Another scheme uses the unquantized outputs from the demodulator and feeds them directly to the decoder, and this decoding process is called soft-decision decoding. Since the decoder makes use of the additional information contained in the unquantized received samples to recover the transmitted codeword, soft-decision decoding provides better error-performance than hard-decision decoding. However, hard-decision decoding algorithms are easier to implement. Various hard-decision decoding algorithms based on the algebraic structure of linear block codes have been devised [McE04]. More recently, effective soft-decision decoding algorithms have been devised and they achieve either optimum error performance or suboptimum error performance with reduced decoding complexity [Wib96].

Let $\hat{\boldsymbol{x}}$ be the estimate of the transmitted codeword $\boldsymbol{x}$ output by the decoder. A decoding error occurs if and only if $\hat{\boldsymbol{x}} \neq \boldsymbol{x}$. Given that $\boldsymbol{r}$ is received, the conditional error probability of the decoder is defined as

$$\Pr(E|\boldsymbol{r}) \stackrel{\text{def}}{=} \Pr(\hat{\boldsymbol{x}} \neq \boldsymbol{x}|\boldsymbol{r}) \tag{2.5}$$

The error probability of the decoder is then given by

$$\Pr(E) = \sum_{\boldsymbol{r}} \Pr(E|\boldsymbol{r}) \Pr(\boldsymbol{r}) \tag{2.6}$$

An optimum decoding rule is one that minimizes $\Pr(E|\boldsymbol{r})$ for all $\boldsymbol{r}$. This translates to choosing a codeword $\hat{\boldsymbol{x}}$ that maximizes

$$\Pr(\hat{\boldsymbol{x}}|\boldsymbol{r}) = \frac{\Pr(\boldsymbol{r}|\hat{\boldsymbol{x}}) \Pr(\hat{\boldsymbol{x}})}{\Pr(\hat{\boldsymbol{x}})} \tag{2.7}$$

that is, $\hat{\boldsymbol{x}}$ is chosen as the most likely codeword conditional on the received sequence $\boldsymbol{r}$. If all codewords are equally likely, maximizing (2.7) is equivalent to maximizing $\Pr(\boldsymbol{r}|\boldsymbol{x})$. For a memoryless channel[1] we have,

$$\Pr(\boldsymbol{r}|\boldsymbol{x}) = \prod_{i=1}^{n} \Pr(r_i|x_i) \tag{2.8}$$

A decoder that chooses its estimate so as to maximize (2.8) is called a maximum-likelihood decoder, and the decoding process is called Maximum-Likelihood (ML) decoding.

The problem of maximum-likelihood decoding (hard-decision and soft-decision) in coding theory is known to be an NP-hard problem [BMvT78]. Nevertheless, it is possible to achieve maximum-likelihood soft-decision decoding of linear codes with a complexity exponent that is much smaller than $n \min\{R, 1-R\}$ (where n and R are the length and rate of the code under consideration) [LKFF98]. Despite their exponential complexity, these algorithms are of significant interest in coding theory due to the significant gap that exists between the performance of hard-decision bounded distance decoding and soft-decision maximum-likelihood decoding. Trellis decoding is the most pragmatic and well researched method of performing soft-decision decoding of this nature [LKFF98]. Therefore, the construction of minimal trellises is of interest as the decoding algorithm reduces to a shortest path algorithm (the Viterbi algorithm [For73, Var98]) on the trellis.

2.2 The Trellis Structure of Linear Block Codes

The trellis was first introduced by Forney [For67] in 1967 as a conceptual device to explain the Viterbi algorithm for decoding convolutional codes. While the trellis has been studied in the context of convolutional codes for the first two decades since its inception, recent times have seen a significant amount of research devoted to the study of the trellis structure of linear block codes [Var98]. In 1974, Bahl, Cocke, Jelinek and Raviv [BCJR74] worked on various aspects of convolutional codes, and showed that linear block codes have combinatorial descriptions in the form of trellises, thus uncovering an important connection between block codes and convolutional codes. We will first define conventional trellises (and introduce

[1] A channel is memoryless if the output r_i at time i depends only on the input x_i at time i.

tail-biting trellises later), and then describe several constructions for minimal conventional trellises. An excellent survey of the theory of conventional trellises for linear block codes is [Var98].

Definition 2.7 *A* conventional trellis $T = (V, E, \Sigma)$ *of depth* n *is an edge-labeled directed graph with the property that the set* V *can be partitioned into* $n+1$ vertex classes

$$V = V_0 \cup V_1 \cup \cdots \cup V_n \tag{2.9}$$

where $|V_0| = |V_n| = 1$*, such that every edge in* T *is labeled with a symbol from the alphabet* Σ*, and begins at a vertex of* V_i *and ends at a vertex of* V_{i+1}*, for some* $i \in \{0, 1, \ldots, n-1\}$*.*

A conventional trellis is just the transition diagram for a finite automaton defined below.

Definition 2.8 *A* nondeterministic finite automaton *(*NFA*)* M *is a 5-tuple* $(Q, \Sigma, \delta, q_0.F)$*, where* Q *is a finite set of* states*,* Σ *is a finite* input alphabet*,* $q_0 \in Q$ *is the* initial state*,* $F \subseteq Q$ *is the set of* final states*, and* $\delta : Q \times \Sigma \to 2^Q$ *is the* transition function*. That is,* $\delta(q, a)$ *is a set of states for each state* $q \in Q$ *and input symbol* $a \in \Sigma$*. If for all* $q \in Q,\ a \in \Sigma$*,* $\delta(q, a)$ *is a unique state* $q' \in Q$*, then the finite automaton is said to be a* deterministic finite automaton *(*DFA*).*

Formal definitions related to automata theory are given in Chapter 7. It is easily seen that a trellis T can be represented by a finite automaton M (either deterministic or nondeterministic), where each vertex of T is a state of M.

The length of a path (in edges) from the root to any vertex is unique and the set of indices $\mathcal{I} = \{0, 1, \ldots, n\}$ for the partition in (2.9) are the time indices. Therefore, $\log_{|\Sigma|} |V_i|$ is the state-complexity of the trellis at time index i and the sequence $\left\{\log_{|\Sigma|} |V_i|,\ 0 \le i \le n\right\}$ defines the state-complexity profile (SCP) of the trellis. We will denote by $s_{\max}(T)$ the maximum state-complexity of T over all time indices. The trellis T is said to represent a block code $\mathcal{C}$ over Σ if the set of all edge-label sequences in T is equal to $\mathcal{C}$. Let $\mathcal{C}(T)$ denote the code represented by the trellis T.

Definition 2.9 ([HU77]) *A finite automaton* M *accepting a set* L *is said to be minimal if the total number of states in* M *is minimized.*

There can be an exponential gap between the size of the minimal DFA and the size of the minimal NFA recognizing a language. That is, there are finite state languages for which the minimal DFA is exponentially larger than the corresponding minimal NFA. An example of such a language is $L = \Sigma^* 0 \Sigma^n$, where $\Sigma = \{0, 1\}$ and n is a fixed positive integer (definitions related to formal language theory may be found in Chapter 7).

However, this is not true for trellises for linear block codes. The minimal trellis (Definition 2.10) for a linear block code is the minimal deterministic automaton. The measure of trellis complexity commonly used by coding theorists has to do with the SCP. The following definition of trellis minimality is due to Muder [Mud88].

Definition 2.10 *A trellis T for a code $\mathcal{C}$ of length n is* minimal *if it satisfies the following property: for each $i = 0, 1, \ldots, n$, the number of vertices in T at time i is less than or equal to the number of vertices at time i in any other trellis for $\mathcal{C}$.*

Note that this is a stronger definition of minimality than that for finite automata.

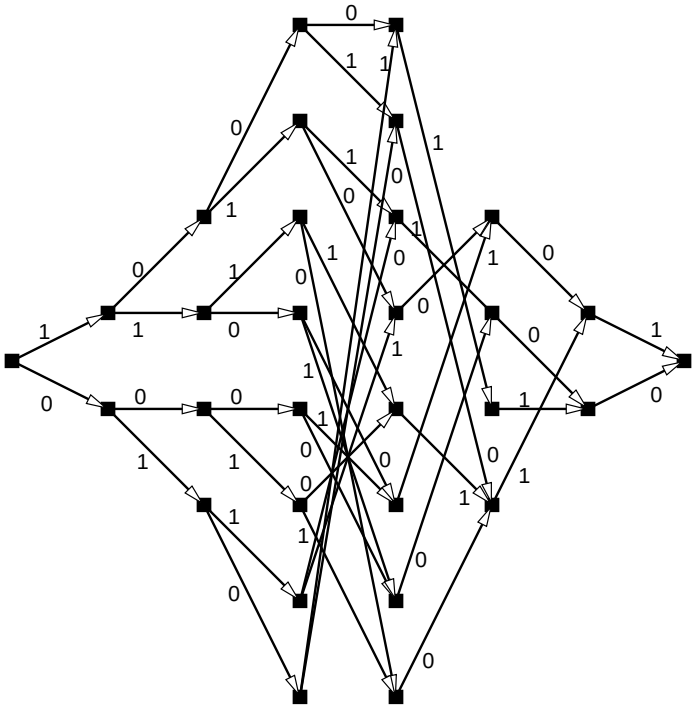


Figure 2.2: The minimal conventional trellis for the $(7, 4)_2$ Hamming code.

Other measures are the maximum number of states at any time index, total number of states, the total number of edges, the maximum number of edges at any time index and the

product of the state cardinalities over all time indices.

Theorem 2.11 ([HU77]) *The minimal state deterministic automaton accepting a set L is unique up to an isomorphism (that is, a renaming of the states).*

Theorem 2.12 ([McE96, Mud88]) *Minimal trellises for linear block codes are unique, and simultaneously satisfy all definitions of minimality.*

While the first part of the theorem is a direct consequence of linearity and Theorem 2.11, the second part is not obvious.

Definition 2.13 *A trellis is said to be* biproper *if any pair of edges directed towards a vertex has distinct labels (co-proper), and so also any pair of edges leaving a vertex (proper).*

All biproper conventional trellises are minimal and vice versa [Var98]. Figure 2.2 illustrates the minimal conventional trellis representing the $(7,4)_2$ Hamming code defined in Example 2.1. However, all biproper conventional trellises are not necessarily linear [KS95].

We will now briefly survey some well-known constructions of minimal conventional trellises for linear block codes. Although the constructions are different, the fact that the minimal trellis is unique implies that they all produce the same trellis up to an isomorphism. We will describe the constructions due to Bahl, Cocke, Jelinek and Raviv [BCJR74], Massey [Mas78], Forney [For88], and Kschischang and Sorokine [KS95].

2.2.1 The Bahl,Cocke, Jelinek, Raviv Construction

Let $\mathcal{H}$ be an $(n-k)\times n$ parity check matrix for an (n,k) linear block code $\mathcal{C}$ over $\mathbb{F}_q$, and let $\mathcal{H} = [\boldsymbol{h}_1\boldsymbol{h}_2\cdots\boldsymbol{h}_n]$, $\boldsymbol{h}_i \in \mathbb{F}_q^{(n-k)\times n}$, $1 \le i \le n$ be the columns of $\mathcal{H}$. Every codeword $\boldsymbol{c} = (c_1, c_2, \ldots, c_n) \in \mathcal{C}$ induces a sequence of states $\{\boldsymbol{s}_i\}_{i=0}^{n}$, each state being labeled by a vector in $\mathbb{F}_q^{(n-k)\times 1}$ as follows.

$$\boldsymbol{s}_i = \begin{cases} \mathbf{0} & \text{if } i = 0 \\ \boldsymbol{s}_{i-1} + c_i\boldsymbol{h}_i & \text{otherwise.} \end{cases} \tag{2.10}$$

Clearly, there will be a single state at time index n as $\mathcal{H}\boldsymbol{c}^t = \mathbf{0}$, for all codewords $\boldsymbol{c} \in \mathcal{C}$.

There is an edge labeled $a \in \mathbb{F}_q$ from state $\boldsymbol{s}_{i-1}$ to state $\boldsymbol{s}_i,\ 1 \leq i \leq n-1$, if and only if

$$\boldsymbol{s}_i = \boldsymbol{s}_{i-1} + a\boldsymbol{h}_i \tag{2.11}$$

We refer to such a labeling as a BCJR labeling of the trellis, and will refer to the labeled sequences in the BCJR label code as the BCJR labeled words.

Lemma 2.14 (State-Space Lemma [McE96]) *The set of vectors that are labels at each time index form a vector space whose dimension is the state-complexity at that time index.*

Lemma 2.15 *Let $\mathcal{H}_i$ (resp. $\mathcal{G}_i$) denote the matrix consisting of the first i columns of $\mathcal{H}$ (resp. $\mathcal{G}$), where $\mathcal{H}$ (resp. $\mathcal{G}$) is the parity check (resp. generator) matrix for an $(n,k)_q$ code $\mathcal{C}$. Then the* BCJR *trellis $T = (V, E, \mathbb{F}_q)$ has vertex space V_i at time index i defined as follows.*

$$V_i = \text{column}-\text{space}(\mathcal{H}_i \mathcal{G}_i^t) \tag{2.12}$$

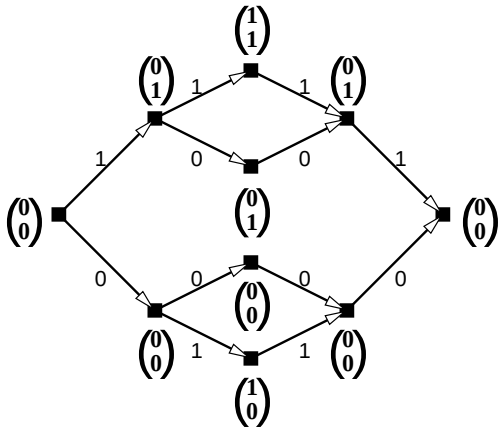


Figure 2.3: The minimal conventional BCJR trellis for the $(4,2)_2$ code.

Example 2.16 *Consider a self dual $(4,2)_2$ code with parity check matrix $\mathcal{H}$ defined as*

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

The minimal conventional trellis resulting from the BCJR *construction for this code is shown in Figure 2.3.*

There are a number of interesting connections between the minimal trellis for a linear code $\mathcal{C}$ and the minimal trellis for its dual code $\mathcal{C}^\perp$. The following theorem was first established by Forney [For88], and the proof follows easily from the properties of the BCJR construction.

Theorem 2.17 *The minimal trellis $T = (V, E, \mathbb{F}_q)$ for a linear code $\mathcal{C}$ of length n and the minimal trellis $T^\perp = \left(V^\perp, E^\perp, \mathbb{F}_q\right)$ for its dual code $\mathcal{C}^\perp$ have identical state-complexity profiles.*

Proof: Let $\mathcal{H}$ and $\mathcal{G}$ respectively, be the parity check and generator matrices for $\mathcal{C}$. From Lemma 2.15, we have $V_i = \mathcal{H}_i \mathcal{G}_i^t$ and $V_i^\perp = \mathcal{G}_i \mathcal{H}_i^t$. Therefore $V_i = {V_i^\perp}^t$, and the theorem follows. ■

The code in Example 2.16 is self dual and therefore the trellis for the dual code is identical to the trellis for the primal code.

2.2.2 The Massey Construction

In contrast to the BCJR construction which uses codeword symbols that have appeared at *past* time indices, the Massey construction uses parity check symbols that have yet to be observed after the current time index to label states. To achieve this the construction requires that the generator matrix for the code be in systematic form defined earlier, and this is the starting point for the Massey algorithm.

Given a word $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{F}_q^n$, let $\triangleright(\boldsymbol{x})$ denote the smallest integer i such that $x_i \neq 0$. Since $\mathcal{G}$ is in RRE form

$$\triangleright(\boldsymbol{g}_1) < \triangleright(\boldsymbol{g}_2) < \cdots < \triangleright(\boldsymbol{g}_k)$$

where $\{\boldsymbol{g}_i\}_{i=1}^k$ are the rows of $\mathcal{G}$. The columns of $\mathcal{G}$ corresponding to positions $\{\triangleright(\boldsymbol{g}_i)\}_{i=1}^k$ are all of Hamming weight equal to one.

The Massey trellis $T = (V, E, \mathbb{F}_q)$ for an $(n, k)_q$ linear block code $\mathcal{C}$ with generator matrix $\mathcal{G}$ (in RRE form) is computed by associating the vertices V_i at time index i with parity symbols that are determined by information symbols that have already been observed at time i, with the remaining information symbols being treated as zeros. Let j be the largest

integer such that $\triangleright(\boldsymbol{g}_j) \leq i$. Then

$$V_i = \{(c_{i+1}, \ldots, c_n) : (c_1, \ldots, c_n) = (u_1, \ldots, u_j, 0, \ldots, 0)\mathcal{G}\} \tag{2.13}$$

where $(u_1, \ldots, u_j) \in \mathbb{F}_q^j$. By convention, we have $V_0 = \{0\}$ and $V_n = \{\epsilon\}$, where ϵ is the empty string.

The set of edges E in T is defined as follows. If $i > \triangleright(\boldsymbol{g}_j)$, there is an edge $e \in E_i$ labeled c_i from a vertex $\boldsymbol{v} \in V_{i-1}$ to a vertex $\boldsymbol{v}' \in V_i$ if and only if there exists a codeword $\boldsymbol{c} = (c_1, c_2, \ldots, c_n)$ such that

$$\begin{aligned} (c_i, c_{i+1} \ldots, c_n) &= \boldsymbol{v}, \\ (c_{i+1}, \ldots, c_n) &= \boldsymbol{v}'. \end{aligned}$$

On the other hand if $i = \triangleright(\boldsymbol{g}_j)$, there is an edge $e \in E_i$ labeled c_i' from a vertex $\boldsymbol{v} \in V_{i-1}$ to a vertex $\boldsymbol{v}' \in V_i$, if and only if there exists a pair of codewords $c = (c_1, c_2, \ldots, c_n)$ and $c' = (c_1', c_2', \ldots, c_n')$ such that

$$\begin{aligned} (c_i, c_{i+1} \ldots, c_n) &= \boldsymbol{v}, \\ \left(c_{i+1}', \ldots, c_n'\right) &= \boldsymbol{v}', \end{aligned}$$

and either $\boldsymbol{c} = \boldsymbol{c}'$ or $\beta(\boldsymbol{c}' - \boldsymbol{c})$ equals $\boldsymbol{g}_j$ for some $\beta \in \mathbb{F}_q$. The resulting trellis is isomorphic to the minimal conventional trellis [Var98].

Example 2.18 *Consider again the self dual* $(4, 2)_2$ *code with an* RRE *form generator matrix* $\mathcal{G}$ *defined as*

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

The minimal conventional trellis resulting from the Massey *construction for this code is the same as the trellis in Figure 2.3, with transposed vertex labels.*

2.2.3 The Forney Construction

Forney [For88] has an elegant algebraic characterization of conventional trellises in terms of a coset decomposition of the linear block code. He has shown that when the code $\mathcal{C}$ is a group

code, there is a natural definition of the state spaces of $\mathcal{C}$ as quotient groups such that $\mathcal{C}$ has a minimal realization with these state spaces. Since the state spaces are unique there is a unique quotient group defined by $\mathcal{C}$. We briefly review his characterization here. Let $\mathcal{C}$ be an $(n,k)_q$ linear code with generator matrix $\mathcal{G}$ and parity check matrix $\mathcal{H} = [\boldsymbol{h}_1 \boldsymbol{h}_2 \cdots \boldsymbol{h}_n]$. Let $\pi_i : \mathcal{C} \rightarrow \pi_i(\mathcal{C})$ be a map defined by $\boldsymbol{c} = (c_1, \ldots, c_n) \mapsto c_1\boldsymbol{h}_1 + \cdots + c_i\boldsymbol{h}_i$. The map π_i thus effectively maps a codeword to its i^{th} partial syndrome. Define the past projection

$$\mathcal{P}_i = \{(c_1, \ldots, c_i) : \boldsymbol{c} = (c_1, \ldots, c_i, c_{i+1}, \ldots, c_n) \in \mathcal{C},\ \pi_i(\boldsymbol{c}) = \boldsymbol{0}\} \tag{2.14}$$

These are projections of codewords that share an all-zero suffix with the all-zero codeword, from index $i+1$ to index n. Define the future projection

$$\mathcal{F}_i = \{(c_{i+1}, \ldots, c_n) : \boldsymbol{c} = (c_1, \ldots, c_i, c_{i+1}, \ldots, c_n) \in \mathcal{C},\ \pi_i(\boldsymbol{c}) = \boldsymbol{0}\} \tag{2.15}$$

These are projections of codewords that share an all zero-prefix with the all-zero codeword, from index 1 to index i.

Definition 2.19 *Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be projections of $\mathbb{F}_q^n$. Then the* product *of $\mathcal{C}_1$ and $\mathcal{C}_2$, denoted by $\mathcal{C}_1 \times \mathcal{C}_2$, is defined as*

$$\mathcal{C}_1 \times \mathcal{C}_2 \stackrel{\text{def}}{=} \{(\boldsymbol{c}_1, \boldsymbol{c}_2) : \boldsymbol{c}_1 \in \mathcal{C}_1,\ \boldsymbol{c}_2 \in \mathcal{C}_2\} \tag{2.16}$$

It is easy to see that the product $(\mathcal{P}_i \times \mathcal{F}_i)$ is a linear subcode of $\mathcal{C}$. Therefore $\mathcal{C}/(\mathcal{P}_i \times \mathcal{F}_i)$ forms a quotient space. The Forney trellis $T = (V, E, \mathbb{F}_q)$ for $\mathcal{C}$ is constructed by identifying vertices in V_i with the quotient group corresponding to cosets of $\mathcal{C}$ modulo $(\mathcal{P}_i \times \mathcal{F}_i)$, that is,

$$V_i \stackrel{\text{def}}{=} \mathcal{C}/(\mathcal{P}_i \times \mathcal{F}_i) \text{ for } i \in \{1, \ldots, n\}\,. \tag{2.17}$$

There is an edge from $e \in E_i$ labeled c_i from a vertex $\boldsymbol{u} \in V_{i-1}$ to a vertex $\boldsymbol{v} \in V_i$, if and only if there exists a codeword $\boldsymbol{c} = (c_1, \ldots, c_n) \in \mathcal{C}$ such that $\boldsymbol{c} \in \boldsymbol{u} \cap \boldsymbol{v}$. The resulting trellis is isomorphic to the minimal conventional trellis [Var98].

Example 2.20 *Consider again the self dual $(4,2)_2$ code $\mathcal{C}$ from Example 2.16 and Example 2.18. The past projections are $\mathcal{P}_0 = \mathcal{P}_1 = \{0\}$, $\mathcal{P}_2 = \{00\}$, $\mathcal{P}_3 = \{000, 011\}$, $\mathcal{P}_4 = \mathcal{C}$. The future projections are $\mathcal{F}_0 = \mathcal{C}$, $\mathcal{F}_1 = \{000, 110\}$, $\mathcal{F}_2 = \{00\}$, $\mathcal{F}_3 = \mathcal{F}_4 = \{0\}$. The subcodes*

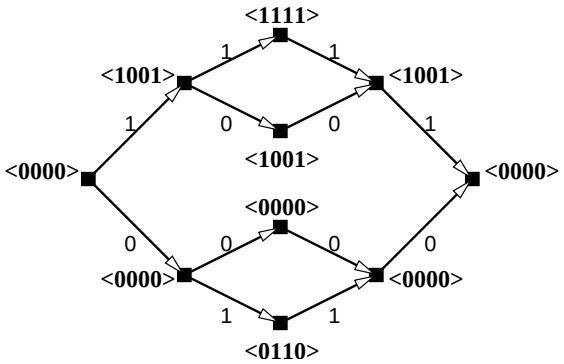


Figure 2.4: The minimal conventional Forney trellis for the $(4, 2)_2$ code.

$\mathcal{P}_i \times \mathcal{F}_i$ *are thus given by*

$$\begin{aligned} \mathcal{P}_1 \times \mathcal{F}_1 &= \{0000, 0110\} \\ \mathcal{P}_2 \times \mathcal{F}_2 &= \{0000\} \\ \mathcal{P}_3 \times \mathcal{F}_3 &= \{0000, 0110\} \end{aligned}$$

The minimal conventional trellis for $\mathcal{C}$ *resulting from the* Forney *construction is shown in Figure 2.4. The vertices in Figure 2.4 are labeled by the coset representatives in* $\mathcal{C}/(\mathcal{P} \times \mathcal{F})$*. For example, the vertex* $\{1001, 1111\} \in V_1$ *is labeled by* $\langle 1001 \rangle$*.*

This determines the quotient space $\mathcal{C}/(\mathcal{P}_i \times \mathcal{F}_i)$ at all time indices. Therefore we have

$$\begin{aligned} V_1 &= \{\{0000, 0110\}, \{1001, 1111\}\} \\ V_2 &= \{\{0000\}, \{0110\}, \{1001\}, \{1111\}\} \\ V_3 &= \{\{0000, 0110\}, \{1001, 1111\}\} \end{aligned}$$

2.2.4 The Kschischang-Sorokine Construction

The main idea in this construction is to represent $\mathcal{C}$ as a sum of certain *elementary* subcodes and then construct a trellis for $\mathcal{C}$ as a product of trellises for these subcodes. This construction differs from the previous three in that it specifies a number of trellises for $\mathcal{C}$, only one of which is minimal.

Define a trellis product operator as follows. This operator is a binary operator on trellises

T_1, T_2 of the same depth, and computes a *product trellis* $T = T_1 \times T_2$ such that

$$\mathcal{C}(T) = \mathcal{C}(T_1) + \mathcal{C}(T_2) \stackrel{\text{def}}{=} \{\boldsymbol{c}_1 + \boldsymbol{c}_2 : \boldsymbol{c}_1 \in \mathcal{C}(T_1),\ \boldsymbol{c}_2 \in \mathcal{C}(T_2)\} \tag{2.18}$$

Let $T_1 = (V', E', \mathbb{F}_q)$ and $T_2 = (V'', E'', \mathbb{F}_q)$ be two trellises on depth n. Then the set of vertices at time index i in $T = (V, E, \mathbb{F}_q)$ is the Cartesian product

$$V_i = V_i' \times V_i'' \stackrel{\text{def}}{=} \{(v', v'') : v' \in V_i',\ v'' \in V_i''\} \tag{2.19}$$

There is an edge $e \in E_i$ in T labeled $a \in \mathbb{F}_q$, from a vertex $(v'_{i-1}, v''_{i-1}) \in V_{i-1}$ to a vertex $(v'_i, v''_i) \in V_i$, if and only if $(v'_{i-1}, a', v'_i) \in E'_i$, $(v''_{i-1}, a'', v''_i) \in E''_i$ and $a = a' + a''$. However, the product trellis T is not necessarily the minimal trellis for $\mathcal{C}(T)$, even if the trellises T_1 and T_2 are minimal trellises for $\mathcal{C}(T_1)$ and $\mathcal{C}(T_2)$ respectively. The trellis product operator is both associative and commutative.

Let $\mathcal{C}$ be an $(n, k)_q$ code and let $\mathcal{G}$ be its generator matrix. Each row $\boldsymbol{g}_i,\ 1 \le i \le k$, in $\mathcal{G}$ generates a one-dimensional subcode of $\mathcal{C}$ which we will denote by $\langle \boldsymbol{g}_i \rangle$. Therefore

$$\mathcal{C} = \langle \boldsymbol{g}_1 \rangle + \langle \boldsymbol{g}_2 \rangle + \cdots + \langle \boldsymbol{g}_k \rangle \tag{2.20}$$

Therefore, if $T_1, T_2, \ldots, T_k$ are trellises for $\langle \boldsymbol{g}_1 \rangle, \langle \boldsymbol{g}_2 \rangle, \ldots, \langle \boldsymbol{g}_k \rangle$ respectively, then their product represents $\mathcal{C}$. Denote by $T_{\boldsymbol{g}_i}$ the minimal trellis for $\langle \boldsymbol{g}_i \rangle$ and define

$$T_{\mathcal{G}} \stackrel{\text{def}}{=} \prod_{i=1}^{k} T_{\boldsymbol{g}_i} \tag{2.21}$$

The trellis $T_{\mathcal{G}}$ is called the Kschischang-Sorokine (KS) trellis for $\mathcal{C}$. Given a codeword $\boldsymbol{c} \in \mathcal{C}$, the structure of $T_{\boldsymbol{c}}$ depends critically on the notion of a span. The span of $\boldsymbol{c}$, denoted by $[\boldsymbol{c}]$, is the nonempty interval $[i, j],\ i \le j$ that contains all the nonzero positions of $\boldsymbol{c}$. The span of $\mathbf{0} \in \mathcal{C}$ is defined to be the empty interval $[\ ]$. The minimal trellis $T_{\boldsymbol{c}}$ for a codeword $\boldsymbol{c}$ with span $[i, j]$ is called an elementary trellis. The elementary trellis $T_{\boldsymbol{c}}$ has q vertices at times $i, i+1, \ldots, j-1$, corresponding to the q different multiples of the word $\boldsymbol{c}$, with each vertex having degree equal to two.

Definition 2.21 *A generator matrix $\mathcal{G}$ is said to be in* minimal-span form *(also called a*

trellis-oriented form (TOF)) *if and only if it does not contain rows that have spans that start at the same position or end at the same position.*

Given two generator matrices $\mathcal{G}_1$ and $\mathcal{G}_2$ in minimal-span form, it is not necessary that the rows of $\mathcal{G}_1$ are a permutation of the rows of $\mathcal{G}_2$. On the other hand, the set of spans in $\mathcal{G}_1$ is equal to the set of spans in $\mathcal{G}_2$. That is, the set of spans for a matrix in minimal-span form are uniquely determined by the code [KS95, McE96].

Lemma 2.22 ([KS95]) *Any generator matrix $\mathcal{G} \in \mathbb{F}_q^{k \times n}$ can be transformed to a minimal-span form in time $\mathcal{O}(k^2)$.*

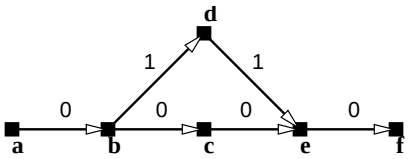


Figure 2.5: The elementary trellis for (0110) with span $[2, 3]$.

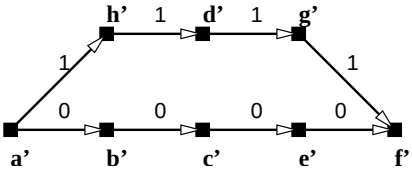


Figure 2.6: The elementary trellis for (1001) with span $[1, 4]$.

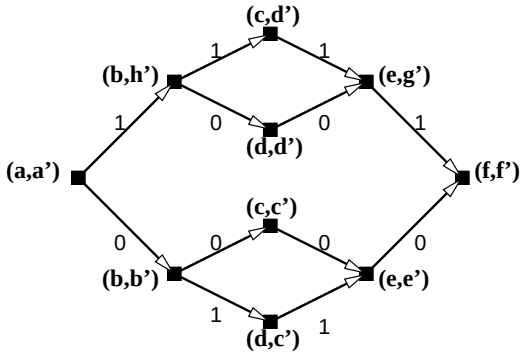


Figure 2.7: The minimal Kschischang-Sorokine product trellis for the $(4, 2)_2$ code.

Kschischang and Sorokine prove the following result concerning the minimal product trellis [KS95].

Theorem 2.23 *The* KS *trellis $T_{\mathcal{G}}$ is a minimal trellis if and only if $\mathcal{G}$ is in minimal-span form.*

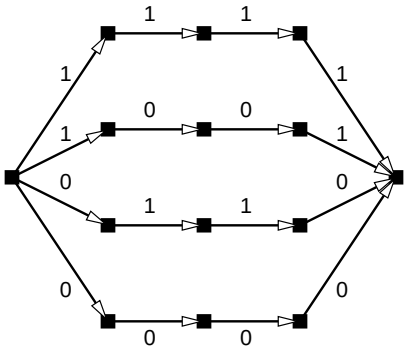


Figure 2.8: A non-biproper Kschischang-Sorokine product trellis for the $(4,2)_2$ code.

Example 2.24 *We will return to the $(4,2)_2$ code $\mathcal{C}$ with generator matrix $\mathcal{G}$ in minimal span form. The spans are adjacent to the rows of $\mathcal{G}$.*

$$\mathcal{G} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \begin{matrix} [2,3] \\ [1,4] \end{matrix}$$

The elementary trellises for the generators in $\mathcal{G}$ are shown in Figures 2.5 and 2.6, and the minimal KS *product trellis for $\mathcal{C}$ is shown in Figure 2.7. Consider a generator matrix that is not in minimal-span form as given below.*

$$\mathcal{G} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix} \begin{matrix} [1,4] \\ [1,4] \end{matrix}$$

The product trellis for $\mathcal{C}$ is shown in Figure 2.8, and this trellis is not biproper. Note that this trellis cannot be computed by the BCJR, Massey *and* Forney *constructions.*

2.2.5 Upper Bounds on Trellis Complexity

For an $(n,k)_q$ code $\mathcal{C}$, the state-complexity of the minimal trellis $T = (V, E, \mathbb{F}_q)$ representing $\mathcal{C}$ is measured by its SCP $\left(\log_q |V_0|, \log_q |V_1|, \ldots, \log_q |V_n|\right)$. Define the state-complexity of $\mathcal{C}$, $s_{\max}(\mathcal{C})$ as follows.

$$s_{\max}(\mathcal{C}) \stackrel{\text{def}}{=} s_{\max}(T) = \max_{0 \le i \le n} \log_q |V_i| \tag{2.22}$$

The following upper bound on the state-complexity of linear codes was first observed by Wolf [Wol78].

Theorem 2.25 (Wolf bound) *Let $\mathcal{C}$ be an $(n,k)_q$ code. Then the state-complexity of $\mathcal{C}$ is upper bounded by*

$$s_{\max}(\mathcal{C}) \leq \min\{k, n-k\} \tag{2.23}$$

In general, the above bound is quite loose. However for cyclic codes [KTFL93] and MDS codes [Mud88, For94], this bound is tight. If the Viterbi algorithm [For73, Var98] (a shortest path decoding algorithm on a trellis) is applied to the minimal trellis of a code $\mathcal{C}$, then the amount of space required is $\mathcal{O}(q^{s_{\max}(\mathcal{C})})$. Therefore, the parameter $s_{\max}(\mathcal{C})$ is a key measure of trellis/decoding complexity.

Another variant of the Wolf bound extends it to an upper bound on the entire SCP of the minimal trellis for a code.

Theorem 2.26 *Let $\mathcal{C}$ be an $(n,k)_q$ code, and let $T = (V, E, \mathbb{F}_q)$ be the minimal trellis for $\mathcal{C}$. Then $\forall i \in \{0, 1, \ldots, n\}$*

$$\log_q |V_i| \leq \min\{i, k, n-k, n-i\} \tag{2.24}$$

2.3 Tail-Biting Trellises for Linear Block Codes

Tail-biting trellises were originally introduced by Solomon and van Tilborg [SvT79] in 1979. The development of the theory of tail-biting trellises for linear block codes started with the work of Calderbank, Forney and Vardy [CFV99], in which efficient constructions of tail-biting trellises for several short codes were computed. Several advances in the understanding of the structure and properties of such trellises have been made in recent years [KV02, KV03, RB99, SB00, SKSR01] primarily due to the growing interest in the subject of *codes on graphs* [For01, Wib96]. Tail-biting trellises for linear block codes are combinatorial descriptions that are somewhat more compact than the corresponding conventional trellis descriptions. Though conventional trellises for block codes have a well understood underlying theory, the theory of tail-biting trellises appears to be somewhat more involved.

Definition 2.27 *A* tail-biting trellis $T = (V, E, \Sigma)$ *of depth* n *is an edge-labeled directed graph with the property that the set* V *can be partitioned into* n *vertex classes*

$$V = V_0 \cup V_1 \cup \cdots \cup V_{n-1} \tag{2.25}$$

such that every edge in T *is labeled with a symbol from the alphabet* Σ*, and begins at a vertex of* V_i *and ends at a vertex of* V_{i+1}*, for some* $i \in \{0, 1, \ldots, n-1\}$*.*

Tail-biting trellises may be viewed as graphs obtained by splitting the finite automaton corresponding to the conventional trellis for the code into identically structured sub-automata and "overlaying" them. The overlayed structure of tail-biting trellises is discussed in Section 3.4.

As in the case with conventional trellises, the set of indices $\mathcal{I} = \{0, 1, \ldots, n-1\}$ for the partition (2.25) are the time indices. We identify $\mathcal{I}$ with $\mathbb{Z}_n$, the residue classes of integers modulo n. An interval of indices $[i, j]$ represents the sequence $\{i, i+1, \ldots j\}$ if $i < j$, and the sequence $\{i, i+1, \ldots n-1, 0, \ldots j\}$ if $i > j$. Every cycle of length n in T starting at a vertex of V_0 defines a vector $(a_1, a_2, \ldots, a_n) \in \Sigma^n$ which is an edge-label sequence. We assume that every vertex and every edge in the tail-biting trellis lies on some cycle. The trellis T is said to represent a block code $\mathcal{C}$ over Σ if the set of all edge-label sequences in T is equal to $\mathcal{C}$. Let $\mathcal{C}(T)$ denote the code represented by the trellis T.

Definition 2.28 *A trellis* T *representing a code* $\mathcal{C}$ *is said to be* one-to-one *if there is a one-to-one correspondence between the cycles in* T *and the codewords in* $\mathcal{C}(T)$*, and it is* reduced *if every vertex and every edge in* T *belongs to at least one cycle.*

In addition to the labeling of edges, each vertex in the set V_i is labeled by a sequence of length $l_i \geq \lceil \log_{|\Sigma|} |V_i| \rceil$ of elements in Σ , all vertex labels at a given depth being distinct. Thus every cycle in this labeled trellis defines a sequence of length $n + l$ (where $l = l_1 + l_2 + \cdots + l_n$) over Σ, consisting of alternating labels of vertices and edges in T. This sequence is called the label sequence of a cycle in T.

Definition 2.29 *The set of all label sequences in a labeled tail-biting trellis* T *denoted* $\mathcal{S}(T)$*, is called the* label code *represented by* T*.*

Figure 2.9 shows a tail-biting trellis for the $(7, 4)_2$ Hamming code defined in Example 2.1.

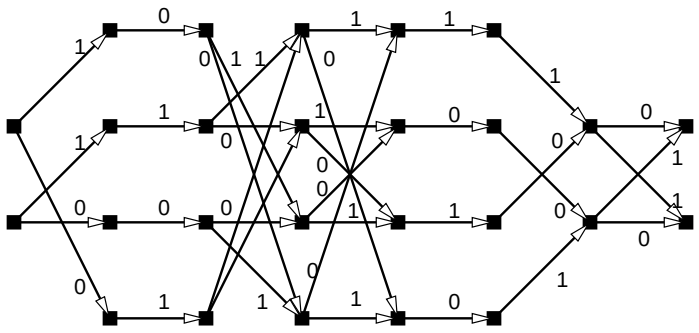


Figure 2.9: A tail-biting trellis for the $(7, 4)_2$ Hamming code.

A number of examples are known where the complexity of a tail-biting trellis for a given code is much lower than the minimal conventional trellis for that code [CFV99, SB00, SKSR01]. A result of Wiberg, Loeliger and Kötter [WLK95] implies that the maximum number of states at any time index in a tail-biting trellis could be as low as the square root of the number of states in a conventional trellis at its midpoint.

Lemma 2.30 *Let $\mathcal{C}$ be a code and let s_{mid} be the minimum possible state-complexity of a conventional trellis for $\mathcal{C}$ at its midpoint, under any coordinate ordering. Then*

$$s_{\max} \geq \left\lceil \frac{s_{\text{mid}}}{2} \right\rceil$$

where $s_{\max}$ is the maximum state-complexity of any tail-biting trellis for $\mathcal{C}$.

The proof of Lemma 2.30 may be found in [CFV99]. We also state a variant of this lower bound due to Reuven and Be'ery [RB99].

Lemma 2.31 (The Square root bound) *Let T be a minimal tail-biting trellis representing linear block code $\mathcal{C}$. Then*

$$s_{\max}(T) \geq \left\lceil \frac{s_{\max}(\mathcal{C})}{2} \right\rceil \tag{2.26}$$

While we have seen that there exists a unique minimal conventional trellis representing any linear block code code [Mud88], this property is not true for tail-biting trellises. A general theory of *linear tail-biting trellises* has been developed by Kötter and Vardy [KV02, KV03], and this is the primary focus of the next chapter.

Chapter 3

Linear Tail-Biting Trellises

The structural and algorithmic results in this thesis pertain to the class of linear tail-biting trellises. The efficient construction of minimal linear tail-biting trellis is as yet an open problem. In this chapter we describe the structure and properties of linear tail-biting trellises. We also review Kötter and Vardy's *characteristic matrix* formulation for the minimal trellis problem. Finally, we describe the *overlayed* structure of linear tail-biting trellises, where a linear tail-biting trellis for a linear block code $\mathcal{C}$ may be constructed by overlaying subtrellises obtained from a coset decomposition of $\mathcal{C}$.

3.1 Introduction

In 1996 McEliece [McE96] introduced the class of *simple linear trellises*, which is the class of trellises computed by the BCJR construction. He thus distinguished between trellises that possess certain linearity properties and those that do not. The class of tail-biting trellises which contains the class of conventional trellises is so broad that nothing much can be said about tail-biting trellises in general. Therefore we restrict ourselves to a study of the class of linear tail-biting trellises which have a rich and beautiful theory [KV02], and this also lays the foundation for the study of minimal linear tail-biting trellises [KV03].

3.2 Definition and Properties of Linear Trellises

We will now develop some definitions and notation [KV02, KV03]. The reader might find it useful to recall the definition of a tail-biting trellis from Section 2.3.

Definition 3.1 *A trellis $T = (V, E, \mathbb{F}_q)$ is said to be* linear *if it is reduced, and there exists a vertex labeling of T such that the label code $\mathcal{S}(T)$ is a vector space.*

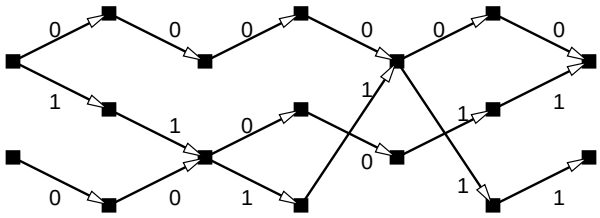


Figure 3.1: A non-linear tail-biting trellis for the $(6, 2)_2$ code.

Note that a necessary condition for a trellis $T = (V, E, \mathbb{F}_q)$ to be linear, is that for every $i \in \{0, 1, \ldots, n-1\}$, $|V_i|$ must be a power of q.

Example 3.2 *Let $\mathcal{C}$ be a $(6, 2)_2$ code with generator matrix*

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

A nonlinear trellis for $\mathcal{C}$ is shown in Figure 3.1.

The notion of non-mergeability [Ksc96, VK96, Var98] will also be useful to us.

Definition 3.3 *A trellis T is* non-mergeable *if there do not exist vertices in the same vertex class of T that can be replaced by a single vertex, while retaining the edges incident on the original vertices, without modifying $\mathcal{C}(T)$.*

Example 3.4 *Consider a $(3, 2)_2$ code with generator matrix $\mathcal{G}$ defined as follows.*

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

A mergeable linear tail-biting trellis for this code is shown in Figure 3.2 – the mergeable vertices are enclosed by dashed ellipses.

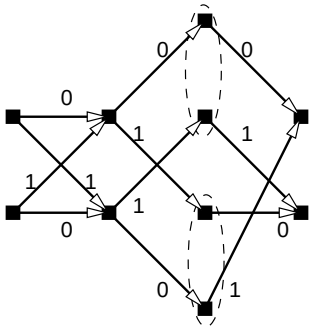


Figure 3.2: A mergeable tail-biting trellis for the $(3, 2)_2$ code.

Kötter and Vardy [KV02] have shown that if a linear trellis is non-mergeable, then it is also biproper. However, though the converse is true for conventional trellises, it is not true in general for tail-biting trellises as illustrated by Figure 3.2. They show that

$$\{\text{linear trellises}\} \supset \{\text{biproper linear trellises}\} \supset \{\text{non–mergeable linear trellises}\} \qquad (3.1)$$

Kötter and Vardy have extended the product construction for conventional trellises described in Section 2.2.4 to tail-biting trellises [KV02]. In particular, they prove that any linear trellis, conventional or tail-biting, for an $(n, k)_q$ linear code $\mathcal{C}$ can be constructed as a *trellis product* of the representation of the individual trellises corresponding to the k rows of the generator matrix G for $\mathcal{C}$. The definition of the trellis product operator is the same as that described in Section 2.2.4. We recall this definition below.

Definition 3.5 *Let $T_1 = (V', E', \mathbb{F}_q)$ and $T_2 = (V'', E'', \mathbb{F}_q)$ be two trellises (either conventional or tail-biting) of depth n. Then the product trellis $T' \times T''$ is the trellis $T = (V, E, \mathbb{F}_q)$ of depth n whose vertex and edge classes are Cartesian products defined as follows.*

$$\begin{aligned} V_i &\stackrel{\text{def}}{=} \{(v', v'') : v' \in V'_i,\ v'' \in V''_i\} \\ E_i &\stackrel{\text{def}}{=} \{(v'_{i-1}, v''_{i-1}), a' + a'', (v'_i, v''_i) : (v'_{i-1}, a', v'_i) \in E'_i, (v''_{i-1}, a'', v''_i) \in E''_i\} \end{aligned}$$

T represents the code $\mathcal{C}$ defined as

$$\mathcal{C} = \mathcal{C}(T_1) + \mathcal{C}(T_2) \stackrel{\text{def}}{=} \{\boldsymbol{c}_1 + \boldsymbol{c}_2 : \boldsymbol{c}_1 \in \mathcal{C}(T_1),\ \boldsymbol{c}_2 \in \mathcal{C}(T_2)\}$$

Let $\mathcal{C}$ be an $(n,k)_q$ code and let $\mathcal{G}$ be its generator matrix. Each row $\boldsymbol{g}_i,\ 1 \leq i \leq k$, in $\mathcal{G}$ generates a one-dimensional subcode of $\mathcal{C}$ which we will denote by $\langle \boldsymbol{g}_i \rangle$. Therefore

$$\mathcal{C} = \langle \boldsymbol{g}_1 \rangle + \langle \boldsymbol{g}_2 \rangle + \cdots + \langle \boldsymbol{g}_k \rangle$$

Thus, if $T_1, T_2, \ldots, T_k$ are trellises for $\langle \boldsymbol{g}_1 \rangle, \langle \boldsymbol{g}_2 \rangle, \ldots, \langle \boldsymbol{g}_k \rangle$ respectively, then their product represents $\mathcal{C}$. Denote by $T_{\boldsymbol{g}_i}$ the minimal trellis for $\langle \boldsymbol{g}_i \rangle$ and define

$$T_{\mathcal{G}} \stackrel{\text{def}}{=} \prod_{i=1}^{k} T_{\boldsymbol{g}_i} \tag{3.2}$$

To specify the component trellises in the trellis product above, we will need to introduce the notions of *linear* and *circular spans*, and *elementary trellises* [KV02]. The primary difference here from the KS product construction is that we allow spans of the form $[i,j]$ such that $i > j$. Given a codeword $\boldsymbol{c} \in \mathcal{C}$, the linear span of $\boldsymbol{c}$ is the smallest interval $[i,j]$, $i,\ j \in \{1,2,\ldots,n\},\ i < j$, that contains all the non-zero positions of $\boldsymbol{c}$. A circular span has exactly the same definition with $i > j$. Note that for a given vector, the linear span is unique, but circular spans are not – they depend on the runs of consecutive zeros chosen for the complement of the span with respect to the index set $\mathcal{I}$. For a vector $\boldsymbol{x} = (x_1, \ldots, x_n)$ over the field $\mathbb{F}_q$ with span $[a,b]$ (either linear or circular), there is a unique elementary trellis representing $\langle \boldsymbol{x} \rangle$ [KV02]. This trellis has q vertices at those positions that belong to $[a,b)$, and a single vertex at other positions. Consequently, T_i in (3.2) is the elementary trellis representing $\langle \boldsymbol{g}_i \rangle$ for some choice of span (either linear or circular).

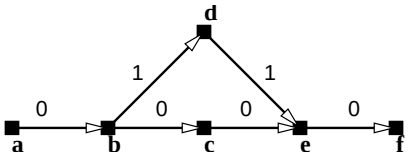


Figure 3.3: Elementary trellis for (0110) with span $[2,3]$.

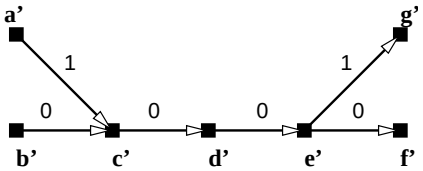


Figure 3.4: Elementary trellis for (1001) with span $[4,1]$.

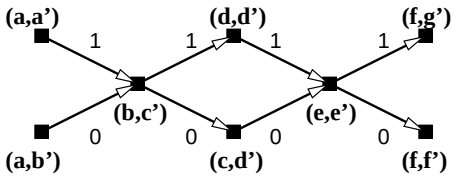


Figure 3.5: The KV product trellis for the $(4,2)_2$ linear code.

The following lemma is due to Kötter and Vardy [KV02].

Lemma 3.6 *A trellis is linear if and only if it factors into a product of elementary trellises.*

Kötter and Vardy [KV02] have also shown that any linear trellis, conventional or tail-biting can be constructed from a generator matrix whose rows can be partitioned into two sets, those that have linear span, and those taken to have circular span. The trellis T representing the code is formed as a product of the elementary trellises corresponding to these rows. We will represent such a generator matrix as

$$\mathcal{G}_{\mathsf{KV}} = \left[\begin{array}{c} \mathcal{G}_l \\ \hline \mathcal{G}_c \end{array}\right] \tag{3.3}$$

where $\mathcal{G}_l$ is the submatrix consisting of rows with linear span, and $\mathcal{G}_c$ the submatrix of rows with circular span. We will henceforth refer to this product trellis T as the KV trellis.

Example 3.7 *Consider a* $(4,2)_2$ *linear block code whose* KV *generator matrix is*

$$\mathcal{G}_{\mathsf{KV}} = \left[\begin{array}{cccc} 0 & 1 & 1 & 0 \\ \hline 1 & 0 & 0 & 1 \end{array}\right] \begin{array}{c} [2,3] \\ [4,1] \end{array}$$

The spans and elementary trellises for the rows 0110 *and* 1001 *are shown in Figures 3.3 and 3.4 respectively. The resulting* KV *product trellis is shown in Figure 3.5.*

Even though a minimal-span form generator matrix for a linear code yields the minimal conventional product trellis, this is not so for tail-biting trellises. Therefore, the computation of minimal product tail-biting trellises also entails the problem of choosing appropriate spans for the rows of the generator matrix.

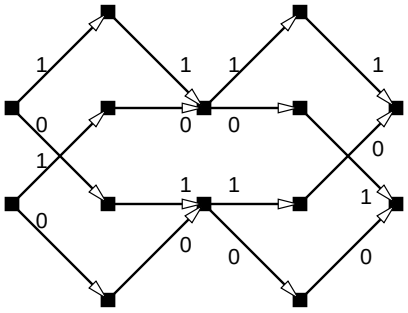


Figure 3.6: A non-minimal tail-biting trellis for the $(4,2)_2$ code.

Example 3.8 *For the* $(4,2)_2$ *code with product generator matrix*

$$\mathcal{G} = \left[\begin{array}{cccc} 1 & 0 & 0 & 1 \\ \hline 0 & 1 & 1 & 0 \end{array}\right] \begin{array}{c} [1,4] \\ [3,2] \end{array}$$

we obtain the non-minimal product trellis shown in Figure 3.6.

As the matrices that we consider in this thesis are full rank matrices, *we will specify each matrix as a set, with elements being the rows of the matrix.* Therefore all set operators apply to these matrices as well. For example, let $M \in \mathbb{F}_q^{m \times n}$ – then $M \cup \{\boldsymbol{g}\}$, $\boldsymbol{g} \in \mathbb{F}_q^{1 \times n}$, denotes the $(m+1) \times n$ matrix formed by adding a row $\boldsymbol{g}$ to M.

3.3 Computing Minimal Tail-Biting Trellises

Kötter and Vardy [KV03] define various notions of minimality for linear trellises and show that these minimal trellises are all computable from a certain *characteristic matrix* for the code under consideration. We will first present various definitions of minimality for tail-biting trellises and then describe the computation of the characteristic matrix.

3.3.1 Definitions and Notation

Let $T = (V, E, \mathbb{F}_q)$ be a trellis, either conventional or tail-biting for an $(n,k)_q$ linear block code $\mathcal{C}$. Let $\Theta(\mathcal{C})$ denote the SCP of T. The SCP's of all possible trellises for $\mathcal{C}$ form a partially ordered set under the operation of component-wise comparison. We say that a trellis $T_1 = (V', E', \mathbb{F}_q)$ is smaller than or equal to a trellis $T_2 = (V'', E'', \mathbb{F}_q)$, denoted by

$T_1 \preceq_\Theta T_2$, if

$$|V_i'| \leq |V_i''|, \quad \forall i \in \{1, 2, \ldots, n-1\} \tag{3.4}$$

If equality does not hold for any i in Equation (3.4), then we write $T_1 \prec_\Theta T_2$.

Definition 3.9 *A trellis T is said to be $\prec_\Theta$-*minimal*, if there does not exist a trellis T' such that $T' \prec_\Theta T$.*

In contrast to Muder's notion of minimality (see Definition 2.10), this is a rather weak notion of minimality. For a given code $\mathcal{C}$, there may be several $\prec_\Theta$-minimal trellises representing $\mathcal{C}$.

Example 3.10 *For the $(4,2)_2$ code, the trellises in Figure 2.3 and Figure 3.5 are both $\prec_\Theta$-minimal.*

In fact, any trellis with a state-complexity of zero at any time index is $\prec_\Theta$-minimal. There are also other measures of minimality for tail-biting trellises [KV03], which we will now present.

Definition 3.11 *A* total order *$\prec_\mathcal{O}$ on a set S is an order under which any two elements of S are comparable.*

For example, the ordering induced by $\prec_\Theta$ on the set of trellises is not a total order, since for every linear code there are many $\prec_\Theta$-minimal trellises that are incomparable. Let $T = (V, E, \mathbb{F}_q)$ and $T' = (V', E', \mathbb{F}_q)$ be trellises of depth n. Kötter and Vardy define the following total orders.

product order: $T \preceq_\Pi T'$ if $\prod_{i=0}^{n-1} |V_i| \leq \prod_{i=0} |V_i'|$.

max order: $T \preceq_{\max} T'$ if $\max_i |V_i| \leq \max_i |V_i'|$.

vertex sum order: $T \preceq_\Sigma T'$ if $\sum_{i=0}^{n-1} |V_i| \leq \sum_{i=0}^{n-1} |V_i'|$.

edge-product order: $T \preceq_{\Pi\mathcal{E}} T'$ if $\prod_{i=0}^{n-1} |E_i| \leq \prod_{i=0}^{n-1} |E_i'|$.

edge max order: $T \preceq_{\max \mathcal{E}} T'$ if $\max_i |E_i| \leq \max_i |E_i'|$.

edge sum order: $T \preceq_{\Sigma\mathcal{E}} T'$ if $\sum_{i=0}^{n-1} |E_i| \leq \sum_{i=0}^{n-1} |E_i'|$.

Definition 3.12 *A trellis T is $\prec_{\Pi}$-*minimal *if there does not exist a trellis T' such that $T' \prec_{\Pi} T$. A trellis T is $\prec_{\max}$-*minimal *if there is no trellis T' such that either $T' \prec_{\max} T$, or $s_{\max}(T') = s_{\max}(T)$ and $T' \prec_{\Theta} T$.*

Definition 3.13 *An order $\prec_{\mathcal{O}}$* preserves *the order described by $\prec_{\Theta}$, if $T \preceq_{\Theta} T'$ implies that $T \prec_{\mathcal{O}} T'$.*

The orders $\prec_{\Pi}$, $\prec_{\max}$, $\prec_{\Sigma}$ and $\prec_{\Pi\mathcal{E}}$ preserve $\prec_{\Theta}$, while the orders $\prec_{\max\mathcal{E}}$ and $\prec_{\Sigma\mathcal{E}}$ do not [KV03].

Proposition 3.14 ([KV03]) *The set of $\prec_{\Pi}$-minimal trellises and the set of $\prec_{\max}$-minimal trellises for a given code are subsets of the set of $\prec_{\Theta}$-minimal trellises.*

3.3.2 The Characteristic Matrix

Given an $(n,k)_q$ code $\mathcal{C}$, whether there exists a polynomial time computation of a generator matrix yielding a $\prec_{\max}$-minimal trellis representing $\mathcal{C}$ is still an open question. The feasible space of linear trellises representing $\mathcal{C}$ has cardinality equal to $\mathcal{O}(q^{k^2}n^k)$ [KV03]. Kötter and Vardy [KV03] reduce the size of this search space for minimal trellises to $\mathcal{O}(n^k)$. Specifically, they prove that every $\prec_{\Theta}$-minimal linear trellis for an $(n,k)_q$ code $\mathcal{C}$ can be computed from an $n \times n$ *characteristic matrix* for $\mathcal{C}$. They also describe an $\mathcal{O}(k^2)$ procedure to compute a product generator matrix for a $\prec_{\Pi}$-minimal trellis for $\mathcal{C}$. Further, they formulate the problem of computing an $\prec_{\max}$-minimal trellis as a linear program (LP), and propose an algorithm to solve this LP that is efficient in practice. The worst case complexity of their algorithm is $\mathcal{O}(n^k)$. The description of this algorithm may be found in [KV03]. It turns out that polynomial time algorithms to compute minimal trellises for certain subclasses of linear block codes, for example, a subclass of cyclic codes, exist [SKSR01].

Some properties of $\prec_{\Theta}$-minimal trellises will be stated before we give the description of the characteristic matrix computation for a code. Let $T_{\mathcal{G}}$ be a $\prec_{\Theta}$-minimal KV trellis for an $(n,k)_q$ code $\mathcal{C}$ with generator matrix $\mathcal{G}_{KV}$.

The following lemma characterizes the structure of the generator matrix for $\prec_{\Theta}$-minimal tail-biting trellises analogous to that given in Lemma 2.23 for conventional trellises.

Lemma 3.15 ([KV03]) *The* KV *trellis $T_{\mathcal{G}}$ is a $\prec_{\Theta}$-minimal trellis only if $\mathcal{G}_{\mathsf{KV}}$ does not have rows with spans that start at the same position or end at the same position.*

For all $\boldsymbol{x} \in \mathbb{F}_q^n$, let $\sigma_i(\boldsymbol{x})$ ($\rho_i(\boldsymbol{x})$) denote a cyclic shift to the left (right) i times of $\boldsymbol{x}$, where $i \in \{0, 1, 2, \ldots, n-1\}$. Given a matrix M, let M_i^* denote a minimal-span form (see Section 2.23) basis for the vector space $\sigma_i(\langle M \rangle)$.

Definition 3.16 *A* characteristic generator *for an $(n,k)_q$ code $\mathcal{C}$ with generator matrix $\mathcal{G}$ is a pair is a pair $(\boldsymbol{c}, [i,j])$, where $\boldsymbol{c} \in \mathcal{C}$, and $[i,j]$ is either a linear or circular span for $\boldsymbol{c}$ defined as follows. The set X of characteristic generators for $\mathcal{C}$ is given by*

$$X \stackrel{\text{def}}{=} \mathcal{G}_0^* \cup \rho_1(\mathcal{G}_1^*) \cup \cdots \cup \rho_{n-1}(\mathcal{G}_{n-1}^*) \tag{3.5}$$

where each $\boldsymbol{x} \in \rho_i(\mathcal{G}_i^*)$ has span $[(a+i) \bmod n, (b+i) \bmod n]$, $[a,b]$ being the linear span of the vector $\sigma_i(\boldsymbol{x})$. The characteristic matrix *$\chi(\mathcal{C})$ for $\mathcal{C}$ is the $|X| \times n$ matrix having the elements of X as its rows. When taking the union in (3.5), if two vectors have the same span (either linear or circular), only one of them is chosen.*

Example 3.17 *Consider a $(4,2)_2$ code $\mathcal{C}$ with generator matrix*

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

The characteristic matrix $\chi(\mathcal{C})$ for $\mathcal{C}$ is a 4×4 matrix given by

$$\chi(\mathcal{C}) = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \begin{matrix} [2,3] \\ [1,4] \\ [3,2] \\ [4,1] \end{matrix}$$

The trellis in Figure 3.5 is a $\prec_\Theta$-minimal trellis computed by the trellis product of elementary trellises representing the characteristic generators $(0110, [2,3])$ and $(1001, [4,1])$.

We state without proof some properties of the characteristic matrix [KV02].

Lemma 3.18 *The spans of any two characteristic generators start at different positions and also end at different positions.*

Lemma 3.19 *The cardinality of the set X of characteristic generators for an $(n,k)_q$ code is equal to n. Therefore the characteristic matrix $\chi(\mathcal{C})$ is an $n \times n$ matrix.*

Lemma 3.20 *Every $\prec_\Theta$-minimal linear tail-biting trellis for an $(n,k)_q$ code $\mathcal{C}$ can be constructed as a trellis product of k linearly independent characteristic generators for $\mathcal{C}$.*

Given an $(n,k)_q$ code $\mathcal{C}$, the computation of $\chi(\mathcal{C})$ can be performed in time $\mathcal{O}(n^2)$ [KV03]. Moreover, from Lemma 3.20, it follows that the feasible space for the computation of a $\prec_\Theta$-minimal trellis has cardinality $\mathcal{O}(n^k)$.

3.4 Overlayed Structure of Linear Tail-Biting Trellises

We will now examine the overlayed structure of linear tail-biting trellises. The main idea here is to view a tail-biting trellis $T = (V, E, \mathbb{F}_q)$ as a collection of conventional trellises, each corresponding to a different initial vertex in V_0. In particular, we will see that every linear tail-biting trellis always has an overlayed structure [DSDR00, LS00, SB00, SDDR03, SvT79].

Definition 3.21 *Let $T = (V, E, \mathbb{F}_q)$ be a tail-biting trellis. The subgraph of T induced by all cycles containing a fixed vertex of V_0 is called a* subtrellis *of T.*

By definition, a subtrellis is a conventional trellis. If $\mathcal{T}$ is the set of all subtrellises of T, then

$$\mathcal{C}(T) = \bigcup_{T' \in \mathcal{T}} \mathcal{C}(T') \tag{3.6}$$

For the special case of linear tail-biting trellises, the subtrellises are isomorphic (upto the labeling of vertices) as indicated in the following lemma [SB00].

Lemma 3.22 *Let T be a linear tail-biting trellis for a linear code $\mathcal{C}$. Then every subtrellis represents a coset of a fixed subcode of $\mathcal{C}$. Moreover, all subtrellises of T are isomorphic.*

Given a trellis T, we will refer to any trellis that is structurally isomorphic to T as a copy of T. We will now establish a connection between the KV construction and the overlayed structure of $\prec_{\max}$-minimal tail-biting trellises. The following property follows from Propositions 3 and 4 of [SB00].

Lemma 3.23 (Intersection Property) *Let $\mathcal{G}$ be a KV generator matrix for a $\prec_{\max}$-minimal trellis. Define a* zero-run *of a circular span generator in $\mathcal{G}$ to be the complement of its span. Then every pair of circular span generators in $\mathcal{G}$ has intersecting zero-runs.*

The following lemma follows from Theorem 3.2 in [DSDR00].

Lemma 3.24 *Let T be a biproper linear tail-biting trellis for a linear block code $\mathcal{C}$ over $\mathbb{F}_q$ computed by a* KV *construction using the matrix $\mathcal{G} = \left[\begin{array}{c} \mathcal{G}_l \\ \hline \mathcal{G}_c \end{array}\right]$, with $\mathcal{G}_l$ having l rows and $\mathcal{G}_c$ having c rows. Let T_l be the minimal conventional trellis for the generators in $\mathcal{G}_l$. Then T has the following properties.*

(i) *The trellis T has q^c start and final states. It has q^c subtrellises that are copies of T_l. Each subtrellis represents a coset in the coset decomposition of $\mathcal{C}$ over the subcode generated by $\mathcal{G}_l$.*

(ii) *Let t_i be the number of zero-runs in $\mathcal{G}_c$ that contain time index i. Then there are q^{c-t_i} groups of subtrellises, each containing q^{t_i} trellises at time index i. Each subtrellis has q^{l_i} states at time index i, l_i being the state-complexity of T_l at that time index.*

Definition 3.25 *Every pair of subtrellises T and T' whose coset leaders*[1] *have a maximum zero-run intersection equal to $[i, j]$, share all states in $[i, j]$ and no states outside $[i, j]$. We will refer to $[i, j]$ as the* merging interval *of T and T'. If $[i, j]$ is the zero-run of a coset leader $\boldsymbol{v}_l$ of the coset $\mathcal{C}_l$, we will refer to $[i, j]$ as the merging interval of $\boldsymbol{v}_l$. Note that this corresponds to the merging interval of the subtrellises $T(\mathcal{C}_l)$ and $T(\mathcal{C}_0)$.*

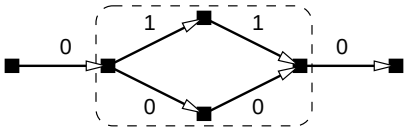


Figure 3.7: A minimal trellis representing the code $\mathcal{C}_0 = \{0000, 0110\}$.

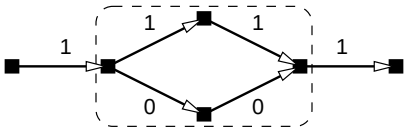


Figure 3.8: A minimal trellis representing the coset $\mathcal{C}_1 = \{1001, 1111\}$.

Examples 3.26 and 3.27 illustrate the trellis overlaying technique.

[1] *if $\mathcal{C}_i = \mathcal{C}_0 + \boldsymbol{v}_i$ is a coset in $\mathcal{C}/\mathcal{C}_0$, then $\boldsymbol{v}_i$ is the* coset leader *for $\mathcal{C}_i$.*

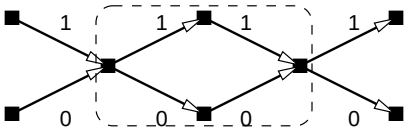


Figure 3.9: An overlayed trellis for the $(4,2)_2$ code.

Example 3.26 *Consider a* $(4,2)_2$ *code* $\mathcal{C}$ *with a* KV *generator matrix* $\mathcal{G}$ *defined as*

$$\mathcal{G}_{\mathsf{KV}} = \left[\begin{array}{cccc} 0 & 1 & 1 & 0 \\ \hline 1 & 0 & 0 & 1 \end{array}\right] \begin{array}{c} [2,3] \\ [4,1] \end{array}$$

The linear subcode $\mathcal{C}_0$ *representing* $\langle \mathcal{G}_l \rangle$ *is* $\mathcal{C}_0 = \{0000, 0110\}$*, and the coset* $\mathcal{C}_1$ *in* $\mathcal{C}/\mathcal{C}_0$ *is* $\mathcal{C}_1 = \{1001, 1111\}$*. The vector* 1001 *is the coset leader for coset* $\mathcal{C}_1$ *with merging interval* $[2,3]$*. The overlayed trellis representing* $\mathcal{C}$ *shown in Figure 3.9 is obtained by overlaying the trellises in Figures 3.7 and 3.8. The overlayed portions are enclosed by the dashed boxes.*

Example 3.27 *Let* $\mathcal{C}$ *be a* $(7,4)_2$ *Hamming code a product generator matrix* $\mathcal{G}_{\mathsf{KV}}$ *defined as*

$$\mathcal{G}_{\mathsf{KV}} = \left[\begin{array}{ccccccc} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ \hline 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{array}\right] \begin{array}{c} [1,6] \\ [3,7] \\ [6,2] \\ [7,4] \end{array}$$

The linear subcode $\mathcal{C}_0$ *representing* $\langle \mathcal{G}_l \rangle$ *is* $\mathcal{C}_0 = \{0000000, 1000110, 0010111, 1010001\}$*, and the cosets in* $\mathcal{C}/\mathcal{C}_0$ *are*

$$\begin{aligned} \mathcal{C}_1 &= \{0100011, 1100101, 0110100, 1110010\} \\ \mathcal{C}_2 &= \{0111001, 1111111, 0101110, 1101000\} \\ \mathcal{C}_3 &= \{0011010, 1011100, 0001101, 1001011\} \end{aligned}$$

The vector 0100011 *is the coset leader for coset* $\mathcal{C}_1$ *with merging interval* $[3,5]$*, and the vector* 0111001 *is the coset leader for coset* $\mathcal{C}_2$ *with merging interval* $[5,6]$*. The overlayed trellis representing* $\mathcal{C}$ *shown in Figure 3.14 is obtained by overlaying the trellises in Figures 3.10, 3.11, 3.11 and 3.12.*

We note here that defining an overlayed structure needs a specification of cosets, as well as *coset leaders*.

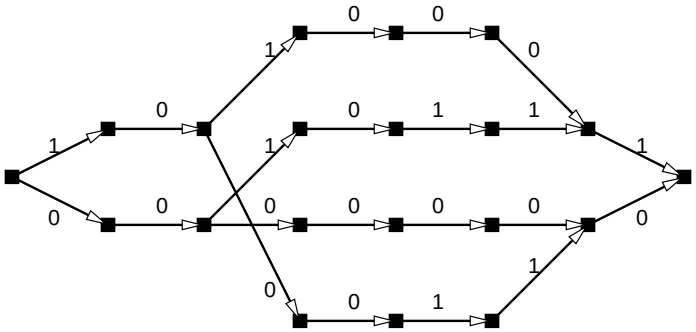


Figure 3.10: A minimal trellis representing the code $\mathcal{C}_0 = \{0000000, 1000110, 0010111, 1010001\}$.

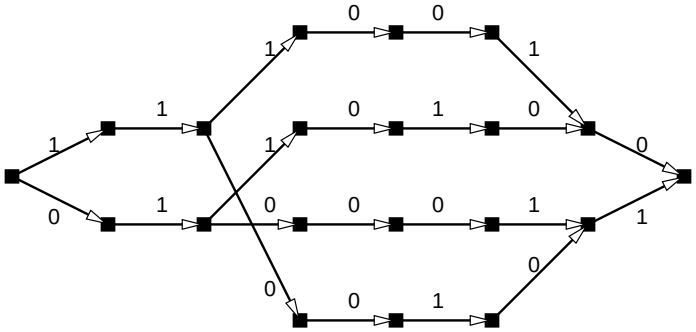


Figure 3.11: A minimal trellis representing the coset $\mathcal{C}_1 = \{0100011, 1100101, 0110100, 1110010\}$.

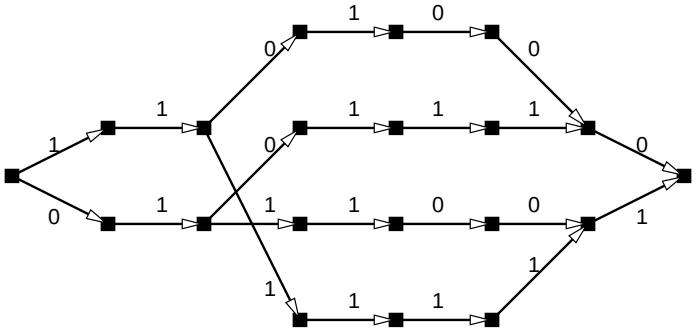


Figure 3.12: A minimal trellis representing the coset $\mathcal{C}_2 = \{0111001, 1111111, 0101110, 1101000\}$.

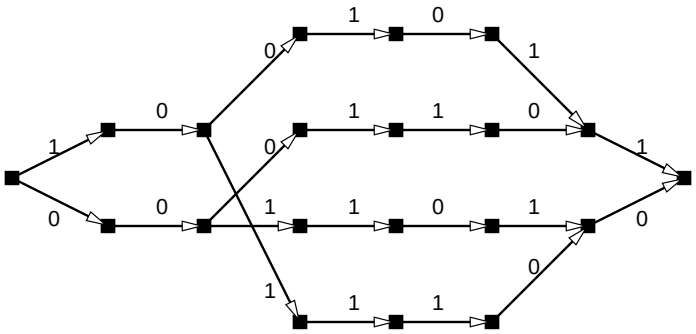


Figure 3.13: A minimal trellis representing the coset $\mathcal{C}_3 = \{0011010, 1011100, 0001101, 1001011\}$.

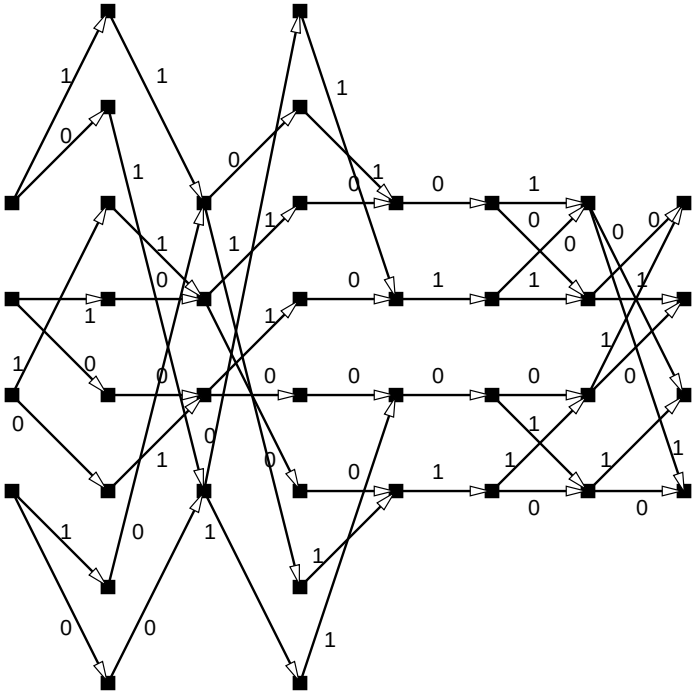


Figure 3.14: An overlayed trellis for the $(7,4)_2$ Hamming code.

Chapter 4

The Tail-Biting BCJR Trellis

In this chapter we generalize the BCJR construction for conventional trellises described in Section 2.2.1 to obtain tail-biting trellises. We show that a linear tail-biting trellis for an $(n,k)_q$ code $\mathcal{C}$ can be constructed by specifying an arbitrary parity check matrix $\mathcal{H}$ for $\mathcal{C}$ along with an $(n-k)\times k$ *displacement matrix* $\mathcal{D}$, with coefficients from $\mathbb{F}_q$. The set of BCJR labels is computed from $\mathcal{H}$ and $\mathcal{D}$. Finally, we present a dynamic algorithm that starts with a generator matrix for an $(n,k)_q$ code in trellis-oriented form, and computes $\mathcal{D}$ for a minimal trellis in time $\mathcal{O}(n^k)$.

4.1 The Tail-biting BCJR Trellis

The BCJR algorithm described in Section 2.2.1 is now generalized to construct labeled tail-biting trellises for any linear code with the resultant trellis satisfying the following two properties.

(i) The trellis formed is biproper and linear.

(ii) The state labels at each time index form a vector space.

Let $\mathcal{C}$ be an $(n,k)_q$ linear block code with generator matrix $\mathcal{G} = \{\boldsymbol{g}_1, \boldsymbol{g}_2, \ldots, \boldsymbol{g}_k\}$, where $\boldsymbol{g}_i \in \mathbb{F}_q^{1\times n},\ 1 \le i \le k$, are the k rows of $\mathcal{G}$, and parity check matrix $\mathcal{H} = [\boldsymbol{h}_1 \boldsymbol{h}_2 \cdots \boldsymbol{h}_n]$, with columns $\boldsymbol{h}_i \in \mathbb{F}_q^{(n-k)\times 1},\ 1 \le i \le n$.

The tail-biting BCJR specification includes an $(n-k)\times 1$ displacement vector $\boldsymbol{d_g}$ with every generator row $\boldsymbol{g} \in \mathcal{G}$. Specifically, the set of displacement vectors for an $k \times n$ generator

matrix $\mathcal{G}$ is an arbitrary set of k vectors from $\mathbb{F}_q^{(n-k)\times k}$, which could be linearly independent or dependent, with repetitions allowed. The displacement vector $\boldsymbol{d_c}$ for any codeword $\boldsymbol{c} \in \mathcal{C}$ is defined as follows.

$$\boldsymbol{d_c} = \sum_{i=1}^{k} \alpha_i \boldsymbol{d_{g_i}}, \text{ where } \boldsymbol{c} = \sum_{i=1}^{k} \alpha_i \boldsymbol{g}_i, \ \alpha_i \in \mathbb{F}_q, \ \boldsymbol{g}_i \in \mathcal{G}. \tag{4.1}$$

Definition 4.1 *Given an $(n,k)_q$ linear code $\mathcal{C}$ with generator matrix $\mathcal{G}$ and displacement vectors $\{\boldsymbol{d_g}\}_{\boldsymbol{g}\in\mathcal{G}}$, the $(n-k)\times k$ matrix $\mathcal{D}$ whose i^{th} column is equal to $\boldsymbol{d_{g_i}}$, where $\boldsymbol{g}_i$ is the i^{th} row of $\mathcal{G}$, $1 \le i \le k$, is called a* displacement matrix *for $\mathcal{C}$.*

The following lemma elucidates the structure imposed by the displacement matrix on the code.

Lemma 4.2 *Let $\mathcal{C}$ be an $(n,k)_q$ linear code with generator matrix $\mathcal{G}$ and displacement matrix $\mathcal{D}$. Then $\mathcal{D}$ specifies a coset decomposition $\mathcal{C}/\mathcal{C}_0$ of the code $\mathcal{C}$, where $\mathcal{C}_0 = \{\boldsymbol{c} \in \mathcal{C} : \boldsymbol{d_c} = \boldsymbol{0}\}$.*

Proof: It is easily seen that $\mathcal{C}_0 = \{\boldsymbol{c} \in \mathcal{C} : \boldsymbol{d_c} = \boldsymbol{0}\}$ is a linear subcode of $\mathcal{C}$. Consider a coset $\mathcal{C}_i = \mathcal{C}_0 + \boldsymbol{v}_i, \boldsymbol{v}_i \in \mathcal{C} \setminus \mathcal{C}_0$. Since every codeword $\boldsymbol{x} \in \mathcal{C}_i$ takes the form $\boldsymbol{x} = \boldsymbol{c} + \boldsymbol{v}_i$ for some $\boldsymbol{c} \in \mathcal{C}_0$, it follows that $\boldsymbol{d_x} = \boldsymbol{d_{v_i}}$ (since $\boldsymbol{d_c} = \boldsymbol{0}$). Therefore, every coset is associated with a unique displacement vector and the lemma follows. ■

It should be noted that corresponding to any coset decomposition of a code $\mathcal{C}$, there can be many displacement matrices inducing that decomposition.

Example 4.3 *Let $\mathcal{C}$ be a self dual $(4,2)_2$ code with generator matrix $\mathcal{G}$ as follows.*

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

Let the displacement matrix $\mathcal{D} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$. Therefore $\boldsymbol{d}_{0110} = \binom{0}{0}$ and $\boldsymbol{d}_{1001} = \binom{0}{1}$, and from Equation 4.1, we have $\boldsymbol{d}_{0000} = \binom{0}{0}$ and $\boldsymbol{d}_{1111} = \binom{0}{1}$. The matrix $\mathcal{D}' = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}$ also induces the same coset decomposition as the matrix $\mathcal{D}$.

Definition 4.4 (T–BCJR labeling) *Every codeword* $\boldsymbol{c} = (c_1, c_2, \ldots, c_n) \in \mathcal{C}$ *induces a sequence of states* $\{\boldsymbol{s}_i\}_{i=0}^{n-1}$*, each state being labeled by a vector in* $\mathbb{F}_q^{(n-k)\times 1}$ *as follows.*

$$\boldsymbol{s}_i = \begin{cases} \boldsymbol{d_c} & \text{if } i = 0 \\ \boldsymbol{s}_{i-1} + c_i \boldsymbol{h}_i & \text{otherwise} \end{cases} \tag{4.2}$$

There is an edge labeled $a \in \mathbb{F}_q$ *from state* $\boldsymbol{s}_{i-1}$ *to state* $\boldsymbol{s}_i$*,* $1 \leq i \leq n-2$*, if and only if*

$$\boldsymbol{s}_i = \boldsymbol{s}_{i-1} + a\boldsymbol{h}_i \tag{4.3}$$

We refer to such a labeling as a Tail-biting BCJR *(*T–BCJR*) trellis.*

Definition 4.5 *Let* $\mathcal{C}$ *be an* $(n,k)_q$ *code with parity check matrix* $\mathcal{H}$*, generator matrix* $\mathcal{G}$ *and a displacement matrix* $\mathcal{D}$*. The* label generator matrix $\mathcal{G}_{\mathcal{H},\mathcal{D}}$ *is defined to be the matrix consisting of* BCJR *label rows of* $\mathcal{G}$ *formed with respect to* $\mathcal{H}$ *and* $\mathcal{D}$*.*

Let $\mathcal{C}$ be an $(n,k)_q$ code with generator matrix $\mathcal{G}$ and parity check matrix $\mathcal{H}$. The T–BCJR trellis T constructed with respect to $\mathcal{H}$, $\mathcal{G}$ and some arbitrary $\mathcal{D} \in \mathbb{F}_q^{(n-k)\times k}$ is thus given by

$$T = \langle \mathcal{G}_{\mathcal{H},\mathcal{D}} \rangle \tag{4.4}$$

where we recall that $\langle \mathcal{G}_{\mathcal{H},\mathcal{D}} \rangle$ is the vector space generated by the rows of $\langle \mathcal{G}_{\mathcal{H},\mathcal{D}} \rangle$.

Example 4.6 *For the* $(4,2)_2$ *code from Example 4.10,*

$$\mathcal{G}_{\mathcal{H},\mathcal{D}} = \begin{bmatrix} \binom{0}{0} & 0 & \binom{0}{0} & 1 & \binom{1}{0} & 1 & \binom{0}{0} & 0 & \binom{0}{0} \\ \binom{0}{1} & 1 & \binom{0}{0} & 0 & \binom{0}{0} & 0 & \binom{0}{0} & 1 & \binom{0}{1} \end{bmatrix}$$

and $\langle \mathcal{G}_{\mathcal{H},\mathcal{D}} \rangle$ *is the* T–BCJR *trellis shown in Figure 4.1.*

Let T be a T–BCJR trellis constructed for $\mathcal{C}$. The properties of T are elucidated by the following lemmas.

Lemma 4.7 *The trellis* T *is linear and represents* $\mathcal{C}$*.*

Proof: We first prove that $\mathcal{C}(T) = \mathcal{C}$. Assume to the contrary that $\exists \boldsymbol{x} = (x_1, \ldots, x_n) \in \mathcal{C}(T)$ such that $\boldsymbol{x} \notin \mathcal{C}$. Let $\boldsymbol{d_x}$ be the start vertex of the cycle representing the word $\boldsymbol{x}$. The

invariant maintained by the algorithm for every edge $\boldsymbol{e} = (\boldsymbol{v}_{i-1}, a, \boldsymbol{v}_i) \in V_{i-1} \times \mathbb{F}_q \times V_i$ is

$$\boldsymbol{v}_{i-1} + a\boldsymbol{h}_i = \boldsymbol{v}_i$$

Therefore, $\boldsymbol{d_x} + \sum_{i=1}^{n} x_i \boldsymbol{h}_i = \boldsymbol{d_x} \Rightarrow \mathcal{H}\boldsymbol{x}^t = \boldsymbol{0} \Rightarrow \boldsymbol{x} \in \mathcal{C}$, thus contradicting our original assumption.

Let $\boldsymbol{x},\ \boldsymbol{y} \in \mathcal{C}(T)$ and let $\boldsymbol{x}',\ \boldsymbol{y}' \in \mathcal{S}(T)$ respectively, be their label codewords. Since $\mathcal{C}(T) = \mathcal{C}$, we have $\boldsymbol{z} = \boldsymbol{x} + \boldsymbol{y} \in \mathcal{C}(T)$. In order to prove linearity of T, we need to show that $\boldsymbol{z}' = \boldsymbol{x}' + \boldsymbol{y}'$ also belongs to $\mathcal{S}(T)$. We have $\boldsymbol{x}' = \langle \boldsymbol{d_x}, x_1, \boldsymbol{u}_1, \ldots, x_n, \boldsymbol{d_x} \rangle$, such that $\boldsymbol{u}_i \mid_{1 \le i \le n} = \boldsymbol{d_x} + \sum_{j=1}^{i} x_j \boldsymbol{h}_j$, and $\boldsymbol{y}' = \langle \boldsymbol{d_y}, y_1, \boldsymbol{v}_1, \ldots, y_n, \boldsymbol{d_y} \rangle$, such that $\boldsymbol{v}_i \mid_{1 \le i \le n} = \boldsymbol{d_y} + \sum_{j=1}^{i} y_j \boldsymbol{h}_j$. Therefore

$$\boldsymbol{z}' = \langle \boldsymbol{d_x} + \boldsymbol{d_y}, x_1 + y_1, \boldsymbol{u}_1 + \boldsymbol{v}_1, \ldots, \boldsymbol{x}_n + \boldsymbol{y}_n, \boldsymbol{d_x} + \boldsymbol{d_y} \rangle = \langle \boldsymbol{d_z}, z_1, \boldsymbol{d_z} + z_1 \boldsymbol{h}_1, \ldots, \boldsymbol{z}_n, \boldsymbol{d_z} \rangle$$

which shows that $\boldsymbol{z}'$ is the label codeword in $\mathcal{S}(T)$ representing the codeword $\boldsymbol{z}$, thus proving that T is indeed a linear trellis representing $\mathcal{C}$. ■

Lemma 4.8 *The trellis T is biproper.*

Proof: Assume there exists a vertex $\boldsymbol{v}$ at some time index i, with two outgoing edges $(\boldsymbol{v}, a, \boldsymbol{v}_1)$ and $(\boldsymbol{v}, a, \boldsymbol{v}_2)$, $\boldsymbol{v}_1 \neq \boldsymbol{v}_2$ (that is, T is not proper). From the BCJR construction we know that $\boldsymbol{v} + a\boldsymbol{h}_{i+1} = \boldsymbol{v}_1$ and $\boldsymbol{v} + a\boldsymbol{h}_{i+1} = \boldsymbol{v}_2 \Rightarrow \boldsymbol{v}_1 = \boldsymbol{v}_2$, which contradicts our assumption that $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ are distinct. Therefore T must be proper. A similar argument shows that T must also be co-proper (that is, T with edges reversed is also proper), thus proving that T is a biproper trellis. ■

Let $\mathcal{G}_i$ and $\mathcal{H}_i$ respectively, denote the submatrices consisting of the first i columns of $\mathcal{G}$ and $\mathcal{H}$. For every $i \in \{0, 1, \ldots, n-1\}$, define a matrix M_i as follows.

$$M_i = \left[\mathcal{H}_i {\mathcal{G}_i}^t + \mathcal{D}\right] \tag{4.5}$$

Then we have the following lemma, which is a generalization of Lemma 2.15 for the conventional BCJR trellis.

Lemma 4.9 (State-Space Lemma) *For all time indices $i \in \mathcal{I}$, V_i the state space of the trellis T at time index i equals the column-space of M_i.*

Proof: For every generator $\boldsymbol{g} = (g_1, g_2, \ldots, g_n) \in \mathcal{G}$, the state $\boldsymbol{v}_i$ at time index i is given by $\boldsymbol{v}_i = \boldsymbol{d_g} + \mathcal{H}_i(g_1, g_2, \ldots, g_i)^t$. Therefore the set of states at time index i for generators in $\mathcal{G}$ is defined by the columns of $M_i = [\mathcal{H}_i \mathcal{G}_i{}^t + \mathcal{D}]$, and the lemma follows. ■

Example 4.10 *For the $(4,2)_2$ code defined in Example 4.3, the* T–BCJR *trellis shown in Figure 4.1.*

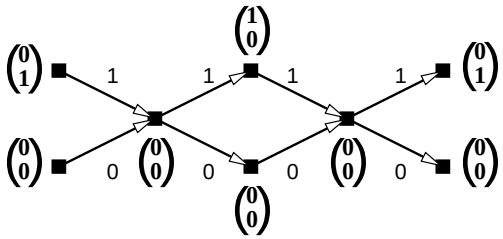


Figure 4.1: A T–BCJR trellis for the $(4,2)_2$ code.

Example 4.11 *Consider the $(7,4)_2$ Hamming code $\mathcal{C}$ with parity check and generator matrices defined as follows.*

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

Choosing $\mathcal{D} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$*, we obtain a minimal* T–BCJR *trellis T for $\mathcal{C}$ with*

$s_{\max}(T) = 2$*, as illustrated in Figure 4.2.*

We will now define a displacement matrix derived directly from a KV product specification of a non-mergeable linear trellis.

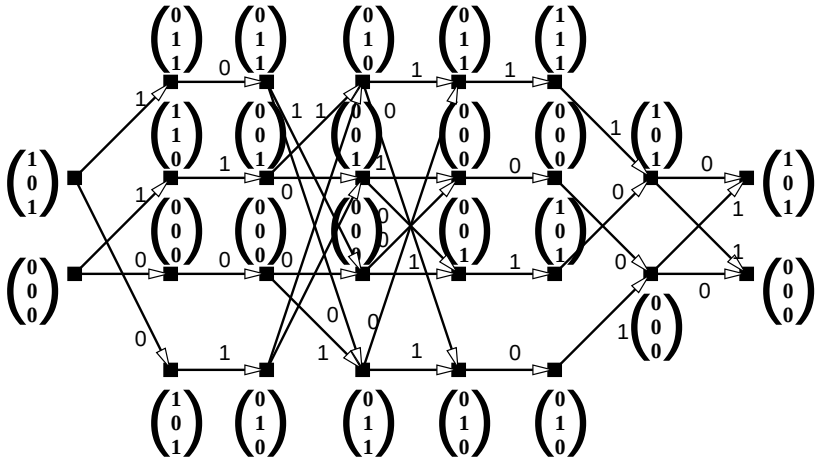


Figure 4.2: A minimal T–BCJR trellis for the $(7,4)_2$ Hamming code.

Definition 4.12 *Let $\mathcal{G}$ be a* KV *product matrix for a non-mergeable linear trellis T representing an $(n,k)_q$ code $\mathcal{C}$. Let the parity check matrix for $\mathcal{C}$ be $\mathcal{H} = [\boldsymbol{h}_1 \boldsymbol{h}_2 \cdots \boldsymbol{h}_n]$, with columns $\boldsymbol{h}_i \in \mathbb{F}_q^{(n-k)\times 1}$, $1 \leq i \leq n$. A* T–BCJR *labeling for T is defined by the displacement matrix $\mathcal{D}$ with respect to $\mathcal{H}$ and $\mathcal{G}$ as follows.*

The i^{th} column of $\mathcal{D}$ is equal to $\mathbf{0}$ if the i^{th} row of $\mathcal{G}$ has linear span, else it is equal to $\boldsymbol{d}_{\boldsymbol{g}} = \sum_{j=a}^{n} g_j \boldsymbol{h}_j$, where $\boldsymbol{g} = (g_1, g_2, \ldots, g_n)$ is the i^{th} row of $\mathcal{G}$ with circular span $[a,b]$.

The following lemma is proved in [KV02].

Lemma 4.13 *If any of the elementary trellises in the trellis product for a tail-biting trellis T is mergeable, then T is mergeable.*

The following lemma follows easily from the T–BCJR construction.

Lemma 4.14 *Every vector $\boldsymbol{g}$ in the* KV *product matrix is associated with a non-mergeable elementary trellis $T_{\boldsymbol{g}}$, via a* BCJR *labeling of $\langle \boldsymbol{g} \rangle$ defined in Definition 4.12, and this trellis is isomorphic to the elementary trellis (conventional or tail-biting), defined by the vector along with its associated span.*

Lemma 4.15 *If $\boldsymbol{c} = (\boldsymbol{c}_1, \ldots, \boldsymbol{c}_n)$ and $\boldsymbol{c}' = (\boldsymbol{c}'_1, \boldsymbol{c}'_2, \ldots, \boldsymbol{c}'_n)$ are two codewords that lead to distinct vertices at level i in a non-mergeable* KV *trellis, then $\boldsymbol{c}$ and $\boldsymbol{c}'$ must induce distinct* T–BCJR *labels (as given in Definition 4.12) at level i.*

Proof: This follows from the fact that if $\boldsymbol{c}$ and $\boldsymbol{c}$ induce the same T–BCJR label (as given in Definition 4.12), then merging the corresponding vertices in the KV trellis T will not change $\mathcal{C}(T)$, thus implying that T is mergeable. ■

Theorem 4.16 *Let $\mathcal{G}$ be a* KV *product matrix for a non-mergeable linear trellis T representing an $(n,k)_q$ linear block code $\mathcal{C}$. Let the* T–BCJR *trellis T' be defined as in Definition 4.12. Then T is isomorphic to T'.*

Proof: Assume that $T = (V, E, \mathbb{F}_q)$ is a non-mergeable linear KV trellis. Let each elementary trellis of T be labeled according to the labeling stated in Lemma 4.14. Let $L_i = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_k\}$ be a set of vertex labels, where $\boldsymbol{v}_j,\ 1 \leq j \leq k$ is a label for a vertex at level i in the j^{th} elementary trellis. Let $\boldsymbol{v} = (\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_k) \in V_i$ be the corresponding node in the product trellis T. For the T-BCJR trellis $T' = \langle \mathcal{G}_{\mathcal{H},\mathcal{D}} \rangle$, the vertex $\boldsymbol{v}' \in V_i'$ computed from the vertex labels in L is given as follows (this follows from the T–BCJR construction).

$$\boldsymbol{v}' = (\boldsymbol{v}_1 + \boldsymbol{v}_2 + \cdots + \boldsymbol{v}_k)$$

Define a map $\phi : V \rightarrow V'$, from the labeled vertices in T to the labeled vertices in T' as follows.

$$(\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_k) \mapsto (\boldsymbol{v}_1 + \boldsymbol{v}_2 + \cdots \boldsymbol{v}_k)$$

We first prove that ϕ is a bijection. Let $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k),\ \boldsymbol{y} = (\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_k) \in V$ be two distinct vertices in V. Since the trellis T is non-mergeable, by Lemma 4.15, $\phi(\boldsymbol{x})$ and $\phi(\boldsymbol{y})$ must be distinct. Therefore ϕ is injective. For every $\boldsymbol{x}' = (\boldsymbol{x}_1 + \boldsymbol{x}_2 + \cdots + \boldsymbol{x}_k) \in V'$ in the T–BCJR trellis T', there exists a vertex $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k) = \phi^{-1}(\boldsymbol{x}')$ in the KV trellis T (by definition of the product trellis). Therefore ϕ is surjective, and hence ϕ is bijective. Next, let $(\boldsymbol{x}_1, \alpha, \boldsymbol{x}_2) \in E$ be an edge in the KV trellis T. Then by definition, the T–BCJR trellis has an edge $(\phi(\boldsymbol{x}_1), \alpha, \phi(\boldsymbol{x}_2)) \in E'$. Therefore ϕ is an isomorphism, and the trellises T and T' are isomorphic. ■

Corollary 4.17 *Every non-mergeable linear tail-biting trellis can be constructed by a* T–BCJR *construction.*

It is worth noting that every biproper linear trellis cannot be constructed by T–BCJR construction. The biproper linear trellis for the $(3,2)_2$ code in Figure 4.3 is one such example.

Note that this trellis is computable by the KV product construction with the product matrix defined as

$$\mathcal{G} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \begin{matrix} [1,3] \\ [2,1] \end{matrix}$$

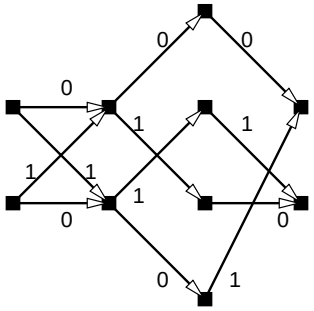


Figure 4.3: A trellis for the $(3,2)_2$ code not computable by a T–BCJR construction.

The following is an example of a T–BCJR trellis which is biproper but not one-to-one.

Example 4.18 *Consider a $(3,2)_2$ code $\mathcal{C}$ with generator matrix $\mathcal{G}$, parity check matrix $\mathcal{H}$ and displacement matrix $\mathcal{D}$ defined as follows.*

$$\mathcal{G} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \qquad \mathcal{H} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \qquad \mathcal{D} = \begin{bmatrix} 0 & 1 \end{bmatrix}$$

The T–BCJR trellis shown in Figure 4.4 is non-mergeable but not one-to-one.

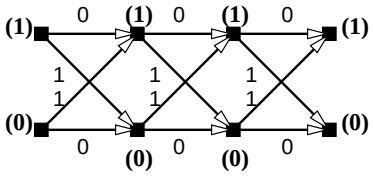


Figure 4.4: A non one-to-one non-mergeable T–BCJR trellis for the $(3,2)_2$ code.

A question that naturally arises here is whether the T–BCJR trellises exactly represent the class of non-mergeable trellises. Below is an example to show they do not.

Example 4.19 *Consider the self dual $(4,2)_2$ linear code specified by*

$$\mathcal{H} = \mathcal{G} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad \mathcal{D} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

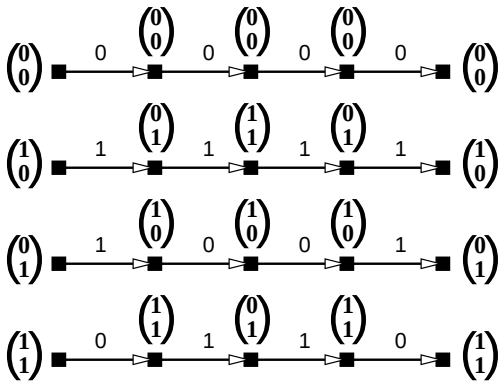


Figure 4.5: A mergeable T–BCJR trellis for the $(4,2)_2$ code.

The T–BCJR *trellis (for the above choice of $\mathcal{D}$) for $\mathcal{C}$ is shown in Figure 4.5 and is mergeable. It is easy to see that to see that this trellis corresponds to a coset decomposition of the code with respect to the subcode $\mathcal{C}_0 = \{0000\}$.*

Therefore, we have the following lemma.

Lemma 4.20 *The class of trellises computed by the* T–BCJR *construction lies in between the class of non-mergeable linear trellises and the class of biproper linear trellises.*

4.2 Construction of Minimal Trellises

We will now address the problem of constructing a trellis T for an $(n,k)_q$ code $\mathcal{C}$ such that $s_{\max}(T)$ is minimized for a fixed permutation of the code. We define an $s_{\max}$-minimal trellis as follows.

Definition 4.21 *A trellis T is said to be $s_{\max}$-*minimal*, if and only if for every trellis T', $s_{\max}(T) \leq s_{\max}(T')$.*

Note that the class of $s_{\max}$-minimal trellises subsumes the class of $\prec_{\max}$-minimal trellises given in Definition 3.12. For example, the trellis in Figure 4.4 is $s_{\max}$-minimal but not $\prec_{\max}$-minimal, since it is not a $\prec_{\Theta}$-minimal trellis. To the best of our knowledge the complexity of the problem of computing $s_{\max}$-minimal trellises is not yet known to be polynomially bounded. As discussed in Section 3.3.2, the feasible space of linear trellises has cardinality equal to $\mathcal{O}(q^{k^2}n^k)$, and this has been reduced to $\mathcal{O}(n^k)$ by Köetter and Vardy [KV03].

Observe that the $s_{\max}$-minimality problem for an $(n,k)_q$ code $\mathcal{C}$ with parity check matrix $\mathcal{H}$ and generator matrix $\mathcal{G}$ may be stated as follows.

Problem: Find a displacement matrix $\mathcal{D}$ that minimizes $s_{\max}(T)$ for a T–BCJR trellis $T = \langle \mathcal{G}_{\mathcal{H},\mathcal{D}} \rangle$ representing $\mathcal{C}$.

$$\mu(\mathcal{C}) \stackrel{\text{def}}{=} \min_{\mathcal{D} \in \mathbb{F}_q^{(n-k)\times k}} s_{\max}(\langle \mathcal{G}_{\mathcal{H},\mathcal{D}} \rangle) \tag{4.6}$$

A naive scheme would involve an $\mathcal{O}(q^{k^2})$ search over the space of all $\mathcal{D}$ matrices. *It is interesting to note that the product of this search complexity and the search complexity from the* KV *characteristic matrix is equal to the total number of linear trellises* $\mathcal{O}(q^{k^2} n^k)$. A better approach is a dynamic algorithm which begins with the generator matrix in trellis-oriented form (see Definition 2.21), and computes a $s_{\max}$-minimal trellis from this matrix in time $\mathcal{O}(n^k)$. While this is a non-polynomial time algorithm, it illustrates the difference between the KV specification and the T–BCJR specification of the minimal trellis.

Let $\mathcal{C}$ be an $(n,k)_q$ code with generator matrix Ψ in trellis-oriented form (TOF), and arbitrary parity check matrix $\mathcal{H}$. Recall that the matrix Ψ is said to be in *trellis-oriented form* if no two rows in Ψ have linear spans that start in the same position or end in the same position. It is well known that the computation of Ψ can be performed in $\mathcal{O}(k^2)$ steps [KS95]. Given Ψ, we have to find a displacement matrix $\mathcal{D}$ which yields a minimal trellis. This is illustrated by the following example.

Example 4.22 *For the* $(7,4)_2$ *Hamming code with generator matrix and parity check matrix as defined in Example 4.11,*

$$\Psi = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

and $\langle \Psi_{\mathcal{H},\mathcal{D}} \rangle$ *with* $\mathcal{D} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$ *is an* $s_{\max}$*-minimal trellis, and is shown in Figure 4.2.*

We will now describe our dynamic algorithm for computing an $s_{\max}$-minimal tail-biting trellis. This involves computing in time $\mathcal{O}(n^k)$ the displacement matrix $\mathcal{D}$ for a $s_{\max}$-minimal trellis from the TOF matrix Ψ for $\mathcal{C}$.

Let $part(\Psi)$ denote the set of all two-partitions of the matrix Ψ. Given a partition $\left[\begin{array}{c} \Psi_l \\ \hline \Psi_c \end{array}\right] \in part(\Psi)$, the algorithm Compute–Displacement computes a set of at most n displacement vectors for every $\boldsymbol{y} \in \Psi_c$. Denote by $\boldsymbol{x}_{[i,j]}$ the projection of $\boldsymbol{x} \in \mathbb{F}_q^n$ onto its components in the closed interval $[i,j]$. Given $A \in F_q^{m \times n}$, let $A_{[i,j]}$ denote the *projected* matrix whose rows are the rows of A projected onto components in $[i,j]$. The algorithm essentially finds, given a partition of the matrix Ψ, for each basis vector $\boldsymbol{y}$ in the partition Ψ_c and every time index $i \in \{0, 2, \ldots, n-1\}$, the longest interval of consecutive time indices (starting at time index i) where the state-complexity of the trellis representing the subcode $\langle \left[\begin{array}{c} \mathcal{G}_l \\ \hline \boldsymbol{y} \end{array}\right] \rangle$ is the same as the state-complexity of the trellis representing the subcode $\langle \mathcal{G}_l \rangle$.

Algorithm Compute–Displacement

Input: The matrices $\mathcal{H}$ and $\Psi = \left[\begin{array}{c} \Psi_l \\ \hline \Psi_c \end{array}\right] \in part(\Psi)$ for an $(n,k)_q$ linear block code $\mathcal{C}$.

Compute: The set of displacement vectors for each $\boldsymbol{y} \in \Psi_c$.

begin

    for each $\boldsymbol{y} \in \Psi_c$

        for each $i \in [n]$

            solve$(\Psi_l, \boldsymbol{y}, i)$

end

function solve

Input: A matrix $A \in \mathbb{F}_q^{m \times n}$, a vector $\boldsymbol{y} \in \mathbb{F}_q^{1 \times n}$ and an integer $s \in [n]$.

Compute: A displacement vector $\boldsymbol{d_y}$ for $\boldsymbol{y}$.

begin

Find the maximum t such that the system of equations given by

$$(\alpha_1, \alpha_2, \ldots, \alpha_m) A_{[s,t]} = \boldsymbol{y}_{[s,t]},\ \alpha_i \in \mathbb{F}_q,\ i \in [m]$$

has a solution. Let $\boldsymbol{u} \in \mathbb{F}_q^m$ be one such solution.

Let $\boldsymbol{z} = (z_1, \ldots, z_n) = \boldsymbol{u}A - \boldsymbol{y}$ and set $\boldsymbol{d_y} = \sum_{i=t}^{n} z_i \boldsymbol{h}_i$.

end

Lemma 4.23 *The function* solve *has time complexity* $\mathcal{O}(n^4)$.

Proof: For each $t \in [n]$, solving the system of equations as described in the function solve involves a computational effort of $\mathcal{O}(n^3)$. Therefore, finding the maximum such t takes $\mathcal{O}(n^4)$ steps. ■

Lemma 4.24 *The algorithm* Compute–Displacement *has time complexity* $\mathcal{O}(n^6)$.

Proof: For each $\boldsymbol{y} \in \Psi_c$, the algorithm Compute–Displacement invokes the function solve an $\mathcal{O}(n)$ number of times. Therefore from the fact that $|\Psi_c| \leq n$ and Lemma 4.23, it follows that the algorithm Compute–Displacement has time complexity $\mathcal{O}(n^6)$. ■

Define an optimal partition of $\Psi \in part(\Psi)$ to be a partition that results in an $s_{\max}$-minimal trellis $\langle \Psi_{\mathcal{H},\mathcal{D}} \rangle$, for some $\mathcal{D} = [\mathbf{0}\ \mathcal{D}_c]$, $\mathcal{D}_c \in \mathbb{F}_q^{(n-k)\times|\Psi_c|}$.

Lemma 4.25 *Given an optimal partition of* $\Psi = \left[\begin{array}{c} \Psi_l \\ \hline \Psi_c \end{array}\right] \in part(\Psi)$*, there exists a displacement matrix* $\mathcal{D}$ *obtained from the displacements computed by* Compute–Displacement *such that* $\langle \Psi_{\mathcal{H},\mathcal{D}} \rangle$ *specifies an* $s_{\max}$*-minimal trellis for* $\mathcal{C}$*. Furthermore, this can be done in time* $\mathcal{O}(n^{|\Psi_c|})$.

Proof: Let $\mathcal{G} = \left[\begin{array}{c} \mathcal{G}_l \\ \hline \mathcal{G}_c \end{array}\right]$ be the KV product matrix for an $s_{\max}$-minimal trellis T. From Definition 4.12, we may also represent T as $\langle \mathcal{G}_{\mathcal{H},\mathcal{D}} \rangle$, for some $\mathcal{D} \in \mathbb{F}_q^{(n-k)\times k}$. Since we are considering an optimal partition of Ψ, we may assume that $\langle \mathcal{G}_l \rangle = \langle \Psi_l \rangle$ (recall that Ψ is in trellis-oriented form). Let $\boldsymbol{y}$ be an arbitrary vector in Ψ_c. We will now argue that there exists a correct displacement vector out of the k possible displacements for $\boldsymbol{y}$. Let $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{l}$, where $\boldsymbol{l} \in \langle \mathcal{G}_l \rangle$ and $\boldsymbol{c} \in \langle \mathcal{G}_c \rangle$. Let $\boldsymbol{c}$ have a zero-run of $[i, j]$ (from Lemma 3.23, any vector in

$\langle \mathcal{G}_c \rangle$ must have a nonempty zero-run). Since T is an $s_{\max}$-minimal trellis, $[i,j]$ is the longest run of zeros starting at time index i. Therefore the function solve will find a solution to the system of equations

$$(\alpha_1, \alpha_2, \ldots, \alpha_m)\Psi_{l[i,j]} = \boldsymbol{y}_{[i,j]},\ \alpha_i \in \mathbb{F}_q,\ i \in [|\Psi_l|]$$

In general, there could be many solutions to these equations. Let $\boldsymbol{u}$ be an arbitrarily chosen solution and let $\boldsymbol{z} = \boldsymbol{u}\Psi_l - \boldsymbol{y}$. Note that $\boldsymbol{z}$ has a zero-run in the interval $[i,j]$ and therefore the same circular span as $\boldsymbol{c}$. Let $\mathcal{G}' = \mathcal{G} \setminus \{\boldsymbol{c}\} \cup \{\boldsymbol{z}\}$ and let $T' = \langle \mathcal{G}'_{\mathcal{H},\mathcal{D}} \rangle$. Since $\boldsymbol{z} = \boldsymbol{u}\Psi_l - \boldsymbol{c} - \boldsymbol{l} = \boldsymbol{l}' - \boldsymbol{c}$, where $\boldsymbol{l}' \in \langle \mathcal{G}_l \rangle$, we have $\mathrm{rank}(\mathcal{G}) = \mathrm{rank}(\mathcal{G}')$, and therefore $s_{\max}(T) = s_{\max}(T')$. Therefore $\boldsymbol{d_y}$ as defined by Compute–Displacement is a correct choice for obtaining an $s_{\max}$-minimal trellis. Since for each vector $\boldsymbol{y} \in \Psi_c$, there exists a correct choice for $\boldsymbol{d_y}$, the lemma follows. ■

Theorem 4.26 *Given an $(n,k)_q$ code $\mathcal{C}$, an $s_{\max}$-minimal trellis representing $\mathcal{C}$ can be computed in time $\mathcal{O}(n^k)$.*

Proof: Given an optimal partition $\left[\frac{\Psi_l}{\Psi_c}\right]$, we know from Lemmas 4.24 and 4.25 that an $s_{\max}$-minimal trellis can be computed in time $\mathcal{O}(n^i)$, where $i = |\Psi_c|$. The number of partitions such that $|\Psi_c| = i$ is equal to $\binom{k}{i}$. Therefore, computing an $s_{\max}$-minimal trellis can be performed in time bounded above by

$$\sum_{i=0}^{k} \binom{k}{i} n^i = \mathcal{O}(n^k)$$

■

We will conclude this section by presenting a new formulation of the minimal tail-biting trellis problem that based on the T-BCJR specification. Let $\mathcal{C}$ be an $(n,k)_q$ code with generator matrix $\mathcal{G}$ and parity check matrix $\mathcal{H}$. Then from the BCJR construction, we know that a conventional trellis for $\mathcal{C}$ is completely specified by the sequence

$$\Phi_{\mathcal{C}} = \{\mathcal{S}_i\}_{i=0}^{n},\ \text{where } \mathcal{S}_i = \mathcal{H}_i \mathcal{G}_i^t \tag{4.7}$$

where we recall that the column-space of each $\mathcal{S}_i$ represents the set of states at level i of the trellis. From the results stated in Section 4.1, it follows that every non-mergeable linear tail-biting trellis for $\mathcal{C}$ can be specified as

$$\Phi_{\mathcal{C}} + \mathcal{D} \stackrel{\text{def}}{=} \{\mathcal{S}_i + \mathcal{D}\}_{i=0}^{n}, \text{ where } \mathcal{D} \in \mathbb{F}_q^{(n-k)\times k} \tag{4.8}$$

Define the rank of the sequence $\Phi_{\mathcal{C}} + \mathcal{D}$ as follows.

$$\text{rank}(\Phi_{\mathcal{C}} + \mathcal{D}) \stackrel{\text{def}}{=} \max_{0\le i\le n} \text{rank}(\mathcal{S}_i + \mathcal{D}). \tag{4.9}$$

The $s_{\max}$-minimal tail-biting trellis computation problem may thus be formulated as follows.

Problem: Find a matrix $\mathcal{D}$ such that $\text{rank}(\Phi_{\mathcal{C}} + \mathcal{D})$ is minimized.

$$\mu(\mathcal{C}) \stackrel{\text{def}}{=} \min_{\mathcal{D}\in\mathbb{F}_q^{(n-k)\times k}} \text{rank}(\Phi_{\mathcal{C}} + \mathcal{D}) \tag{4.10}$$

Chapter 5

The Tail-Biting Forney Trellis

In this chapter we generalize the Forney construction for conventional trellises described in Section 2.2.3 to obtain tail-biting trellises. Specifically, we show that there is a generalization of "pasts" and "futures" which enables the construction of a linear tail-biting trellis for a linear block code from a coset decomposition of the code, with respect to a subcode of the code.

5.1 Introduction

As mentioned in Section 3.4, tail-biting trellises for block codes arise from a coset decomposition of the code with respect to a subcode [DSDR00, LS00, SDDR03, SB00]. The subtrellises for the codes associated with the cosets are actually structural copies of the trellis for the subcode and differ only in the labels labeling the edges. Recall that the Forney conventional trellis for a linear block code $\mathcal{C}$ is constructed by identifying the vertices of the trellis at a given time index $i \in \mathcal{I}$ with cosets of $\mathcal{C}$ modulo a certain subcode $\mathcal{C}_0$ of $\mathcal{C}$. This subcode $\mathcal{C}_0$ is computed by identifying *past* and *future* projections (see Section 2.2.3) of the code $\mathcal{C}$, at each time index.

We will now show that there is a natural generalization for pasts and futures for tail-biting trellises, depending on the coset decomposition which uniquely specifies the tail-biting trellis.

5.2 The Tail-Biting Forney Trellis

Our adaptation of Forney's coset construction computes a trellis (henceforth referred to as the T–Forney trellis) that uses a coset decomposition of the code and the associated merging interval (see Section 3.4) for each coset. Let $\mathcal{C}_0$ be a linear subcode of an $(n,k)_q$ code $\mathcal{C}$. Denote by $\{\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_m\}$, the coset decomposition of $\mathcal{C}$ over $\mathcal{C}_0$, where each coset leader $\boldsymbol{v}_i$ has merging interval $[a_i, b_i]$. For all vectors in coset $\mathcal{C}_l$ with coset leader $\boldsymbol{v}_l$, having merging interval $[a_l, b_l]$ with the all-zero codeword, the past (perhaps more appropriately, the *circular* past) at i begins at time index b_l, wraps around mod n and ends at i. Similarly, a *future* of a vector in $\mathcal{C}_l$ at time index i begins at $i+1$ and ends at a_l. Define maps $\pi_{il},\ 0 \le i \le n,\ 0 \le l \le m$, as follows.

$$\pi_{il} : \mathcal{C}_l \to \pi_{il}(\mathcal{C}_l), \text{ defined by } \boldsymbol{c} = (c_1, \ldots, c_n) \mapsto c_{b_l}\boldsymbol{h}_{b_l} + \cdots + c_n\boldsymbol{h}_n + c_1\boldsymbol{h}_1 + \cdots c_i\boldsymbol{h}_i \quad (5.1)$$

Example 5.1 *Let $\mathcal{C}$ be a $(7,4)_2$ Hamming code specified by*

$$\mathcal{H} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad \mathcal{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

Let the linear subcode $\mathcal{C}_0 = \{0000000, 1000110, 0010111, 1010001\}$. Then the rest of the cosets in $\mathcal{C}/\mathcal{C}_0$ are

$$\begin{aligned} \mathcal{C}_1 &= \{0100011, 1100101, 0110100, 1110010\} \\ \mathcal{C}_2 &= \{0111001, 1111111, 0101110, 1101000\} \\ \mathcal{C}_3 &= \{0011010, 1011100, 0001101, 1001011\} \end{aligned}$$

with coset leaders and merging intervals defined as follows.

$$\begin{aligned} \boldsymbol{v}_1 &= 0100011 \quad [3,5] \\ \boldsymbol{v}_2 &= 0111001 \quad [5,6] \\ \boldsymbol{v}_3 &= 0011010 \quad [6,6] \end{aligned}$$

For $i = 3$ and $l = 2$, the map π_{il} is given by

$$\pi_{32}(0111001) = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

$$\pi_{32}(1111111) = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

$$\pi_{32}(0101110) = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

$$\pi_{32}(1101000) = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

For every time index i, we define the past $\mathcal{P}_i(\mathcal{C}_l)$ and future $\mathcal{F}_i(\mathcal{C}_l)$ for vectors in coset $\mathcal{C}_l$ as follows.

$$\mathcal{P}_i(\mathcal{C}_l) = \{(c_1, \ldots, c_i) : \boldsymbol{c} = (c_1, \ldots, c_i, c_{i+1}, \ldots, c_n) \in \mathcal{C}_l,\ \pi_{il}(\boldsymbol{c}) = \boldsymbol{0}\} \tag{5.2}$$

$$\mathcal{F}_i(\mathcal{C}_l) = \{(c_{i+1}, \ldots, c_n) : \boldsymbol{c} = (c_1, \ldots, c_i, c_{i+1}, \ldots, c_n) \in \mathcal{C}_l,\ \pi_{il}(\boldsymbol{c}) = \boldsymbol{0}\} \tag{5.3}$$

Lemma 5.2 *The set $\bigcup_{0 \le l \le m}(\mathcal{P}_i(\mathcal{C}_l) \times \mathcal{F}_i(\mathcal{C}_l))$ is a linear subcode of $\mathcal{C}$.*

Proof: Fix a time index i. Let $\boldsymbol{c}_1 \in \mathcal{P}_i(\mathcal{C}_r) \times \mathcal{F}_i(\mathcal{C}_r)$ and $\boldsymbol{c}_2 \in \mathcal{P}_i(\mathcal{C}_s) \times \mathcal{F}_i(\mathcal{C}_s)$. Let $\boldsymbol{z} = \alpha\boldsymbol{c}_1 + \beta\boldsymbol{c}_2,\ \alpha, \beta \in F_q$, belong to coset $\mathcal{C}_t$. It can easily be verified that $\pi_{it}(\boldsymbol{z}) = \alpha\pi_{ir}(\boldsymbol{c}_1) + \beta\pi_{is}(\boldsymbol{c}_2) = \boldsymbol{0}$, and therefore $\boldsymbol{z} \in \mathcal{P}_i(\mathcal{C}_t) \times \mathcal{F}_i(\mathcal{C}_t)$. ■

The T–Forney trellis $T = (V, E, \mathbb{F}_q)$ for $\mathcal{C}$ is constructed by identifying vertices in V_i with the quotient group corresponding to cosets of $\mathcal{C}$ modulo $\bigcup_{0 \le l \le m}(\mathcal{P}_i(\mathcal{C}_l) \times \mathcal{F}_i(\mathcal{C}_l))$, that is,

$$V_i = \mathcal{C} / \bigcup_{0 \le l \le m} (\mathcal{P}_i(\mathcal{C}_l) \times \mathcal{F}_i(\mathcal{C}_l)) \tag{5.4}$$

There is an edge from $e \in E_i$ labeled c_i from a vertex $\boldsymbol{u} \in V_{i-1}$ to a vertex $\boldsymbol{v} \in V_i$, if and only if there exists a codeword $\boldsymbol{c} = (c_1, \ldots, c_n) \in \mathcal{C}$ such that $\boldsymbol{c} \in \boldsymbol{u} \cap \boldsymbol{v}$.

Example 5.3 *Let $\mathcal{C}$ be a self dual $(4, 2)_2$ code with parity check matrix $\mathcal{H}$ and generator*

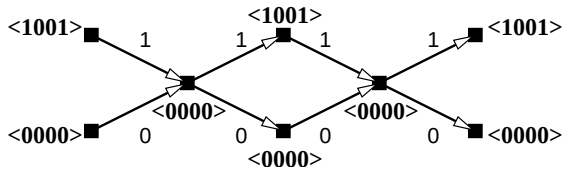


Figure 5.1: A T–Forney trellis for the $(4,2)_2$ code.

matrix $\mathcal{G}$ defined as follows.

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

Let the linear subcode of $\mathcal{C}$ be defined as

$$\mathcal{C}_0 = \{0000, 0110\}$$

Let the coset leader for the coset $\mathcal{C}_1$ be $\boldsymbol{v}_1 = 1001$ (with merging interval $[2,3]$). The subcodes $\bigcup_{0\le l\le 1} \mathcal{P}_i(\mathcal{C}_l) \times \mathcal{F}_i(\mathcal{C}_l)$, $0 \le i \le 3$, are given by

$$\begin{aligned}
\bigcup_{0\le l\le 1} \mathcal{P}_0(\mathcal{C}_l) \times \mathcal{F}_0(\mathcal{C}_l) &= \{0000, 0110\} \\
\bigcup_{0\le l\le 1} \mathcal{P}_1(\mathcal{C}_l) \times \mathcal{F}_1(\mathcal{C}_l) &= \{0000, 0110, 1001, 1111\} \\
\bigcup_{0\le l\le 1} \mathcal{P}_2(\mathcal{C}_l) \times \mathcal{F}_2(\mathcal{C}_l) &= \{0000, 0110\} \\
\bigcup_{0\le l\le 1} \mathcal{P}_3(\mathcal{C}_l) \times \mathcal{F}_3(\mathcal{C}_l) &= \{0000, 0110, 1001, 1111\}
\end{aligned}$$

The T–Forney trellis for $\mathcal{C}$ is shown in Figure 5.1. The vertices in Figure 5.1 are labeled by the coset representatives in V_i, $0 \le i \le 3$. For example, the vertex $\{1001, 1111\} \in V_0$ is labeled by $\langle 1001 \rangle$. We observe that since the coset with respect to which the decomposition is computed is the whole code at time indices 1 and 3, there is a single state at those time indices.

Example 5.4 *Let $\mathcal{C}$ be the $(7,4)_2$ Hamming code from Example 5.1. The subcodes*

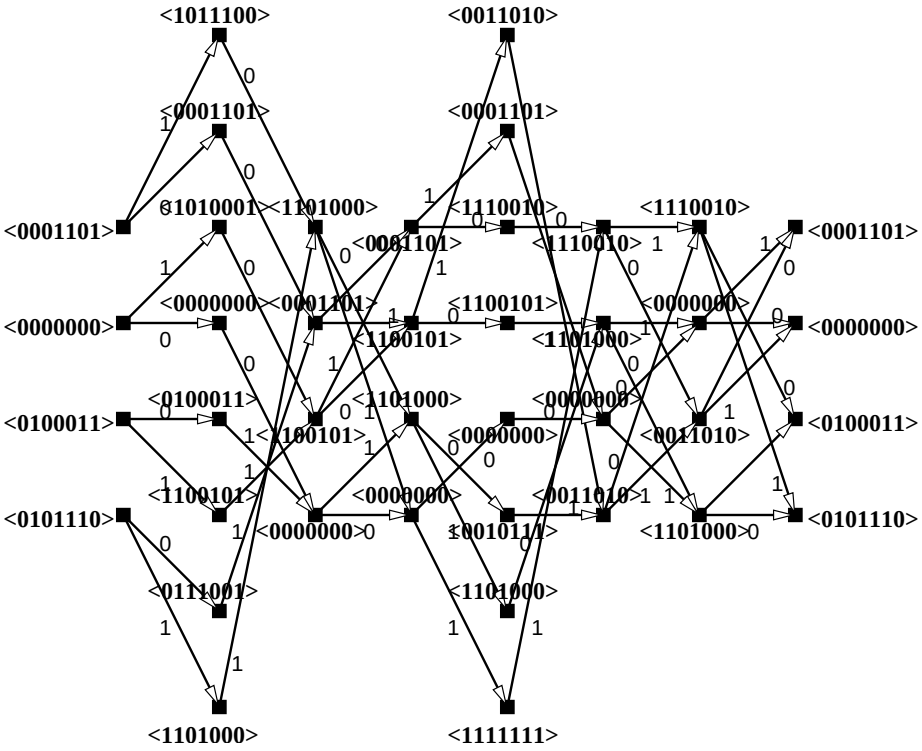


Figure 5.2: A T–Forney trellis for the $(7,4)_2$ Hamming code.

$\bigcup_{0\le l\le 2} \mathcal{P}_i(\mathcal{C}_l) \times \mathcal{F}_i(\mathcal{C}_l)$, $0 \le i \le 6$, *are given by:*

$$\begin{aligned}
\bigcup_{0\le l\le 2} \mathcal{P}_0(\mathcal{C}_l) \times \mathcal{F}_0(\mathcal{C}_l) &= \{0000000, 0010111, 1010001, 1000110\} \\
\bigcup_{0\le l\le 2} \mathcal{P}_1(\mathcal{C}_l) \times \mathcal{F}_1(\mathcal{C}_l) &= \{0000000, 0010111\} \\
\bigcup_{0\le l\le 2} \mathcal{P}_2(\mathcal{C}_l) \times \mathcal{F}_2(\mathcal{C}_l) &= \{0000000, 0010111, 0110100, 0100011\} \\
\bigcup_{0\le l\le 2} \mathcal{P}_3(\mathcal{C}_l) \times \mathcal{F}_3(\mathcal{C}_l) &= \{0000000, 1111111, 1011100, 0100011\} \\
\bigcup_{0\le l\le 2} \mathcal{P}_4(\mathcal{C}_l) \times \mathcal{F}_4(\mathcal{C}_l) &= \{0000000, 0100011\} \\
\bigcup_{0\le l\le 2} \mathcal{P}_5(\mathcal{C}_l) \times \mathcal{F}_5(\mathcal{C}_l) &= \{0000000, 0001101, 0100011, 0101110\} \\
\bigcup_{0\le l\le 2} \mathcal{P}_6(\mathcal{C}_l) \times \mathcal{F}_6(\mathcal{C}_l) &= \{0000000, 0001101, 1001011, 1000110\}
\end{aligned}$$

The T–Forney *trellis for* $\mathcal{C}$ *is shown in Figure 5.2. The vertices in Figure 5.2 are labeled by the coset representatives in* V_i, $0 \le i \le 6$. *For example, the vertex* $\{1101000, 1111111\} \in V_1$

is labeled by $\langle 1101000 \rangle$.

The following theorem relates the T–BCJR and T–Forney trellises.

Theorem 5.5 *Given an* $(n,k)_q$ *code* $\mathcal{C}$ *with generator matrix* $\mathcal{G}$*, parity check matrix* $\mathcal{H}$ *and displacement matrix* $\mathcal{D}$*, the* T–BCJR *and* T–Forney *trellises are isomorphic to each other.*

Proof: Given a coset leader $\boldsymbol{v} = (v_1, v_2, \ldots, v_n)$ with merging interval $[a,b]$, define the displacement vector for $\boldsymbol{v}$ as follows.

$$\boldsymbol{d_v} = \sum_{i=0}^{a} v_i \boldsymbol{h}_i \tag{5.5}$$

Therefore $\mathcal{H}$,$\mathcal{G}$ and $\mathcal{D}$ also specify a T–Forney trellis. From Lemma 4.9, we also know that the state-cardinality of the T–BCJR trellis at level i is given by the column-space cardinality of $M_i = [\mathcal{H}_i {\mathcal{G}_i}^t + \mathcal{D}]$. Define a family of maps π_i, $0 \leq i \leq n$, as follows.

$$\pi_i \stackrel{\text{def}}{=} \bigcup_{0 \leq l \leq m} \pi_{il} \tag{5.6}$$

where the maps π_{il}, $0 \leq l \leq m$, are defined in (5.1). We also have $\forall i,\ 0 \leq i \leq n$,

$$\text{Im } \pi_i = \text{column} - \text{space}(M_i) \tag{5.7}$$

The kernel of $\pi_i,\ 0 \leq i \leq n$, also turns out to be

$$\text{Ker } \pi_i = \left\{ \boldsymbol{c} : \boldsymbol{c} \in \bigcup_{0 \leq l \leq m} (\mathcal{P}_i(\mathcal{C}_l) \times \mathcal{F}_i(\mathcal{C}_l)) \right\} \tag{5.8}$$

We therefore have

$$\text{Im } \pi_i \cong \mathcal{C} / \bigcup_{0 \leq l \leq m} (\mathcal{P}_i(\mathcal{C}_l) \times \mathcal{F}_i(\mathcal{C}_l)) \tag{5.9}$$

Since the vertex class V_i at level i of the T–Forney trellis is identified with $\mathcal{C} / \bigcup_{0 \leq l \leq m} \mathcal{P}_i(\mathcal{C}_l) \times \mathcal{F}_i(\mathcal{C}_l)$, the T–Forney and T–BCJR trellises have the same number of vertices at all time indices. From the previous statements, it can also be easily verified that the T–Forney condition for edge placement between vertices translates to the T–BCJR edge placement condition. Therefore, the T–BCJR and T–Forney trellises are isomorphic to each other. ■

Chapter 6

The Tail-Biting Dual Trellis

As seen in Section 2.2.1, there are a number of interesting connections between the minimal conventional trellis for a linear code and the minimal conventional trellis for its dual code. It is an interesting fact that algebraic and combinatorial duality are related, and this was first established by Forney [For88]. Kötter and Vardy [KV03] have defined a special product operation called the *intersection product* to construct a linear tail-biting dual trellis directly from a generator matrix for the primal code. This results in a linear tail-biting trellis $T^{\perp}$ for the dual code that has the same SCP as the primal trellis T, only if T is $\prec_{\Theta}$-minimal. Their construction is somewhat indirect and involves a new trellis product invented solely for the purpose of constructing dual trellises. We give a much simpler and direct construction. Our construction that is based on the T–BCJR specification of linear tail-biting trellises extends the Kötter-Vardy result to a larger class of trellises. We begin this chapter by describing the intersection product construction for dual trellises, followed by our dual T–BCJR construction.

6.1 The Intersection Product

Kötter and Vardy [KV03] define a trellis product operator called the intersection product, and use this operation to construct dual trellises directly from the generator matrix for the primal code. In particular, they prove that given any $\prec_{\Theta}$-minimal trellis T for a code $\mathcal{C}$, there exists a corresponding $\prec_{\Theta}$-minimal dual trellis $T^{\perp}$ for the dual code $\mathcal{C}^{\perp}$, such that the SCP of T is identical to the SCP of $T^{\perp}$.

Definition 6.1 *Let $T_1 = (V', E', \mathbb{F}_q)$ and $T_2 = (V'', E'', \mathbb{F}_q)$ be two trellises (either conventional or tail-biting) of depth n. Then the* intersection product *trellis $T' \odot T''$ is the trellis $T = (V, E, \mathbb{F}_q)$ of depth n, whose vertex and edge classes are Cartesian products defined as follows.*

$$V_i \stackrel{\text{def}}{=} \{(v', v'') : v' \in V_i',\ v'' \in V_i''\} \tag{6.1}$$

$$E_i \stackrel{\text{def}}{=} \{(v'_{i-1}, v''_{i-1}), a, (v'_i, v''_i) : (v'_{i-1}, a, v'_i) \in E'_i, (v''_{i-1}, a, v''_i) \in E''_i\} \tag{6.2}$$

Then T represents the code $\mathcal{C}$ defined as

$$\mathcal{C} = \mathcal{C}(T_1) \cap \mathcal{C}(T_2) \tag{6.3}$$

The construction of Kötter and Vardy is based on the following lemma.

Lemma 6.2 *Let $\mathcal{C}$ be an $(n, k)_q$ code defined by the set of generators $\{\boldsymbol{g}_1, \boldsymbol{g}_2, \ldots, \boldsymbol{g}_k\}$. Then*

$$\mathcal{C}^\perp = \langle \boldsymbol{g}_1 \rangle^\perp \cap \langle \boldsymbol{g}_2 \rangle^\perp \cdots \langle \boldsymbol{g}_k \rangle^\perp \tag{6.4}$$

We will now introduce the *elementary dual trellis* $T_{\boldsymbol{g}}^\perp$ as defined in [KV03]. Given an arbitrary vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ over the field $\mathbb{F}_q = \{0, \beta_1, \ldots, \beta_{q-1}\}$, along with its span $[a, b]$, the corresponding elementary trellis $T_{\boldsymbol{x}}$ is completely determined by $\boldsymbol{x}$ and $[a, b]$, and represents the one-dimensional code $\langle \boldsymbol{x} \rangle$ over $\mathbb{F}_q$ (see Section 3.2). Analogously, the elementary dual trellis $T_{\boldsymbol{x}}^\perp$ is completely determined by $\boldsymbol{x}$ and $[a, b]$, and represents the $(n-1)$-dimensional code $\langle \boldsymbol{x} \rangle^\perp$ over $\mathbb{F}_q$. The trellis $T_{\boldsymbol{x}}^\perp$ has q vertices labeled $0, \beta, \ldots, \beta_{q-1}$ at those positions that belong to $(a, b]$, and a single vertex labeled 0 at other positions. For ease of exposition, assume that $x_a \neq 0$ and $x_b \neq 0$. We will describe the edges of $T_{\boldsymbol{x}}^\perp$ for the following cases.

(i) $j \notin [a, b]$. There are q edges labeled by the elements of $\mathbb{F}_q$ between $v \in V_j$ and $v' \in V_{j+1}$.

(ii) $j = a$. The vertex $v \in V_j$ is connected by an edge labeled $\frac{\beta}{x_j}$ to each of the q vertices $v' \in V_{j+1}$, where β is the label of the vertex v'.

(iii) $j \in (a, b)$ and $x_j = 0$. The vertices $v \in V_j$ and $v' \in V_{j+1}$ are connected if and only

if they have the same label $\beta \in \mathbb{F}_q$ – in which case there are q edges labeled by the elements of $\mathbb{F}_q$ from v to v'.

(iv) $j \in (a, b)$ and $x_j \neq 0$. Each vertex in $v \in V_j$ is connected by a single edge labeled by $\frac{\beta' - \beta}{x_j}$ to each vertex in $v' \in V_{j+1}$, where β and β' are labels of v and v' respectively.

(v) $j = b$. The vertex $v \in V_j$ is connected by an edge labeled $\frac{-\beta}{x_j}$ to each of the q vertices $v' \in V_{j+1}$, where β is the label of the vertex v'.

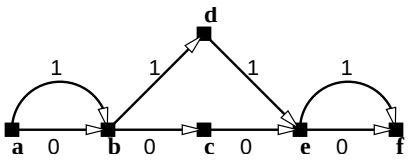


Figure 6.1: An elementary dual trellis for the vector (0110) with span $[2, 3]$.

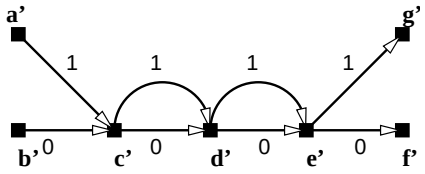


Figure 6.2: An elementary dual trellis for the vector (1001) with span $[4, 1]$.

Example 6.3 *Two elementary dual trellises for the vectors* $\boldsymbol{g}_1 = (0110) \in \mathbb{F}_2^4$ *(with span* $[2, 3]$*) and* $\boldsymbol{g}_2 = (1001) \in \mathbb{F}_2^4$ *(with span* $[4, 1]$*) are shown in Figures 6.1 and 6.2 respectively.*

The following lemma describes the intersection product construction that computes dual linear tail-biting trellises from the specification of their primal counterparts.

Lemma 6.4 *Let* $\mathcal{C}$ *be an* $(n, k)_q$ *code defined by a set of generators* $\{\boldsymbol{g}_1, \boldsymbol{g}_2, \ldots, \boldsymbol{g}_k\}$*. Let* T *be a linear trellis computed as* $T = T_{\boldsymbol{g}_1} \times T_{\boldsymbol{x}_2} \times \cdots \times T_{\boldsymbol{x}_k}$*. Then the trellis*

$$T^{\perp} = T^{\perp}_{\boldsymbol{g}_1} \odot T^{\perp}_{\boldsymbol{g}_2} \odot \cdots \odot T^{\perp}_{\boldsymbol{g}_k} \tag{6.5}$$

is a linear trellis that represents the dual code $\mathcal{C}^{\perp}$ *of* $\mathcal{C}$*, and has an* SCP *identical to the* SCP *of* T*.*

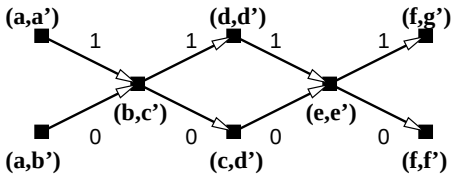


Figure 6.3: A dual trellis for the $(4,2)_2$ code computed by an intersection product.

Example 6.5 *For the* $(4,2)_2$ *code specified by*

$$\mathcal{G} = \left[\begin{array}{cccc} 0 & 1 & 1 & 0 \\ \hline 1 & 0 & 0 & 1 \end{array}\right] \begin{array}{c} [2,3] \\ [4,1] \end{array}$$

The elementary trellises for the rows of $\mathcal{G}$ *are shown in Figures 6.1 and 6.2, and the dual trellis computed by the intersection product is shown in Figure 6.3.*

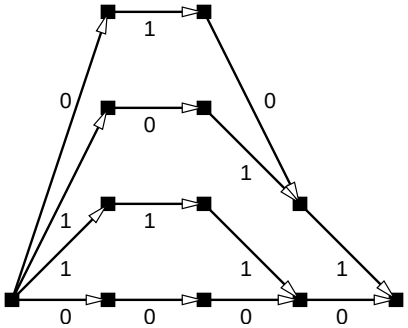


Figure 6.4: A non-minimal trellis for the $(4,2)_2$ code in Example 6.6.

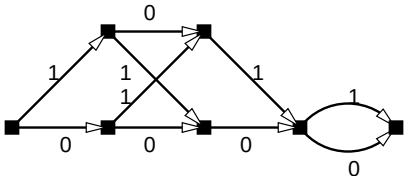


Figure 6.5: An elementary trellis for the code $\langle 1110 \rangle^{\perp}$.

The trellis $T^{\perp}$ defined in Lemma 6.4, is in general not a *reduced* trellis, as illustrated by the following example.

Example 6.6 ([KV03]) *Let* $\mathcal{C}$ *be a* $(4,2)_2$ *code with* KV *product matrix*

$$\mathcal{G} = \left[\begin{array}{cccc} 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{array}\right] \begin{array}{c} [1,3] \\ [1,4] \end{array}$$

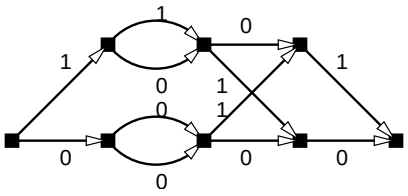


Figure 6.6: An elementary trellis for the code $\langle 1011 \rangle^{\perp}$.

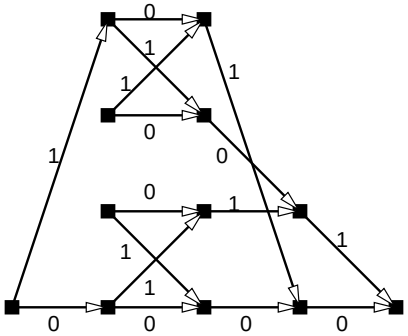


Figure 6.7: A dual trellis for the dual code in Example 6.6 that is not reduced.

The product primal trellis T for $\mathcal{C}$ is shown in Figure 6.4. The elementary trellises for $\langle 1110 \rangle^{\perp}$ and $\langle 1011 \rangle^{\perp}$ are shown in Figures 6.5 and 6.6 respectively. The dual trellis $T^{\perp}$ representing $\mathcal{C}^{\perp}$ computed by the intersection product is shown in Figure 6.7. As illustrated by the figures, the SCP *of $T^{\perp}$ is identical to the* SCP *of T, and $T^{\perp}$ is a not a reduced trellis.*

But this situation does not arise if the original primal trellis T is Θ-minimal [KV03]. Therefore we have the following theorem.

Theorem 6.7 *Let T be a $\prec_{\Theta}$-minimal linear trellis, either conventional or tail-biting, for a linear code $\mathcal{C}$. Then the dual trellis $T^{\perp}$ is a $\prec_{\Theta}$-minimal trellis for $\mathcal{C}^{\perp}$, and the* SCP *of $T^{\perp}$ is identical to the* SCP *of T.*

6.2 The Tail-Biting Dual Trellis

We will now describe our solution to the dual trellis computation problem. Let $\mathcal{G}$, $\mathcal{H}$ and $\mathcal{D}$ respectively, be the generator, parity and displacement matrices for a primal code $\mathcal{C}$. Given a primal trellis specification $T = \langle \mathcal{G}_{\mathcal{H},\mathcal{D}} \rangle$, the dual BCJR construction (T–BCJR$^{\perp}$) computes a biproper linear tail-biting trellis $T^{\perp}$ for the dual code $\mathcal{C}^{\perp}$, with the property that the SCP of $T^{\perp}$ is equal to the SCP of T. In other words, given a *minimal* (under any definition of

minimality) trellis T for the primal code, T–BCJR$^{\perp}$ computes a *minimal* linear tail-biting trellis $T^{\perp}$ for the dual code such that the SCP of T is equal to the SCP of $T^{\perp}$. We define the dual trellis $T^{\perp}$ representing the dual code $\mathcal{C}^{\perp}$ as follows.

Definition 6.8 (T–BCJR$^{\perp}$ Construction) *Let $\mathcal{C}$ be an $(n,k)_q$ code with generator and parity check matrices $\mathcal{G}$ and $\mathcal{H}$ respectively. Let $T = \langle \mathcal{G}_{\mathcal{H},\mathcal{D}} \rangle$ be a biproper linear trellis representing $\mathcal{C}$ for some $\mathcal{D} \in \mathbb{F}_q^{(n-k)\times k}$. Then the dual* BCJR *(T–BCJR$^{\perp}$) trellis $T^{\perp} = \left(V^{\perp}, E^{\perp}, \mathbb{F}_q\right)$ representing the $(n, n-k)_q$ dual code $\mathcal{C}^{\perp}$ is defined as*

$$T^{\perp} = \langle \mathcal{H}_{\mathcal{G},\mathcal{D}^t} \rangle \tag{6.6}$$

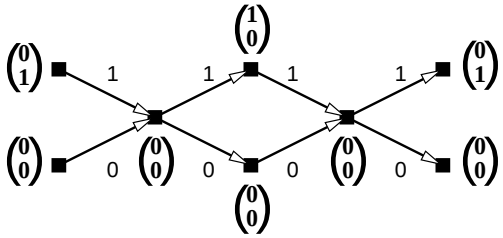


Figure 6.8: A T–BCJR$^{\perp}$ trellis for the $(4,2)_2$ code.

Example 6.9 *For the $(4,2)_2$ code defined in Example 4.10, we have*

$$\mathcal{H}_{\mathcal{G},\mathcal{D}^t} = \begin{bmatrix} \binom{0}{0} & 0 & \binom{0}{0} & 1 & \binom{1}{0} & 1 & \binom{0}{0} & 0 & \binom{0}{0} \\ \binom{0}{1} & 1 & \binom{0}{0} & 0 & \binom{0}{0} & 0 & \binom{0}{0} & 1 & \binom{0}{1} \end{bmatrix}$$

and $\langle \mathcal{H}_{\mathcal{G},\mathcal{D}^t} \rangle$ is the trellis shown in Figure 6.8.

Example 6.10 *For the $(7,4)_2$ Hamming code given in Example 4.11, we have*

$$\mathcal{H}_{\mathcal{G},\mathcal{D}^t} = \begin{bmatrix} \binom{0}{0}{0}{0}{1} & 1 & \binom{0}{1}{0}{0} & 1 & \binom{0}{0}{0}{0} & 0 & \binom{0}{0}{0}{0} & 0 & \binom{0}{0}{0}{0} & 1 & \binom{1}{0}{0}{0} & 0 & \binom{1}{0}{0}{0} & 1 & \binom{0}{0}{0}{1} \\ \binom{0}{0}{0}{0} & 1 & \binom{0}{1}{0}{1} & 1 & \binom{0}{0}{0}{1} & 1 & \binom{0}{0}{1}{0} & 0 & \binom{0}{0}{1}{0} & 0 & \binom{0}{0}{1}{0} & 1 & \binom{0}{0}{0}{0} & 0 & \binom{0}{0}{0}{0} \\ \binom{0}{0}{0}{1} & 0 & \binom{0}{0}{0}{1} & 1 & \binom{0}{1}{0}{1} & 1 & \binom{0}{1}{1}{0} & 1 & \binom{1}{0}{0}{0} & 0 & \binom{1}{0}{0}{0} & 0 & \binom{1}{0}{0}{0} & 1 & \binom{0}{0}{0}{1} \end{bmatrix}$$

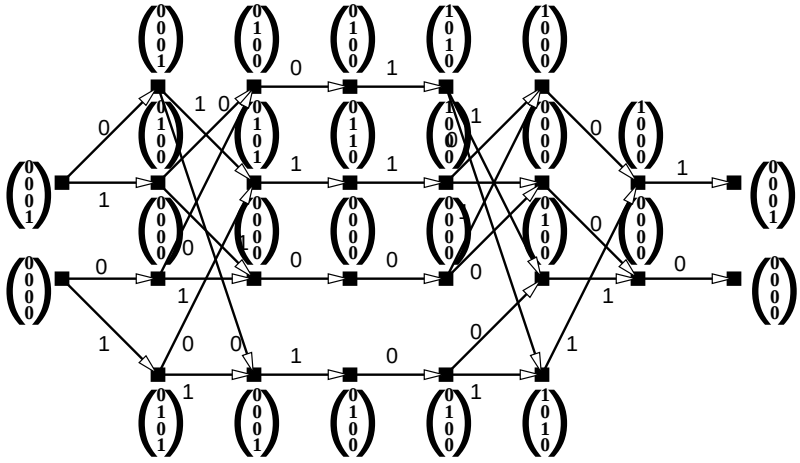


Figure 6.9: A T–BCJR$^\perp$ trellis for the $(7,4)_2$ Hamming code.

and $\langle \mathcal{H}_{\mathcal{G},\mathcal{D}^t} \rangle$ *is the trellis shown in Figure 6.9.*

The following lemma states the properties of the T–BCJR$^\perp$ trellis $T^\perp$.

Lemma 6.11 *The trellis* $T^\perp$ *is a biproper linear trellis that represents* $\mathcal{C}^\perp$*.*

Proof: Since $T^\perp$ is a T–BCJR trellis, the lemma follows from Lemmas 4.7 and 4.8. ■

Let M_i be the matrix defined in (4.5). Recall that

$$M_i = \left[\mathcal{H}_i {\mathcal{G}_i}^t + \mathcal{D}\right]$$

Lemma 6.12 *For all time indices* $i \in \mathcal{I}$*, the state space of* $T^\perp$ *at time index* i *equals the column-space of* M_i^t*.*

Proof: Follows directly from Definition 6.8. ■

Example 6.13 *The* T–BCJR$^\perp$ *trellises for the* $(4,2)_2$ *self dual code and the* $(7,4)_2$ *Hamming code from Examples 6.9 and 6.10, are shown in Figures 6.8 and 6.9 respectively. These trellises have the same* SCP*s as their primal counterparts (Figures 4.1 and 4.2).*

Lemma 6.14 *Let* T *and* $T^\perp$ *be the trellises computed by the* T–BCJR *and* T–BCJR$^\perp$ *constructions respectively. Then for all all time indices* $i \in \mathcal{I}$*, the state-cardinality of* T *at level* i *equals the state-cardinality of* $T^\perp$ *at level* i*. In other words,* $|V_i| = \left|V_i^\perp\right|$*.*

Proof: From Lemmas 4.9 and 6.12 we know that V_i is equal to the column-space of M_i, and $V_i^{\perp}$ is equal to the column-space of M_i^t. Therefore, by the "row rank=column rank" theorem of linear algebra [HK61], $|V_i| = \left|V_i^{\perp}\right|$. ■

Finally, we have the following theorem.

Theorem 6.15 *Let T be a minimal linear trellis, either conventional or tail-biting, for a linear code $\mathcal{C}$. Then there exists a minimal linear dual trellis $T^{\perp}$ for the dual code $\mathcal{C}^{\perp}$ such that the* SCP *of $T^{\perp}$ is identical to the* SCP *of T.*

Proof: Follows from Corollary 4.17, Lemma 6.11 and Lemma 6.14. ■

Chapter 7

Abstract Characterization of Tail-Biting Trellises

This chapter describes an abstract characterization for linear tail-biting trellises in terms of an equivalence relation defined on a certain language derived from the code. In the context of formal language theory, this characterization is related to the Myhill-Nerode theorem for regular languages. In this chapter we will also review the Myhill-Nerode theorem from formal language theory which is central to the understanding of our abstract characterization of tail-biting trellises. We refer the reader to the classic text [HU77] for more background on formal language theory.

7.1 The Myhill-Nerode Theorem for Regular Languages

We will first survey the principal mathematical ideas necessary for understanding the material in this section. An alphabet Σ is any finite set of symbols. A finite word over Σ is any finite sequence of symbols from Σ. The length of a word w, denoted $|w|$ is the number of symbols composing the word. The empty word, denoted by ϵ, is the word consisting of zero symbols. Therefore $|\epsilon| = 0$. A prefix of a word w is a word x such that there exists a word z and $w = xz$. A suffix of a word w is a word x such that there exists a word z and $w = zx$. The concatenation w of two words w_1, w_2 is written as $w = w_1 w_2$. We will denote by Σ^* the set of all words on Σ. A language L is any subset of Σ^*.

Example 7.1 *Let* $\Sigma = \{0, 1\}$. *Then* $\Sigma^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\}$ *is the set of all binary words. The language* $L = \{\epsilon, 11, 011, 101, 110, 0011, \ldots\}$ *consisting of all words with an even number of* 1*'s is a subset of* Σ^*.

Definition 7.2 *A* deterministic finite automaton *(*DFA*)* M *is a 5-tuple* $(Q, \Sigma, \delta, q_0.F)$, *where* Q *is a finite set of* states, Σ *is a finite* input alphabet, $q_0 \in Q$ *is the* initial state, $F \subseteq Q$ *is the set of* final states, *and* $\delta : Q \times \Sigma \to Q$ *is the* transition function. *That is,* $\delta(q, a)$ *is a state for each state* $q \in Q$ *and input symbol* $a \in \Sigma$.

To formally describe the behaviour of a DFA M on a word $w \in \Sigma^*$, we must extend the transition function δ to apply to a state and w rather than a state and a symbol. We define $\hat{\delta} : Q \times \Sigma^* \to Q$, such that $\hat{\delta}(q, w)$ is the state of the DFA after reading the word w starting in state q. Formally,

$$\hat{\delta}(q, \epsilon) = q, \text{ and } \hat{\delta}(q, wa) = \delta(\hat{\delta}(q, w), a), \ \forall a \in \Sigma, \ w \in \Sigma^* \tag{7.1}$$

For convenience, δ and $\hat{\delta}$ are used interchangeably. A word $w \in \Sigma^*$ is said to be accepted by a DFA $M = (Q, \Sigma, \delta, q_0.F)$, if $\delta(q_0, x) = f$, for some $f \in F$. Therefore, the language $L(M)$ accepted by M is defined as

$$L(M) \stackrel{\text{def}}{=} \{w \in \Sigma^* : \delta(q_0, w) \in F\} \tag{7.2}$$

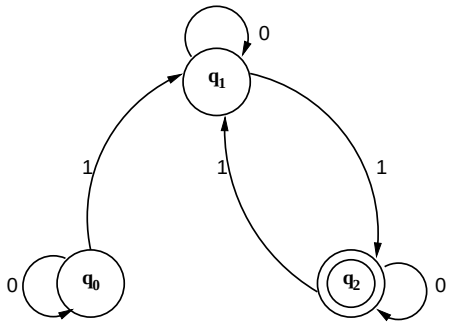


Figure 7.1: A DFA for the language in Example 7.4.

Definition 7.3 *A language* $L \subseteq \Sigma^*$ *is said to be* regular *if it is the language accepted by some finite automaton.*

states	inputs	
Q	0	1
q_0	q_0	q_1
q_1	q_1	q_2
q_2	q_2	q_1

Figure 7.2: $\delta(q, a)$ for the DFA in Figure 7.1.

Example 7.4 *Let $\Sigma = \{0, 1\}$. Consider the language $L \in \Sigma^*$ consisting of words with an even number of 1's. The* DFA *$M = (Q, \Sigma, \delta, q_0, F)$ for L is shown in Figure 7.1. We have $Q = \{q_0, q_1, q_2\}$, $F = \{q_2\}$, and δ is defined in Figure 7.2.*

Definition 7.5 *A* binary relation *$R \subseteq S \times S$ is said to be*

(i) reflexive *if and only if $\forall s \in S$, $(s, s) \in R$.*

(ii) symmetric *if and only if $\forall s, t \in S$, $(s, t) \in R \Rightarrow (t, s) \in R$.*

(iii) transitive *if and only if $\forall s, t, u \in S$, $(s, t) \in R$ and $(t, u) \in R \Rightarrow (s, u) \in R$.*

Definition 7.6 *A reflexive, symmetric and transitive relation R is called an* equivalence relation. *The* equivalence class *of an element of S under R is the set $\{x : (x, a) \in R\}$, denoted $[a]$. The set of equivalence classes is a* partition *of S. The cardinality of this partition is called the* index *of R.*

There is a natural equivalence relation on words induced by any automaton. Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA. For $x, y \in \Sigma^*$, let $(x, y) \in R_M$ if and only if $\delta(q_0, x) = \delta(q_0, y)$ – that is, if and only if the words x and y are taken to the same state from the start state by the action of the automaton. The relation R_M is reflexive, symmetric and transitive, since "=" has these properties, and thus R_M is an equivalence relation.

Definition 7.7 *An equivalence relation $R \subseteq \Sigma^*$ is said to be* right-invariant *if $(x, y) \in R \Rightarrow (xa, ya) \in R$, $\forall a \in \Sigma$.*

For a DFA M, the equivalence relation R_M induced by it has the following two properties.

(i) R_M is of finite index.

(ii) R_M is right-invariant.

We now state the well-known Myhill-Nerode theorem [Myh57, Ner58].

Theorem 7.8 *The following statements are equivalent.*

(i) *The set $L \subseteq \Sigma^*$ is accepted by some* DFA.

(ii) *L is the union of some of the equivalence classes of a right-invariant equivalence relation of finite index.*

(iii) *Let equivalence relative R_L be defined by: $(x, y) \in R_L$ if and only if $\forall z \in \Sigma^*$, $xz \in L$ exactly when $yz \in L$. Then R_L is of finite index.*

It can also be shown that the minimal DFA for a regular language can be derived from Theorem 7.8 [HU77].

Definition 7.9 *Let Σ be a finite alphabet and let L, L_1 and L_2 be subsets of Σ^*. The* concatenation *of L_1 and L_2, denoted L_1L_2, is the set $\{xy : x \in L_1 \text{ and } y \in L_2\}$. Define $L^0 = \{\epsilon\}$ and $L^i = LL^{i-1}$ for $i \geq 1$. The* Kleene closure *of L, denoted L^*, is the set*

$$L^* \stackrel{\text{def}}{=} \bigcup_{i=0}^{\infty} L^i \tag{7.3}$$

and the positive closure *of L, denoted L^+, is the set*

$$L^+ \stackrel{\text{def}}{=} \bigcup_{i=1}^{\infty} L^i \tag{7.4}$$

Example 7.10 *Let $\Sigma = \{0, 1\}$ and let $L_1 = \{0, 01\}$ and $L_2 = \{1, 101\}$. Then $L_1L_2 = \{01, 0101, 011, 01101\}$. We also have $L_1^* = \{\epsilon, 0, 01, 00, 000, 001, 010, 0101, \ldots\}$.*

7.2 An Abstract Characterization of Linear Tail-Biting Trellises

We now propose an abstract characterization of linear tail-biting trellises along the lines of the Myhill-Nerode theorem described in the earlier section. But first we give some necessary definitions.

Definition 7.11 *Let Σ be a finite alphabet and let $L \subseteq \Sigma^*$ be a language over Σ. Let R be a binary relation on L. R is said to be* restricted right-invariant *if $\forall a \in \Sigma$, $(x, y) \in$*

$R \Rightarrow (xa, ya) \in R$, *whenever* $xa,\ ya \in L$. R *is said to be* right-cancellative *whenever* $(xa, ya) \in R \Rightarrow (x, y) \in R,\ \forall a \in \Sigma$. *If* R *is both restricted right-invariant and right-cancellative, then it is* bi-invariant.

Definition 7.12 *Let* R *be an equivalence relation on* $L \subset \mathbb{F}_q^*$. *For a fixed integer* $n \in \mathbb{Z}^+$, R *is said to have an index profile* $\mathcal{J} = \{\iota_i\}_{i=0}^{n-1}$, *if* $\iota_i = |\{[x] \in R : |x| = i \bmod n\}|,\ 0 \leq i \leq n-1$. *We will therefore denote the partition on* R *induced by* $\mathcal{J}$ *as*

$$R = \bigcup_{i=0}^{n-1} R_i \tag{7.5}$$

where $R_i \subseteq R$ *is an equivalence relation of index* ι_i.

Definition 7.13 *Let* R *be an equivalence relation on* $L \subseteq \mathbb{F}_q^*$ *with an index profile of length* n. *Then* R *is said to be* linear *if* $\forall x_1, x_2, y_1, y_2 \in L$, *where* $|x_1| = |x_2|$, $|y_1| = |y_2|$,

$$\{(x_1, y_1), (x_2, y_2)\} \subseteq R_i, \Rightarrow (\alpha x_1 + \beta x_2, \alpha y_1 + \beta y_2) \in R_i,\ \forall \alpha, \beta \in \mathbb{F}_q,\ 0 \leq i \leq n-1. \tag{7.6}$$

From now on we will be using the terms *word* and *vector* interchangeably. Denote by $C = \bigcup_{i=0}^{m} C_i$, the coset decomposition of a linear block code C over C_0 (see Section 3.4). Let P_i denote the language of all prefixes of C_i^*. Define languages U and W on $\mathbb{F}_q$ as follows.

$$U \stackrel{\text{def}}{=} \bigcup_{i=0}^{m} C_i P_i \text{ and } W \stackrel{\text{def}}{=} \bigcup_{i=0}^{m} C_i^+ \tag{7.7}$$

Note here that W is not a concatenation of codewords, but is restricted to concatenation of codewords that lie in the same coset. We will now specify a tail-biting trellis in terms of the equivalence relations defined earlier.

Theorem 7.14 *Let* C *be a linear block code of length* n *over* $\mathbb{F}_q$, *and let the languages* U *and* W *be as defined in (7.7). Then* T *is a reduced one-to-one biproper linear tail-biting trellis representing* C *with* SCP $(h_0, h_1, \ldots, h_{n-1})$, *if and only if* W *is the union of* q^{h_0} *equivalence classes of a linear bi-invariant equivalence relation over* U *with index profile* $\left(q^{h_0}, q^{h_1}, \ldots, q^{h_{n-1}}\right)$.

Proof: ($\Rightarrow$) Assume $T = (V, E, \mathbb{F}_q)$ is a reduced one-to-one biproper linear trellis representing C. Define a trellis transition function $\delta : V \times \mathbb{F}_q \rightarrow V$, where $\delta(v_1, a) = v_2$, if

$(v_1, a, v_2) \in E$. This definition is also extended naturally to paths in T as shown in (7.1).

Define a relation R_T on U as follows. For $x, y \in U$, $(x, y) \in R_T$ if and only if $\exists\ i$ and j such that $\delta(s_i, x) = \delta(s_j, y)$, where s_i and s_j are start states for the subtrellises representing cosets C_i and C_j. It is easy to see that R_T satisfies the properties of reflexivity, symmetry and transitivity, and is hence an equivalence relation. We have $\forall a \in \mathbb{F}_q$, $\delta(s_i, x) = \delta(s_j, y) \Rightarrow \delta(s_i, xa) = \delta(s_j, ya)$, whenever $xa, ya \in U$, and hence $(xa, ya) \in R_T$. Therefore R_T is restricted right-invariant.

Next $\delta(s_i, xa) = \delta(s_j, ya)$, for $xa, ya \in U \Rightarrow \delta(s_i, x) = \delta(s_j, y)$ as T is biproper. Hence R_T is right-cancellative. The construction procedure ensures that R_T is linear as T is linear. Since T is reduced, every state of T is reachable and the index profile of R_T is $\{q^{h_0}, q^{h_1}, \ldots, q^{h_{n-1}}\}$.

($\Leftarrow$) Given a code C of length n, assume that R_C is a linear bi-invariant equivalence relation over U, with index profile $\mathcal{J} = \left(q^{h_0}, q^{h_1}, \ldots, q^{h_{n-1}}\right)$. Construct a tail-biting trellis T_C representing C as follows. The start states of T_C at time index 0 are the equivalence classes $\{[c_i] : c_i \in C_i\}_{1 \le i \le q^{h_0}}$. Also define $\delta_C([x], a) = [xa]$, $x \in U$, $a \in \mathbb{F}_q$. Since R_C is restricted right-invariant, this choice is consistent with the choice of any other representative $y \in [x]$. Let $c_i \in C_i$, for some $i \in \{0, 1, \ldots, q^{h_0}\}$. Then from the construction of T_C, it follows that $[c_i] = C_i^+$. Therefore, $W = \bigcup_{i=1}^{q^{h_0}} \{[c_i] : c_i \in C_i\}$. Note that the trellis T_C is biproper. To see this, for any state $[x]$, there is at most one edge out of $[x]$ on $a \in \mathbb{F}_q$, as R_C is restricted right-invariant. Also there is at most one edge into $[x]$ on $a \in \mathbb{F}_q$ as R_C is right-cancellative. Since R_C is an equivalence relation, T_C is one-to-one. Finally as R_C is linear, it is easy to verify that T_C is also linear with SCP $\{h_0, h_1, \ldots, h_{n-1}\}$. Also, by construction all states of T_C are reachable from at least one of the start states hence T_C is reduced. ■

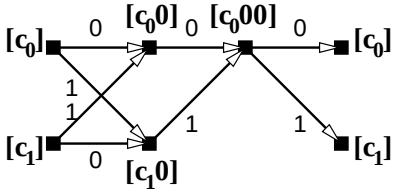


Figure 7.3: A reduced one-to-one non-mergeable tail-biting trellis for the $(3, 2)_2$ code.

Example 7.15 *Consider a* $(3,2)_2$ *code with generator matrix*

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

Therefore $C = \{000, 101, 110, 011\}$. *Let* $C_0 = \{000, 110\}$ *and* $C_1 = \{101, 011\}$ *be the coset decomposition of* C *over* C_0. *Let* U *and* W *be as defined in (7.7). Define the linear bi-invariant equivalence relation* R_C *as follows. Let* $c_0 \in C_0$ *and* $c_1 \in C_1$. *Denote by* E_C *the equivalence classes of* R_C.

$$E_C = \{[c_0], [c_1], [c_0 0], [c_1 0], [c_0 00]\}$$

The trellis T_C *is shown in Figure 7.3.*

An observation that can be made here is that the Myhill-Nerode theorem for deterministic finite automata defines the unique minimal automaton for the language recognized by an automaton and directly leads to a minimization algorithm. Here the automaton obtained from the equivalence relation R_C for the languages defined by each coset are minimal. The choice of coset leaders is specified (indirectly) by the equivalence relation R_C, by noting which words in a coset not containing the all-zero codeword lead to a state on the all-zero codeword path. However, the coset leaders selected may be such that the resulting tail-biting trellis is far from minimal – so unfortunately no useful minimization algorithm can be derived from Theorem 7.14.

Chapter 8

Conclusions

In this concluding chapter, we review our main results and propose avenues of study for the future. This thesis presents unifying views of tail-biting trellises for linear block codes; all of these stem from the observation that a linear tail-biting trellis for a linear block code corresponds to a coset decomposition of the code with respect to a subcode. The thesis addresses a number of problems in the area. These include:

(i) A specification of tail-biting trellises that is a natural generalization of the BCJR construction of conventional trellises for linear block codes. All properties of the original BCJR construction are preserved (except, of course minimality).

(ii) An $\mathcal{O}(n^k)$ dynamic algorithm to compute an $s_{\max}$-minimal trellis for an $(n,k)_q$ linear block code.

(iii) A specification of tail-biting trellises that is a natural generalization of the Forney construction of conventional trellises for linear block codes. This involves the notion of circular pasts and futures as opposed to linear pasts and futures for conventional trellises.

(iv) A simple specification of dual trellises arising from the description of the primal code. The construction is direct and leads to a theorem stating that for every minimal linear trellis T representing a primal code, there exists a minimal linear trellis $T^\perp$ representing the dual code, such that T and $T^\perp$ have identical state-complexity profiles.

(v) An abstract characterization of tail-biting trellises in terms of an equivalence relation defined on a certain language derived from the code.

There are a number of problems that are yet open. The most important one is the problem of computing $s_{\max}$-minimal trellises. In Section 4.2, we presented a dynamic algorithm to compute an $s_{\max}$-minimal trellis for an $(n,k)_q$ code. As a starting point to tackle the $s_{\max}$-minimality problem, it would be interesting to see if this computation can be made "static" – that is, compute an $\mathcal{O}(n)$ number of displacement vectors for the rows of the matrix Ψ, in polynomial time.

Another very interesting problem would be to design a decoding algorithm that operates on a T–BCJR$^{\perp}$ dual trellis and decodes a received word to a primal codeword. This would lead to an efficient decoding algorithm for high rate codes.

Bibliography

[BCJR74] Lalit R. Bahl, John Cocke, Frederick Jelinek, and Josef Raviv. Optimal decoding of linear codes for minimizing symbol error rate. *IEEE Transactions on Information Theory*, 20:284–287, March 1974.

[BMvT78] Elwyn R. Berlekamp, Robert J. McEliece, and H.C.A. van Tilborg. On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, 24:384–386, May 1978.

[CFV99] A. Robert Calderbank, G. David Forney, and Alexander Vardy. Minimal tail-biting trellises: the golay code and more. *IEEE Transactions on Information Theory*, 45:1435–1455, July 1999.

[CT91] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons Inc., New York, 1991.

[DSDR00] Kaustubh Deshmukh, Priti Shankar, Amitava Dasgupta, and B. Sundar Rajan. On the many faces of block codes. In *Symposium on Theoretical Aspects of Computer Science(STACS)*, pages 53–64, 2000.

[For67] G. David Forney. *Final report on a coding system design for advanced solar missions*. Contract NAS2-3637, NASA Ames Research Center, CA, December 1967.

[For73] G. David Forney. The Viterbi algorithm. *Proceedings of the IEEE*, 61:268–278, 1973.

[For88] G. David Forney. Coset codes II: binary lattices and related codes. *IEEE Transactions on Information Theory*, 34:1152–1187, September 1988.

[For94] G. David Forney. Dimension/length profiles and trellis complexity of linear block codes. *IEEE Transactions on Information Theory*, 40:1741–1752, November 1994.

[For01] G. David Forney. Codes on graphs: normal realizations. *IEEE Transactions on Information Theory*, 47:520–548, February 2001.

[HK61] Kenneth Hoffman and Ray Kunze. *Linear Algebra*. Prentice Hall, 1961.

[HU77] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata, Languages and Computation*. Addison Wesley, 1977.

[KS95] Frank R. Kschischang and Vladislav Sorokine. On the trellis structure of block codes. *IEEE Transactions on Information Theory*, 41:1924–1937, November 1995.

[Ksc96] Frank R. Kschischang. The trellis structure of maximal fixed-cost codes. *IEEE Transactions on Information Theory*, 42:1828–1838, November 1996.

[KTFL93] Tadao Kasami, Toyoo Takata, Toru Fujiwara, and Shu Lin. On complexity of trellis structure of block codes. *IEEE Transactions on Information Theory*, 39:1057–1064, March 1993.

[KV02] Ralf Kötter and Alexander Vardy. On the theory of linear trellises. In M. Blaum, P.G. Farrel, and H.C.A. van Tilborg, editors, *Information, Coding and Mathematics*, pages 323–354. MA:Kluwer, 2002.

[KV03] Ralf Kötter and Alexander Vardy. The structure of tail-biting trellises: minimality and basic principles. *IEEE Transactions on Information Theory*, 49:1877–1901, September 2003.

[LKFF98] Shu Lin, Tadao Kasami, Toru Fujiwara, and Marc Fossorier. *Trellises and Trellis-Based Decoding Algorithms for Linear Block Codes*. Kluwer Academic Publishers, 1998.

[LS00] Shu Lin and Rose Y. Shao. General structure and construction of tail-biting trellises for linear block codes. In *IEEE International Symposium on Information Theory, Sorrento, Italy*, page 117, June 2000.

[Mas78] James L. Massey. Foundations and methods of channel encoding. In *International Conference on Information Theory and Systems, NTG-Fachberichte, Berlin*, pages 148–157, 1978.

[McE96] Robert J. McEliece. On the BCJR trellis for linear block codes. *IEEE Transactions on Information Theory*, 42:1072–1092, July 1996.

[McE04] Robert J. McEliece. *The Theory of Information and Coding*. Cambridge University Press, 2004.

[Mud88] Douglas J. Muder. Minimal trellises for block codes. *IEEE Transactions on Information Theory*, 34:1049–1053, September 1988.

[Myh57] John Myhill. Finite automata and the representation of events. Technical Report WADD TR-57-624, Wright Patterson AFB, Ohio, 1957.

[Ner58] Anil Nerode. Linear automaton transformations. *Proceedings of the American Mathematical Society*, 9:541–544, 1958.

[RB99] Ilan Reuven and Yair Be'ery. Tail-biting trellises of block codes: trellis complexity and Viterbi decoding complexity. *IEICE Transactions Fundamentals*, E82-A:2043–2051, October 1999.

[SB00] Yaron Shany and Yair Be'ery. Linear tail-biting trellises, the square-root bound and applications for reed-muller codes. *IEEE Transactions on Information Theory*, 46:1514–1523, July 2000.

[SDDR03] Priti Shankar, Amitava Dasgupta, Kaustubh Deshmukh, and B. Sundar Rajan. On viewing block codes as finite automata. *Theoretical Computer Science*, 290:1775–1795, March 2003.

[Sha48] Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, July 1948.

[SKSR01] Priti Shankar, P. N. Anjaneya Kumar, Harmeet Singh, and B. Sundar Rajan. Minimal tail-biting trellises for certain cyclic block codes are easy to construct. In *International Colloquium on Automata, Languages and Programming (ICALP)*, pages 627–638, 2001.

[SvT79] Gustav Solomon and H.C.A. van Tilborg. A connection between block and convolutional codes. *SIAM Journal on Applied Mathematics*, 37:358–369, October 1979.

[Var98] Alexander Vardy. Trellis structure of codes. In V.S. Pless and W.C. Huffman, editors, *Handbook of Coding Theory*. Elsevier, 1998.

[VK96] Alexander Vardy and Frank R. Kschischang. Proof of a conjecture of McEliece regarding the expansion index of the minimal trellis. *IEEE Transactions on Information Theory*, 42:2027–2034, November 1996.

[vL99] J. H. van Lint. *Introduction to Coding Theory*. Springer-Verlag, Berlin, 1999.

[Wib96] Niclas Wiberg. *Codes and decoding on general graphs*. Ph.D. dissertation, Department of Electrical Engineering, Linköping University, Linköping, Sweden, 1996.

[WLK95] Niclas Wiberg, Hans-Andrea Loeliger, and Ralf Kötter. Codes and iterative decoding on general graphs. *European Transactions on Telecommunications*, 6:513–526, September 1995.

[Wol78] J.K. Wolf. Efficient maximum-likelihood decoding of linear block codes using a trellis. *IEEE Transactions on Information Theory*, 24:76–80, 1978.