Minimal Tail-Biting Trellises for Certain Cyclic Block Codes Are Easy to Construct

Priti Shankar, P.N.A. Kumar, Harmeet Singh, and B.S. Rajan

Indian Institute of Science, Bangalore, 560012.
{priti,pnakumar}@csa.iisc.ernet.in,
harmeet@sasken.com,bsrajan@ece.iisc.ernet.in

Abstract. We give simple algorithms for the construction of generator matrices for minimal tail-biting trellises for a powerful and practical subclass of the linear cyclic codes, from which the combinatorial representation in the form of a graph can be obtained by standard procedures.

Keywords: linear block codes, cyclic codes, Reed-Solomon codes, tail-biting trellises.

1 Introduction

Trellis descriptions of block codes [1,15,8,10,3,6,9] are *combinatorial* descriptions, as opposed to the traditional *algebraic* descriptions of block codes. A minimal *conventional* trellis for a linear block code is just the transition graph for the minimal finite state automaton which accepts the language consisting of the set of all codewords. With such a description, the decoding problem reduces to finding a cheapest accepting path in such an automaton (where transitions are assigned costs based on a channel model.) However, trellises for many useful block codes are often too large to be of practical value. Of immense interest therefore, are *tail-biting* trellises for block codes, recently introduced in [2], which have reduced state complexity. The strings accepted by a finite state machine represented by a trellis are all of the same length, that is the *block length* of the code. Coding theorists therefore attach to all states that can be reached by strings of the same length l, a *time index* l. Conventional trellises use a linear time index, whereas tail-biting trellises use a circular time index. It has been observed [14] that the maximum state cardinality of a tail-biting trellis at any time index can drop to the square root of the maximum state cardinality (over all time indices) of a conventional trellis for the code, thus increasing the potential practical applications of trellis representations for block codes. In this paper, we show that finding a minimal tail-biting trellis corresponds to picking basis vectors of the vector space defining the code in a particular way, and using the selected vectors to build up the trellis. We then show that for various subclasses of cyclic codes, obtaining vectors that span the space and that also yield minimal tail-biting trellises is easy. Section 2 presents the background. Section 3 presents our results on cyclic codes, and Section 4 gives results for Reed-Solomon codes.

2 Background

We give a very brief background on subclasses of block codes called linear codes. Readers are referred to the classic text [7].

Let $GF(q)$ denote the field with q elements. It is customary to define linear codes algebraically as follows:

Definition 1. *A linear block code C of length n over a field $GF(q)$ is a k-dimensional subspace of an n-dimensional vector space over the field $GF(q)$ (such a code is called an (n,k) code.)*

The most common algebraic representation of a linear block code is the generator matrix G. A $k \times n$ matrix G where the rows of G are linearly independent and which generate the subspace corresponding to C is called a *generator matrix* for C. Figure 1 shows a generator matrix for a $(4,2)$ linear code over $GF(2)$, consisting of the four codewords in the set $\{0000,0110,1001,1111\}$. A *cyclic* linear

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

Fig. 1. Generator matrix for a $(4,2)$ linear binary code

block code satisfies the additional property that any cyclic shift of a codeword is also a codeword. (The code of the example above is linear but not cyclic.) Bose-Chaudhari-Hocquengham(BCH) codes and Reed-Solomon codes are the best known cyclic codes which have many practical applications. Cyclic (n,k) codes are generated by k multiples modulo $x^n - 1$ of a polynomial $g(x)$ of degree $n-k$, that divides $x^n - 1$. A codeword corresponds to the coefficients of the polynomial. The polynomial $g(x)$ is chosen to be monic and has degree $n-k$. The codewords corresponding to the multiples $g(x), xg(x), \ldots, x^{k-1}g(x)$ form a basis for the subspace that defines the code. The *parity check polynomial* $h(x)$ (of degree k) is defined as $h(x) = (x^n - 1)/g(x)$. For BCH codes, $g(x)$ has coefficients in a *ground field* $GF(q)$ and roots in an *extension field* $GF(q^m)$. For Reed-Solomon codes, the coefficients and roots of $g(x)$ are in the same field. An example of a BCH code is a binary (7,4) Hamming code. This is generated by $g(x) = x^3 + x + 1$ and has the generator matrix shown in Figure 2. The polynomial $g(x)$ for the (7,4) Hamming code above, has as roots, $\alpha, \alpha^2, \alpha^4$ where α is a primitive element of the field $GF(2^3)$. The polynomial $g(x)$ itself has coefficients in $GF(2)$. The parity check polynomial for this code is $h(x) = x^4 + x^2 + x + 1$ The polynomial $h(x)$ has as roots, all the remaining powers of α, namely, $\alpha^3, \alpha^5, \alpha^6$, and 1. Thus $g(x)$ and $h(x)$ between them have as roots all the non zero elements of the cyclic multiplicative group of the field $GF(2^3)$. These are the seven roots of $x^7 - 1$.

A general block code also has a *combinatorial* description in the form of a *trellis*. We borrow from Kschischang et al. [6] the definition of a trellis for a block code.

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

Fig. 2. Generator matrix for the $(7, 4)$ binary Hamming code

Definition 2. *A trellis for a block code C of length n, is an edge labeled directed graph with edge labels drawn from a set A, with a distinguished* root *vertex s, having in-degree* 0 *and a distinguished* goal *vertex f having out-degree* 0*, with the following properties:*

1. *All vertices can be reached from the root.*
2. *The goal can be reached from all vertices.*
3. *The number of edges traversed in passing from the root to the goal along any path is n.*
4. *The set of n-tuples obtained by "reading off" the edge labels encountered in traversing all paths from the root to the goal is C.*

The length of a path (in edges) from the root to any vertex is unique and is sometimes called the *time index* of the vertex. It is well known that minimal trellises for linear block codes are unique [10] and constructable from a generator matrix for the code [6]. In contrast, minimal trellises for non-linear codes are, in general, neither unique, nor deterministic [6]. Figure 3 shows a trellis for the linear code in Figure 1. Figure 4 shows the minimal conventional trellis for the

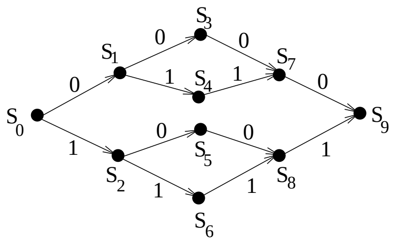

Fig. 3. A trellis for the linear block code of Figure 1 with $S_0 = s$ and $S_9 = f$

Hamming code of Figure 2. Minimal trellises for linear codes are transition graphs for bideterministic automata. The trellises are said to be *biproper*. Biproper trellises minimize a wide variety of structural complexity measures. McEliece [9] has defined a measure of Viterbi decoding complexity in terms of the number of edges and vertices of a trellis, and has shown that the biproper trellis is

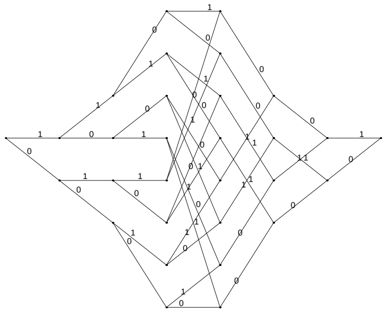

Fig. 4. A conventional trellis for the Hamming code of Figure 2

the "best" trellis using this measure, as well as other measures based on the maximum number of states at any time index, and the total number of states.

We briefly mention the algorithm given in [6] for constructing a minimal trellis from a generator matrix(in a specified form) for the code. An important component of the algorithm is the *trellis product* construction, whereby a trellis for a "sum" code can be obtained as a product of component trellises. The set of vertices of the product trellis at each time index, is just the Cartesian product of the vertices of the component trellis. If we define the i^{th} section as the set of edges connecting the vertices at time index i to those at time index $i+1$, then the edge count in the i^{th} section is the product of the edge counts in the i^{th} section of the individual trellises. Before the product is constructed we put the matrix in *trellis oriented form* described now. Given a non zero codeword $C = (c_1, c_2, \ldots c_n)$, $start(C)$ is the smallest integer i such that c_i is non zero. Also $end(C)$ is the largest integer for which c_i is nonzero. The *linear span* of C is $[start(C), end(C)]$. By convention the span of the all 0 codeword $\mathbf{0}$ is the empty span [ ]. The minimal trellis for the binary $(n, 1)$ code generated by a nonzero codeword with span $[a, b]$ is constructed as follows. There is only one path up to $a-1$ from index 0, and from b to n. From $a-1$ there are 2 outgoing branches diverging(corresponding to the 2 multiples of the codeword), and from $b-1$ to b, there are 2 branches converging. For a code over $GF(q)$ there will be q outgoing branches and q converging branches. It is easy to see that this is the minimal trellis for the 1-dimensional code, and is called the *elementary* trellis corresponding to the codeword. To generate the minimal trellis for C we first put the trellis into *trellis oriented form*, where for every pair of rows, with spans $[a_1, b_1], [a_2, b_2], a_1 \neq b_1$ and $a_2 \neq b_2$. We then construct individual trellises for the k 1-dimensional codes as described above, and then form the trellis product. Conversion of a generator matrix into trellis oriented form requires a sequence of operations similar to Gaussian elimination, applied twice. In the first phase, we apply the method to ensure that each row in the matrix starts its first nonzero entry at a time index one higher than the previous row. In the second phase we ensure that no two rows have their last nonzero entry at the same time index.

The complexity of the algorithm is $O(k^2n + s)$ for an (n, k) linear code whose minimal trellis has s states. We see that the generator matrices displayed earlier are already in trellis oriented form. The elementary trellises for the two rows of the generator matrix in Figure 1 are shown in Figure 5 below. The product of these two elementary trellises yields the trellis in Figure 3.

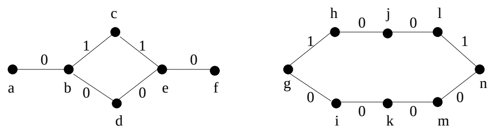

Fig. 5. Elementary trellises for the rows of the generator matrix in Figure 1

A *tail-biting trellis* is defined on a circular time axis of length n which is usually identified with $Z_n = \{0, 1, 2, \ldots n-1\}$, the integers modulo n. All arithmetic operations on indices are performed modulo n. For each time index $j \in Z_n$ there is a finite state space S_j. All edges of the tail-biting trellis are between S_j and $S_{j+1(modn)}$, and as in the conventional case, are labeled with elements from A. One can think of a tail-biting trellis as defined on a sequential time axis with

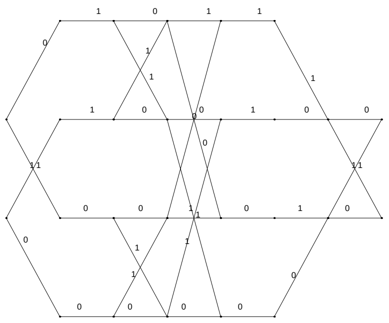

Fig. 6. A tail-biting trellis for the Hamming code of Figure 2

$S_0 = S_n$ and by restricting valid paths to those that begin and end at the same state. Figure 6 shows a tail-biting trellis for the Hamming code of Figure 2, with $|S_0| = |S_n| = 2$. One can also regard a conventional trellis as a tail-biting trellis with $|S_0| = |S_n| = 1$.

The *state cardinality profile* of a tail-biting trellis is the sequence $(|S_0|, |S_1|, \ldots$ $|S_{n-1}|)$. For example, the state cardinality profile for the tail-biting trellis for the (7,4) Hamming code in Figure 6 is (2,4,4,4,4,4,2). The *maximum state cardinality* of a trellis is defined as $S_{max} = max\{|S_0|, |S_1|, \ldots |S_{n-1}|\}$. The minimum state cardinality, S_{min} can similarly be defined. It is well known that for conventional trellises representing linear block codes, there is a unique minimal trellis for a given linear block code. However this is not true in general for tail-biting trellises. There are three notions of minimality that have been suggested for tail-biting trellises. In [2] a trellis is called μ-minimal if S_{max} is minimized over all possible permutations of the time index, and choices of generator matrices. A second definition is π-minimality, also defined in [2] where the goal is to minimize the product of all state space sizes over all permutations of the time indices and choices of generator matrices. Kotter and Vardy [5] define weaker notions of minimality. In their definitions a coordinate ordering is considered to be fixed. One definition is that of Θ-minimality. A trellis T is said to be smaller than or equal to another trellis T' denoted by $T \leq_{\Theta} T'$ if $|S_i| \leq |S'_i|$ for all $i \in \{0, 1, \ldots n-1\}$. If at least one of the indices has a strict inequality in the set of inequalities above, then $T < T'$. A second ordering is on the basis of the product of all state space sizes, as defined earlier, except that permutations of the time index are not considered.

We end this section by stating a few bounds. All of these are found in the excellent survey on trellises for block codes [13] and in [2]. Let C be an (n, k) linear block code over the field $GF(q)$.

1. For a conventional trellis, $S_{max} \leq min(q^k, q^{n-k})$
2. For a tail-biting trellis the product complexity, $\pi \geq q^{(d-1)k}$, where d is the minimum of the Hamming distances between all pairs of codewords. This will be referred to as the *total span* lower bound.
3. For a tail-biting trellis $S_{max} \geq q^{\lceil (d-1)k/n \rceil}$
4. If S_{mid} is the minimum possible state-space size of a conventional trellis for a code C at its midpoint under any permutation of its time index, then $S_{max} \geq \sqrt{S_{mid}}$. This is referred to as the *square root* lower bound.

3 An Algorithm for the Construction of Minimal Tail-Biting Trellises for Cyclic Codes

We first give another definition of minimality. Our definition is for a fixed coordinate ordering. We say that trellis T is Σ-minimal if $\sum_i |S_i| \leq \sum_i |S_i|'$ for any other trellis T'. This is a natural definition from the point of view of automata theory. However, we must recall that we are not dealing with conventional automata, but with automata having multiple start and multiple final states, and with a restricted definition of acceptance. We have observed that subject to minimization of the product of all state space sizes, the minimization of this quantity also minimizes S_{max}. Also this definition seems to favour "flat" trellises over others with the same product space size. Flat minimal trellises, in fact achieve the square root lower bound [2]. Another definition that favours flat trellises is Δ-minimality. We say that a tail-biting trellis is Δ-minimal for a given product size if $log(S_{max}) - log(S_{min})$ is minimal, where the logarithm is to the base q if the code symbols are from $GF(q)$. The decoding complexity in a conventional trellis is closely related to the number of edges and nodes of the trellis. This is also true for tail-biting trellises [11]. It is thus of interest to minimize the Σ-size. We show that in the cases considered here, Δ-minimality implies Σ-minimality.

We now define a *circular span* of a codeword. Let $C = (c_1, c_2 \dots c_n)$ be a codeword such that c_i and c_j are non-zero, $i < j$ and all components between i and j are zero. Then $[j, i]$ is said to be a *circular span* of the codeword. Note that while the linear span of a codeword is unique, a circular span is not, as it depends on the consecutive run of zeros chosen. Given a codeword and a circular span $[j, i]$, there is a unique elementary trellis corresponding to it. If the code symbols are from $GF(q)$, then the trellis has q states from index 0 to index $i-1$, one state from index i to index j and q states from index $j+1$ to index n. If the start states are numbered from 1 to q and final states likewise, only i to i paths are codewords. Shany and Beery[12] have shown that any linear tail-biting trellis can be constructed from a generator matrix whose rows can be partitioned into two sets, those which are taken to have linear span, and those taken to have circular span. The tail-biting trellis is then formed as a product of the elementary trellises corresponding to these rows. Thus the problem of constructing a minimal tail-biting trellis is then reduced to finding a basis for the subspace constituting the code, and a choice of spans, such that the corresponding elementary trellises yield a product that corresponds to a minimal tail-biting trellis.

It is convenient to introduce some notation at this point. A state complexity profile of the form $(q^l, q^l, \dots i_1$ times, $q^j, q^j, \dots i_2$ times, $q^m, q^m, \dots i_3$ times) is represented as $((q^l)^{i_1}, (q^j)^{i_2}, (q^m)^{i_3})$. Let c be a codeword with linear span $[i, j]$. Then the elementary conventional trellis for the codeword has a profile of the form $(1^{i+1} q^{j-i} 1^{n-j-1})$.We abbreviate this as $([1, q, 1], [i+1, j])$, where indices in the trellis begin at 0, as the state cardinality remains constant at q in the interval $[i+1, j]$.

For a codeword with circular span $[j, i]$ the elementary tail–biting trellis has profile $(q^{i+1}, 1^{j-i}, q^{n-j-1})$.This is abbreviated as $([q, 1, q], [j+1, i])$, as the state cardinality remains constant at q over the circular interval $[j+1, i]$. It is clear that the product construction will yield linear trellises where the state cardinality at any time index is always a power of q, where the code is over $GF(q)$.

We now describe an algorithm for the construction of minimal tail–biting trellises, for a subclass of cyclic codes. We establish the basis for the construction by a sequence of results. Firstly we recall, that the generator matrix formed by the vector representation of $g(x), xg(x), x^2g(x), \dots x^{k-1}g(x)$ gives us the minimal conventional trellis for the code. We aim to get a minimal tail-biting trellis having the same product space size as this one. Since each row has linear span of width $(n-k)$, and there are k rows, the product of the state cardinalities is $q^{k(n-k)}$. Since this is a minimal trellis, we cannot do better. Consequently, a lower bound for the maximum size of the state space at any time index is $q^{\lceil k(n-k)/n \rceil}$. This improves the total span lower bound on S_{max} for this class, as $n - k \geq d - 1$. We also note that any row with span (linear or circular) exceeding $n-k$ will increase the product state space size of the code. We can therefore restrict our attention to rows having span of width $n-k$. We prove the main result through a sequence of lemmas. We assume that the codes under consideration are (n, k) cyclic codes.

Lemma 1. *Let $gcd(n, k) = 1$.Then the minimal tail-biting trellis for the cyclic code cannot be flat.*

Proof. Each row contributes a factor q^{n-k} to the product of the state cardinality and there are k rows. Thus the total product state cardinality is $q^{k(n-k)}$. For a flat trellis this must be distributed evenly among the n columns. Thus n must divide $k(n-k)$ which implies n must divide k^2. Since $gcd(n, k) = 1$, this is not possible, and hence a flat trellis cannot exist for this case.

Lemma 2. *For a minimal non-flat trellis for a cyclic code, $log(S_{max}) - log(S_{min})$
≥ 1.*

Proof. The proof follows from the fact that the state cardinality at each time index is always a power of q.

A trellis which has r jumps in its state cardinality profile is called an r-jump trellis. For a given trellis, if $log(S_{max}) - log(S_{min}) = \delta$, the trellis is called a δ-trellis. The tail-biting trellis for the Hamming code in Figure 6 is a 2-jump 1-trellis. We will be dealing with very restricted kinds of tail-biting trellises.These will have state cardinality profiles of the form $([q^l, q^{l+1}, q^l], [i, j])$ or $([q^{l+1}, q^l, q^{l+1}], [j, i])$. The first trellis is of type LHL(L for Low , H for High) and the second of type HLH. We will sometimes use the notation (LHL, i, j, δ) or (HLH, i, j, δ) to refer to 2-jump δ-trellises where the state cardinality remains constant at S_{max} over the linear interval $[i, j]$ or circular interval $[j, i]$ as the case may be. Note that 1-jump trellises do not exist and 0-jump trellises are flat.

We next present a technique to construct a minimal tail-biting trellis for a (n, k) cyclic code when $gcd(n, k) = 1$.

Lemma 3. *The trellis product of elementary trellises corresponding to the code polynomials $g(x), x^{(n-k)}g(x), x^{2(n-k)}g(x), \ldots, x^{(k-1)(n-k)}g(x)$ where all the products are modulo $x^n - 1$, yields a 2-jump 1-trellis with $log(S_{max}) = \lceil (k(n-k)/n) \rceil$.*

Proof. We construct the product trellis step-by-step, forming a trellis product by including one elementary trellis at each step in the order above. At each step the trellis is shown to be a 2-jump 1-trellis. We prove the following by induction.

Hypothesis The trellis generated by the first i codewords $1 \leq i \leq k$ is a 2-jump 1-trellis of type $(LHL, 1, (i)(n-k) \bmod n, 1)$, with $log(S_{max}) = \lceil (i(n-k)/n) \rceil$.

Basis i=1 The codeword corresponding to $g(x)$ has linear span $[0, n-k]$, and generates a $(LHL, 1, n-k, 1)$ trellis. Also $log(S_{max}) = 1$, proving the hypothesis for this case.

Induction Assume the hypothesis is true for the product of the first i or fewer trellises. Thus the trellis generated after i steps is a $(LHL, 1, (i)(n-k) \bmod n, 1)$ trellis, with $log(S_{max}) = \lceil (i(n-k)/n) \rceil$. The next codeword to be added is $x^{(i)(n-k)}g(x) \bmod (x^n - 1)$. If this has linear span, the trellis is of type

$(LHL, (i)(n-k)+1 \bmod n, (i+1)(n-k) \bmod n, 1)$. The product trellis is then of the type $(LHL, 1, (i+1)(n-k) \bmod n, 1)$ and there is no increase in S_{max}. If the next codeword has a circular span, then its trellis is of type $(HLH, (i)(n-k)+1 \bmod n, (i+1)(n-k) \bmod n, 1)$. (Here $i(n-k) \bmod n > (i+1)(n-k) \bmod n$.) Thus the jumps in the product trellis will be at 1 and $(i+1)(n-k)+1 \bmod n$. Also $log(S_{max})$ increases by 1 in this case, as the number of states from indices $i(n-k) \bmod n$ to $n-1$, and 0 to $(i+1)(n-k) \bmod n$ increases by a factor of q. We see that whenever the new elementary trellis to be added has circular span, the value of S_{max} increases by a factor of q. This happens $\lceil k(n-k)/n \rceil$ times. After adding k elementary trellises, the tail-biting trellis construction is complete. The trellis so constructed achieves the improved lower bound on S_{max} for cyclic codes.

While we have proved that the k codewords selected by the lemma do indeed give a trellis with a minimal value of S_{max} we need to show that they form a basis for the subspace defining the code. Thus we need to prove that they are linearly independent. For the next lemma we make the assumption that $gcd(q, n) = 1$. This implies that the generator and parity check polynomials do not have repeated roots. Most practical cyclic codes satisfy this property.

Lemma 4. *If $gcd(n, k) = 1$ and $gcd(q, n) = 1$ then the vectors corresponding to codewords $g(x), x^{(n-k)}g(x), x^{2(n-k)}g(x), \ldots, x^{(k-1)(n-k)}g(x)$ where all the products are modulo $x^n - 1$, are linearly independent.*

Proof. Assume that the vectors are linearly dependent.Then there exist scalars $a_0, a_1, \ldots, a_{k-1}$ in the field $GF(q)$ such that

$$(a_0 + a_1x^{n-k} + \ldots + a_{k-1}x^{(k-1)(n-k)})(g(x)) \equiv 0 \; mod(x^n - 1) \tag{1}$$

Let $A(x) = a_0+a_1x^{n-k}+\ldots+a_{k-1}x^{(k-1)(n-k)}$. From (1) and from the definition of $h(x)$, we conclude that $A(x)$ has as roots all the k roots of $h(x)$. Define $b(x) = a_0 + a_1x + \ldots + a_{k-1}x^{k-1}$. If $\beta \in GF(q^m)$ is a root of $A(x)$, then β^{n-k} is a root of $b(x)$. Also if β_1, β_2 are distinct roots of $h(x)$, β_1^{n-k}, β_2^{n-k} are distinct roots of $b(x)$. Else, if $\beta_1 = \beta^{i_1}$ and $\beta_2 = \beta^{i_2}$ where β generates the cyclic group of order n, $\beta^{(i_1-i_2)(n-k)} = 1$ implying that $i_1 - i_2 \equiv 0 (\text{mod } n)$, thereby giving a contradiction. Hence the vectors are linearly independent.

We finally present the main theorem.

Theorem 1. *For an (n, k) cyclic code over $GF(q)$ with $gcd(q, n) = 1, gcd(n, k) = 1$, there exists a choice of spans, such that the product of elementary trellises corresponding to the codewords $g(x), x^{(n-k)}(g(x)), x^{2(n-k)}(g(x)), \ldots, x^{(k-1)(n-k)}$ $(g(x))$ gives a Δ-minimal trellis for the code which is also Σ-minimal.*

Proof. The Δ-minimality follows from Lemmas 1, 2, 3 and 4. We next show that the Δ-minimal trellises constructed using the procedure above are also $\Sigma-$minimal. The trellis, say T_1, constructed by the above method has $log(S_{max}) - log(S_{min}) = 1$. Assume some other tail biting trellis(say T_2) which is π-minimal, also has $log(S_{max}) - log(S_{min}) = 1$. We show that such a trellis

has a state cardinality profile that is a permutation of T_1. Since a flat trellis does not exist, this trellis is $\Sigma - minimal$. Since $log(S_{max}) - log(S_{min}) = 1$ in T_1, it has a state cardinality of q^{m_1-1} at t_1 time indices and q^{m_1} at $n-t_1$ time indices for some m_1. Similarly T_2 has a state cardinality of q^{m_2-1} at t_2 time indices and q^{m_2} at $n-t_2$ time indices for some m_2,where $m_1, m_2 \geq 1$ and $t_1, t_2 \geq 1$. Without loss of generality assume $t_1 \geq t_2$. Since both of them are π-minimal,we have
$(m_1 - 1) * (t_1) + (m_1) * (n - t_1) = (m_2 - 1) * (t_2) + (m_2) * (n - t_2)$
$(m_1 * n) - t_1 = (m_2 * n) - t_2$
$n * (m_1 - m_2) = t_1 - t_2$
Since $t_1 - t_2 < n$, this is only possible if $t_1 - t_2 = 0$ implying that $m_1 = m_2$. This proves that for a given (n, k) cyclic code, with $gcd(n, k) = 1$, assuming π-minimality, the Σ-minimal trellis is unique up to permutations of the state cardinality profile and is the same as the Δ-minimal trellis.

Figure 6 is the minimal tail-biting trellis for the (7,4) cyclic Hamming code using our technique. We next turn our attention to cyclic codes for which $gcd(n, k) = t = gcd(n, n-k) > 1$. We use the property that if $gcd(n, k) = t$ then the smallest multiple of $n - k$ which is a multiple of n is $(n/t) \times (n - k)$.

Lemma 5. *Let $g(x)$ be the generator polynomial of an (n, k) cyclic code with $gcd(n, k) = t > 1$. Also assume that $k < n/t$. Then the codewords corresponding to the polynomials $g(x), x^{(n-k)}g(x), x^{2(n-k)} \ldots x^{(k-1)(n-k)}$ where all products are modulo $x^n - 1$, generate a minimal 2-jump 1-trellis for the code, provided the following condition is satisfied: if β^i and β^j are roots of $h(x)$ where β generates the cyclic group of order n, then $i - j \neq 0(mod\ n/t)$.*

Proof. From Lemma 3, we know that the vectors generate a trellis which is a 2-jump 1-trellis that is Σ-minimal as well as Δ-minimal. To show that it is a trellis for the code, we need to show that the k vectors are linearly independent. Assume they are not. The proof proceeds along exactly the same lines as that of Lemma 4, with $A(x), b(x), \beta^i, \beta^j$ being defined as in that lemma. If β^i and β^j map to the same root of $b(x)$ then $(i - j)(n - k) = 0(\bmod n)$. But the smallest value of $i - j$ for which this can happen is $i - j = n/t$. Since this is not possible, the k roots of $h(x)$ map into distinct roots of $b(x)$ giving a contradiction, as the degree of $b(x)$ is $k - 1$. Hence the vectors are linearly independent and generate a minimal trellis for the code.

4 Minimal Tail-Biting Trellises for Reed-Solomon Codes

We now present some results on mimimal trellises for Reed-Solomon codes. We recall that for such codes the code symbols and the roots of $g(x)$ are in the same field. We look at cases when $gcd(n, k) = t > 1$, as the case when t is 1 is covered by the results of the previous section. It is easy to prove that minimal flat trellises exist for several subclasses of Reed-Solomon codes.

The lemma below has a simple proof available in [4]. For the sake of brevity the proof is not reproduced here.

Lemma 6. *Let $g(x)$ be the generator polynomial for an (n,k) Reed-Solomon code.Then the codewords represented by polynomials $g(x), x^{i_1}g(x), x^{i_2}g(x), \ldots,$ $x^{i_{k-1}}g(x)$ where $i_1, i_2, \ldots, i_{k-1}$ are distinct positive integers between 1 and n-1(inclusive),(and all products are modulo $x^n - 1$), are linearly independent.*

Lemma 7. *Let $t = gcd(n,k)$.Then the integers $i(n-k)$ mod n,$i = 0, 1, \ldots n/t-1$ are all distinct, and the vectors corresponding to $g(x), x^{n-k}g(x), x^{2(n-k)}g(x), \ldots,$ $x^{(n/t-1)(n-k)}g(x)$, where all the products are modulo x^n-1, generate a flat trellis.*

Proof. We can easily see that the integers are all distinct. The construction procedure of Lemma 3 will after $n/t - 1$ products, generate a trellis with a "jump" up at index 1 and a "jump" down at index $n/t(n-k)+1 = 1$ modulo n. Thus after the $n/t - 1^{st}$ product, the trellis will be a 0-trellis and is hence flat.

Lemma 8. *Assume that $n/t = k$. Then the vectors $g(x), x^{n-k}g(x), x^{2(n-k)}g(x),$ $\ldots, x^{(k-1)(n-k)}g(x)$ where all products are modulo $x^n - 1$, generate a minimal flat trellis for the code. If $n/t > k$, then the vectors generate a minimal 2-jump 1-trellis for the code.*

Proof. We can easily see that in the first case n divides k^2. By Lemmas 6 and 7 we see that the k vectors are distinct, linearly independent and also generate a minimal flat trellis for the code. The proof for the second case follows directly from Lemmas 3 and 6.

Lemma 9. *Let $t = gcd(n,k)$. Then if n/t divides k, the vectors in the union of the sets below:*
$\{g(x), x^{n-k}g(x) \ldots x^{(n/t-1)(n-k)}g(x)\}$
$\{xg(x), x^{n-k+1}g(x) \ldots x^{(n/t-1)(n-k)+1}g(x)\}$
$\{x^2g(x), x^{n-k+2}g(x) \ldots x^{(n/t-1)(n-k)+2}g(x)\}$
$\vdots$
$\{x^{(k/(n/t))-1}g(x), x^{n-k+(k/(n/t))-1}g(x) \ldots x^{(n/t-1)(n-k)+(k/(n/t))-1}g(x)\}$
generate a flat trellis for the code.

Proof. We outline the proof here. Firstly we see that n divides k^2. Also, it is easy to see that the vectors are all distinct. We note from Lemma 7, that the vectors in each set produce a flat trellis of width $q^{((n-k)\times(n/t))/n}$. (Cyclic shifts of a generator matrix that produces a flat trellis also produce flat trellises). The product trellis is just a product of flat trellises, and is hence also flat. To see that it is minimal, we see that the width is $q^{1/n((n/t)\times(n-k)\times k/(n/t))} = q^{k(n-k)/n}$. Thus the trellis is minimal.

The only case left is that when $k > n/t$ but n/t does not divide k. It is easy to see that in this case we will get $\lceil(k/(n/t)\rceil$ sets, such that the last set contains less than n/t vectors, and generates a 2-jump 1-trellis, while the others all generate flat trellises. The result follows from these two observations. Thus we see that all cases for Reed-Solomon codes are covered. We consolidate all the results of this section into one theorem.

Theorem 2. *Reed-Solomon (n,k) codes have minimal flat trellises if and only if n divides k^2. If n does not divide k^2 the minimal trellises are 2-jump 1-trellises.*

Proof. The necessity of the condition for flat trellises is shown in Lemma 1. Let $gcd(n, k) = t$, and assume n divides k^2. We can write $n_1 \times n = k^2 = k_1 \times t \times k$, for integers n_1 and k_1. Clearly $n_1/k_1 = l$, is an integer, as $gcd(k_1, n) = 1$. Therefore $l \times n/t = k$ which, together with lemmas 8 and 9 show the sufficiency of the condition for a flat trellis. The case n does not divide k^2 is covered by lemma 3, the second half of lemma 8 and the previous paragraph.

References

1. L.R.Bahl, J.Cocke, F.Jelinek, and J. Raviv, Optimal decoding of linear codes for minimizing symbol error rate, *IEEE Trans. Inform. Theory* **20**(2), March 1974, pp 284-287.
2. A.R.Calderbank, G.David Forney,Jr., and Alexander Vardy, Minimal Tail-Biting Trellises: The Golay Code and More, *IEEE Trans. Inform. Theory* **45**(5) July 1999, pp 1435-1455.
3. G.D. Forney, Jr. and M.D. Trott, The dynamics of group codes:State spaces, trellis diagrams and canonical encoders, *IEEE Trans. Inform. Theory* **39**(5) Sept 1993, pp 1491-1513.
4. Harmeet Singh, On Tail-Biting Trellises for Linear Block Codes, M.E. Thesis, Department of Electrical Communication Engineering, Indian Institute of Science, Bangalore, 2001.
5. Ralf Kotter and Vardy, A.,Construction of Minimal Tail-Biting Trellises, in *Proceedings IEEE Information Theory Workshop* (Killarney, Ireland, June 1998), 72-74.
6. F.R.Kschischang and V.Sorokine, On the trellis structure of block codes, *IEEE Trans. Inform. Theory* **41**(6), Nov 1995, pp 1924-1937.
7. F.J. MacWilliams and N.J.A. Sloane,*The Theory of Error Correcting Codes*, North-Holland, Amsterdam, 1981.
8. J.L.Massey, Foundations and methods of channel encoding, in *Proc. Int. Conf. on Information Theory and Systems* **65**(Berlin, Germany) Sept 1978.
9. R.J.McEliece, On the BCJR trellis for linear block codes, *IEEE Trans. Inform. Theory* **42**, November 1996, pp 1072-1092.
10. D.J. Muder, Minimal trellises for block codes, *IEEE Trans. Inform. Theory* **34**(5), Sept 1988, pp 1049-1053.
11. Priti Shankar, P.N.A. Kumar, K.Sasidharan and B.S.Rajan, ML decoding of block codes on their tail-biting trellises, to appear in *Proceedings of the 2001 IEEE International Symposium on Information Theory.*
12. Yaron Shany and Yair Be'ery, Linear Tail-Biting Trellises, the Square-Root Bound, and Applications for Reed-Muller Codes, *IEEE Trans. Inform. Theory* **46** (4), July 2000, pp 1514-1523.
13. A.Vardy, Trellis structure of codes, in *Handbook of Coding Theory,*V.S. Pless and W.C. Huffman, Eds., Elsevier Science, 1998.
14. N.Wiberg, H.-A. Loeliger and R.Kotter, Codes and iterative decoding on general graphs, *Eoro. Trans. Telecommun.,***6**, Sept 1995, pp 513-526.
15. J.K. Wolf, Efficient maximum-likelihood decoding of linear block codes using a trellis, *IEEE Trans. Inform. Theory* **24**, pp 76-80.