

Tail-Biting Trellises of Block Codes: Trellis Complexity and Viterbi Decoding Complexity

Ilan REUVEN[†] and Yair BE'ERY[†], *Nonmembers*

SUMMARY Tail-biting trellises of linear and nonlinear block codes are addressed. We refine the information-theoretic approach of a previous work on conventional trellis representation, and show that the same ideas carry over to tail-biting trellises. We present lower bounds on the state and branch complexity profiles of these representations. These bounds are expressed in terms of mutual information between different portions of the code, and they introduce the notions of superstates and superbranches. For linear block codes, our bounds imply that the total number of superstates, and respectively superbranches, of a tail-biting trellis of the code cannot be smaller than the total number of states, and respectively branches, of the corresponding minimal conventional trellis, though the total number of states and branches of a tail-biting trellis is usually smaller than that of the conventional trellis. We also develop some improved lower bounds on the state complexity of a tail-biting trellis for two classes of codes: the first-order Reed-Muller codes and cyclic codes. We show that the superstates and superbranches determine the Viterbi decoding complexity of a tail-biting trellis. Thus, the computational complexity of the maximum-likelihood decoding of linear block codes on a tail-biting trellis, using the Viterbi algorithm, is not smaller than that of the conventional trellis of the code. However, tail-biting trellises are beneficial for suboptimal and iterative decoding techniques.

key words: *trellises, tail-biting trellis, mutual information, maximum-likelihood decoding, Viterbi algorithm*

1. Introduction

The notion of *tail-biting* was originally conceived by Solomon and van Tilborg [12] to convolutionally encode and decode quasi-cyclic block codes. Ma and Wolf [7] have recognized that the tail-biting notion can be utilized to constitute an improved method to convert arbitrary convolutional codes to block codes. This method is superior to the previously used methods: truncation and termination. In a tail-biting trellis, as opposed to the conventional trellis, there are multiple starting states and equally many ending states. The sequences at the output of a tail-biting encoder are associated with trellis paths which begin and end in the same state.

Similarly to the research on trellis representation of block codes, it has been understood that linear block codes can also be represented by a tail-biting trellis. Thus, it uncovered an additional connection between convolutional and block codes. A tail-biting trellis for

a block code is based on a circular index set rather than a sequential index axis, and the transmission can be viewed as a path around a circular trellis. A path through the trellis is constrained to start and end in the same state. The construction of *linear* tail-biting trellises for linear block codes has been studied by Calderbank et al. [3] and by Kötter and Vardy [6]. Minimal linear tail-biting trellises for some short rate-1/2 block codes have been derived in [3]. The *trellis complexity* of a tail-biting trellis (total number of states and edges) is usually smaller than that of a conventional trellis. This complexity reduction is attributed to the additional degree of freedom achieved by not constraining the diagram to begin in the all zero state. That is, the dependence between two portions of a codeword is described by two states: the particular initial (and also final) state of the path and the state through which the codeword passes at the index of interest. The cardinality of each of the two state spaces may be smaller than the single state space that expresses the *past/future* dependence in the conventional trellis representation of the code.

The recent success of turbo codes and iterative decoding techniques to achieve performance very close to the Shannon limit has ignited renewed interest in these structures since they are the simplest graph with cycles (a single cycle). There is no theoretical analysis that explains the amazing performance of these codes in practice or the convergence properties of the sum-product algorithm applied to codes defined on graphs with cycles. Yet, a few theoretical results on iterative decoding on a tail-biting trellis have recently been proved. Anderson and Hladic [1] have proved that the maximum *a posteriori* algorithm, and thus also the maximum-likelihood algorithm, converge on a tail-biting trellis. The performance of iterative decoding on tail-biting trellises was also studied by Weiss [14].

Though the use of iterative decoding algorithms which possibly achieve near maximum-likelihood performance with reduced complexity is the major motivation for the recent investigation of tail-biting trellises, in this paper, we take an approach which is different from that pursued in former works. We shall not be concerned with iterative decoding techniques, but rather we investigate the structure of these graphs and the computational complexity of an exact maximum-likelihood decoding, using the Viterbi algorithm. Fur-

Manuscript received January 18, 1999.

Manuscript revised April 9, 1999.

[†]The authors are with the Department of Electrical Engineering-Systems, Tel-Aviv University, Ramat Aviv 69978, Tel-Aviv, Israel.

ther, we focus on comparing measures of trellis complexity (state and branch complexities) and the decoding complexity of tail-biting trellises for block codes with the corresponding complexity measures of the conventional trellis.

We elaborate our previously developed information-theoretic bounds [9] in order to apply them to tail-biting trellises. We present lower bounds on the state and branch complexity profiles of a tail-biting trellis for block codes. Clearly, not only linear codes can be depicted by a tail-biting trellis but also nonlinear codes. The bounds are formulated in terms of mutual information between two portions of the code symbols, and they may be viewed as an extension of the bounds of our previous work [9] to trellis representations which need not necessarily comprise single initial and final states. The lower bound on the state complexity profile generalizes a result of Wiberg [15] to nonlinear codes, and gives some insight into the conditions under which the bound is achieved. We also introduce a lower bound on the total number of branches between any two levels of the diagram. This bound also applies to nonlinear codes. We use these bounds to define the concepts of *superstates* and *superbranches* of a tail-biting trellis. The bounds imply that the total number of superstates of a tail-biting trellis is not smaller than the state count of the corresponding conventional trellis. The same relation holds between the total number of superbranches of a tail-biting trellis and the size of the branch set of the conventional trellis. These bounds pave the way to the analysis of the complexity of Viterbi decoding of tail-biting trellises. In particular, we develop a lower bound on the state complexity of a tail-biting trellis for two classes of codes: the first-order Reed-Muller codes $RM(1, m)$ and cyclic codes.

Finally, we analyze the maximum-likelihood decoding algorithm for linear block codes through a tail-biting trellis using the Viterbi algorithm. We use the notions of superstates and superbranches to count the total number of operations required by the maximum-likelihood Viterbi algorithm. It is shown that the Viterbi decoding of a tail-biting trellis is dominated by the total number of superstates and superbranches of the tail-biting trellis, rather than the total number of states and branches. Thus, we use the derivation of our bounds in order to prove that the Viterbi decoding complexity of a tail-biting trellis cannot be smaller than the decoding complexity of a conventional trellis. It is hence concluded that the use of a tail-biting trellis has a practical advantage over the conventional trellis only for the implementation of suboptimal and iterative decoding methods provided that their complexity is determined by the total number of states and branches of the tail-biting trellis and not the total number of superstates and superbranches. Instances for such iterative techniques have been suggested by Anderson and Hladic [1].

2. Tail-Biting Trellises

In the sequel, we basically use the nomenclature of [9]. We denote by (n, M) a length- n nonlinear block code that comprises M codewords. When we refer to a linear code of dimension k we denote the parameters of the code by (n, k) . A tail-biting trellis for an (n, M) block code is defined on a circular index set I of length n . That is, the index set I is identified with the set of integers modulo n , \mathbf{Z}_n .

A tail-biting trellis $T = (V, A, E)$ for a block code is an edge-labeled directed graph in which the *vertex (state)* set V is the union of disjoint subsets, $V = \bigcup_{i=0}^{n-1} V_i$, provided that $V_0 = V_n$, the *edge* set $E = \bigcup_{i=1}^n E_i$, consists of ordered triples $E_i = \{(v', \alpha, v'') : v' \in V_{i-1}, v'' \in V_i, \alpha \in A_i\}$, where A_i is the alphabet set at index i , and $A = A_1 \times A_2 \times \cdots \times A_n$ is a finite alphabet set. Any path $(\mathbf{v}, \boldsymbol{\alpha}) \in V \times A$ from V_0 to V_n defines a state sequence and its corresponding n -tuple sequence of edge labels $(\alpha_1, \alpha_2, \dots, \alpha_n)$. Along this work, we use the term *branch* to describe a partial path through the trellis that connects states at different levels. These states need not necessarily be at adjacent levels. The state complexity of the trellis diagram T at level i , $s_i(T)$, is the logarithm of the vertex count at this level, i.e., $s_i(T) \triangleq \log |V_i|$. All the logarithms are taken to the base q , the cardinality of the alphabet set over which the code is defined. The sequence $\mathbf{s}(T) = \{s_i(T), 0 \leq i \leq n-1\}$ is the *state complexity profile* of C . The *state complexity* under a given coordinate permutation is the maximum value of $\mathbf{s}(T)$, $s_{\max}(T) \triangleq \max_i \{s_i(T)\}$. For a linear code C , we denote by T_0 the unique minimal conventional trellis for C under a given symbol order. The minimum $s_{\max}(T_0)$ over all coordinate orderings is called the *state complexity* of C , $s(C)$.

Let $P_J(\mathbf{c})$ denote the projection of a codeword $\mathbf{c} \in C$ onto $J \subseteq I$. That is, if $J = (i_1, i_2, \dots, i_{|J|})$ and $(c_1, c_2, \dots, c_n) = \mathbf{c} \in C$ then $P_J(\mathbf{c}) = (c_{i_1}, c_{i_2}, \dots, c_{i_{|J|}})$. The set of the projection of all the codewords of C onto J is denoted by $P_J(C)$. We use the following notations for two particular complementary subsets: $i^- \triangleq [1, 2, \dots, i]$ and $i^+ \triangleq [i+1, i+2, \dots, n]$, where $0 \leq i \leq n$, with the convention that 0^- and n^+ are empty subsets. Let $B_{i,j}$ denote the set of branches (paths) between states at indices i and j , $j > i$, in a tail-biting trellis T . Each branch is described by the triple $(v_i, P_{[i+1,j]}(\mathbf{c}), v_j)$, where $v_i \in V_i$, $v_j \in V_j$, and $\mathbf{c} \in C$. Similarly, we define $b_{i,j}(T)$, the *branch complexity* of a tail-biting trellis T between the levels i and j , as $b_{i,j}(T) \triangleq \log |B_{i,j}|$. This definition is also applicable to a given coordinate permutation. A branch between adjacent levels $i-1$ and i is henceforth called an *edge*, and the *edge complexity* at level i is defined as

$b_{i-1,i}(T) = \log |E_i|$, where $E_i \triangleq B_{i-1,i}$.

In the next section, we prove some theorems applying to any tail-biting trellis representation of a block code C . This pertains to any edge-labeled graph T of the foregoing structure. The diagram should have the following properties:

- (1) The graph comprises the same number of initial states and ending states: $s_0(T) = s_n(T)$.
- (2) There is at least one path from one of the initial vertices to every vertex in the levels $1 \leq i \leq n - 1$, and at least one path from each vertex (at the above levels) to one of the final states.
- (3) The set $C(T)$ of n -tuples corresponding to all the paths that start and end at the same trellis state is identical to the set of the codewords of C .
- (4) Every trellis edge lies on some valid path that starts and ends at the same trellis state.

Following the definitions of Kötter and Vardy [6], and in order to define a linear tail-biting trellis, we label the vertices of the diagram as follows. Each vertex $v \in V_i$, in a tail-biting trellis T for a block code C over $\text{GF}(q)$, is labeled by a length- $\lceil s_i(T) \rceil$ sequence over $\text{GF}(q)$. All the vertices in the same vertex set V_i are labeled distinctly. We define $s = \lceil s_1(T) \rceil + \lceil s_2(T) \rceil + \dots + \lceil s_n(T) \rceil$. Every tail-biting trellis T defines a set $S(T)$ of ordered sequences of length $n + s$. Each sequence consists of the labels of the edges and vertices of a valid path through the trellis, i.e., a path that starts and ends in the same state. The set $S(T)$ is referred to as the edge-vertex label code of T .

Definition 1 [6]: A tail-biting trellis T of a linear code is said to be linear if there exists a labeling of the vertices of T such that $S(T)$ is a linear code.

Any linear tail-biting trellis for an (n, k) linear code C can be constructed as a trellis product [5], [11] of the representation of the individual trellises corresponding to m codewords of C , with $m \geq k$ [6]. This procedure for generating a tail-biting trellis for linear block codes has been described in [3]. The trellis product of $T_1 = (V', A, E')$ and $T_2 = (V'', A, E'')$ is the Cartesian product $T = (V, A, E) = T_1 \times T_2$ such that if $C_1 = C(T_1)$ and $C_2 = C(T_2)$ then the product trellis T represents the code

$$C = C_1 + C_2 = \{c_1 + c_2 : c_1 \in C_1, c_2 \in C_2\}.$$

Accordingly, V and E are defined as follows:

$$V_i \triangleq V'_i \times V''_i = \{(v', v'') : v' \in V'_i, v'' \in V''_i\},$$

$$E_i \triangleq \{[(v', v''), \alpha_1 + \alpha_2, (u', u'')] : (v', \alpha_1, u') \in E'_i, (v'', \alpha_2, u'') \in E''_i\}.$$

Thus, the state and the branch complexity profiles of the trellis product are the sums of the corresponding complexity measures of the constituent trellises. Consequently, minimal linear tail-biting trellises for a linear (n, k) code may be constructed as a trellis product

of the representation of the individual trellises corresponding to precisely k linearly independent codewords of C [6].

Let c be a member of an (n, k) linear code C . We define the circular interval as follows:

$$[i, j]_n \triangleq \begin{cases} \{i, i + 1, \dots, j\}, & i \leq j \\ \{i, i + 1, \dots, n; 1, 2, \dots, j\} & i > j \end{cases}$$

Let $[i, j]_n$ be a set of consecutive zeros in a non all zeros codeword $c \in C$. The span [5] of c is defined as the index set $[j + 1, i - 1]_n$, where all the indices are taken modulo n . The span of a codeword with no zeros is $[1, n]_n$. It should be noted that the span of c is not unique, and it depends on the choice of the consecutive zeros in the codeword. A conventional trellis does not have this degree of freedom in defining the spans of the codewords of the generator matrix. Consequently, any linear block code has a unique minimal trellis but numerous tail-biting trellises, each minimizing different complexity measures. A codeword $c \in C$ with span $[a, b]_n$ is said to be active [5] at every index of the circular interval $[a, b - 1]_n$ and inactive otherwise. The interval $[a, b - 1]_n$ will be referred to as the active interval of the codeword. A codeword with span $[a, a]_n$ is never active. The state complexity at index i of a trellis product generated by the codewords $\{g_1, g_2, \dots, g_k\}$ is exactly the number of codewords g_i active at index i . The edge complexity of a trellis product generated by the codewords $\{g_1, g_2, \dots, g_k\}$ at index i is exactly the number of codewords g_i whose spans comprise the index i . Finally, it is noteworthy to comment that linear tail-biting trellises for linear block codes need not be the minimal trellises for the code.

Example 1: A tail-biting trellis for a binary linear $(6, 3)$ code whose minimum distance is 3 was given in [15], and it is depicted in Fig. 1. This trellis may be constructed as a trellis product of the following three codewords $\{111000, 001110, 100011\}$. The spans of these codewords are chosen to be $\{[1, 3]_6, [3, 5]_6, [5, 1]_6\}$. Thus, at each index there is only one active codeword, and the state complexity of the resulting trellis is 1 for all indices thereupon.

There is no equivalent systematic method to construct a tail-biting trellis for a nonlinear code. Nevertheless, many classes of nonlinear codes can be defined as the sum of two constituent codes, i.e., $C = C_1 + C_2$, where one of the component codes is linear. In this case, the problem of finding a tail-biting representation for the code amounts to a much simpler problem of generating a tail-biting trellis for the constituent codes.

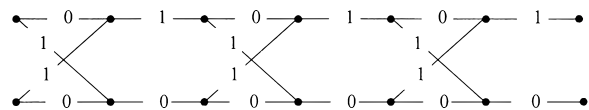


Fig. 1 A tail-biting trellis for the $(6, 3)$ linear code.

3. State and Branch Complexity Profiles of a Tail-Biting Trellis

Under a given coordinate ordering, a linear code has a unique minimal trellis representation that minimizes the total number of states and the total number of branches at all the levels of the diagram, simultaneously [8]. There is no equivalent minimal tail-biting representation of a linear code. The state count and the edge count at the different levels of a tail-biting trellis are dependent on each other, and there is always a tail-biting trellis for a linear code with a single state at any given (single) trellis level.

In this section, we discuss the complexity of tail-biting trellises of block codes. We develop a lower bound on the state and branch complexity profiles of the diagram, under a given ordering of the code components. The bounds apply to any tail-biting representation of any block code, and they are not limited to linear tail-biting trellises. The results also apply to nonlinear codes. These bounds can easily be extended to any coordinate ordering, as done in [9]. The lower bound on the state complexity profile extends a result by Wiberg to nonlinear codes. Our derivation presents the conditions under which the tail-biting representation of the code meets the bounds. In a previous work on conventional trellis representation for block codes [9], we showed that the states and branches of a trellis represent the information induced by one portion of the codewords on the other portion. Loosely speaking, in the present study, we show that this information is split into two portions of the tail-biting trellis: the initial (and end) states of the codewords and the state or branch set at the level of interest.

Similarly to [9], we regard the (n, M) code C as an ensemble whose sample space is the set of codewords, and we assign each codeword a uniform probability of $1/M$. We denote by X_{i-} a random i -tuple variable that takes on the values of the set $P_{i-}(C)$ with a probability function that is imposed by the uniform probability of each entire codeword. Similarly, X_{i+} will denote a random $(n-i)$ -tuple variable whose sample space is the set $P_{i+}(C)$ with probabilities that are induced by the uniform distribution of the codewords. Given a tail-biting trellis for the code, we denote by S_i an ensemble whose sample space is the vertex set of the tail-biting trellis at level i . Again, the probability distribution of this ensemble is imposed by the (uniform) distribution of the codewords.

We express our bounds by means of the basic information-theoretic measures: the *entropy* of an ensemble X , $H(X)$, the *joint entropy*, $H(XY)$, of the joint ensemble XY , and the *mutual information* of this joint ensemble $I(X; Y)$.

Due to the underlying circular index set of a tail-biting trellis, there is complete symmetry between all

indices. However, we henceforth refer to one of the vertex sets V_i as the set of initial states, and denote it by V_0 . This trellis level may be taken to be any one of the n trellis levels.

Definition 2: A *superstate* \hat{v} of a tail-biting trellis at level i is a state-pair $[v', v'']$ such that $v' \in V_i, v'' \in V_0$, and these two states are connected through a valid path (a path that starts and ends at the same trellis state).

An (n, M) block code C poses a relation between the set of the past and future projections of its codewords at each level. Since $C \subseteq P_{i-}(C) \times P_{i+}(C), \forall i, C$ can be viewed as a code of length two over the alphabet $P_{i-}(C) \times P_{i+}(C)$. A tail-biting trellis for C induces a decomposition of $P_{i-}(C)$ and $P_{i+}(C)$ as follows. Let P_{ij} denote the label set of the trellis branches from level 0 to the j th superstate at level i . Clearly, these branches start at the same initial state and terminate at the same state at level i . Similarly, we denote by F_{ij} the labels of branches connecting this j th superstate at level i to the corresponding final state at level n . We also denote by R the total number of superstates at level i . Thus,

$$P_{i-}(C) = \bigcup_{j=1}^R P_{ij}, \quad P_{i+}(C) = \bigcup_{j=1}^R F_{ij},$$

$$C = \bigcup_{j=1}^R P_{ij} \times F_{ij}.$$

We use this decomposition to prove the following lower bound on the state complexity profile of a tail-biting trellis.

Theorem 1: The state complexity profile of any tail-biting trellis T of an (n, M) block code C is bounded by

$$s_i(T) \geq I(X_{i-}; X_{i+}) - s_0(T), \quad 1 \leq i \leq n-1. \quad (1)$$

Equality holds in (1) if and only if the following conditions hold:

- (a) $H(X_{i-}|X_{i+}S_iS_0) = H(X_{i-}|X_{i+})$,
- (b) $H(X_{i+}|X_{i-}S_iS_0) = H(X_{i+}|X_{i-})$,
- (c) S_i and S_0 are statistically independent, i.e., $H(S_iS_0) = H(S_i) + H(S_0)$,
- (d) S_iS_0 is a uniform ensemble.

Proof: The proof is based on Theorem 4 of [9]. Given a tail-biting trellis T for the code, we define a joint $X_{i-}X_{i+}S_iS_0$ ensemble where X_{i-} , X_{i+} , and S_i are the ensembles defined above. The sample space, Ψ , of $X_{i-}X_{i+}S_iS_0$ is the set of quadruples $\{(x, y, v_i, v_0) : (x, y) \in C, v_i \in V_i, v_0 \in V_0\}$. Nonzero probabilities are assigned only to quadruples (x, y, v_i, v_0) that describe codewords of C through the tail-biting diagram. Using the argumentation and formulation of [9] we obtain

$$H(S_iS_0) \geq \log_q M - H(X_{i+}|X_{i-}) - H(X_{i-}|X_{i+})$$

$$= I(X_{i-}; X_{i+}). \quad (2)$$

Equality holds in (2) if and only if conditions (a) and (b) are satisfied. Likewise we have

$$H(S_i S_0) \leq s_i(T) + s_0(T), \quad (3)$$

with equality if and only if conditions (c) and (d) hold. Finally, the last two inequalities establish the bound of (1). \square

We note that conditions (a) and (b) imply that the tail-biting trellis is “biproper.” That is, for any i , $1 \leq i \leq n-1$, all the codewords with the same first i components are depicted by the same length- i branch between the indices 0 and i , and all the codewords with the same last $(n-i)$ components are depicted by the same length- $(n-i)$ branch between the indices i and n . We also notice that in Theorem 1, the sample space of the joint ensemble $S_i S_0$ is the set of superstates at level i . Inequality (2) indicates that the log-cardinality of this set is not smaller than the mutual information between the past and future portions of the code. Hereafter we will find that for decoding purposes, inequality (2), rather than (1), is the important relation. For linear codes, the mutual information $I(X_{i-}; X_{i+})$ is identical to the difference between the inverse ordered *dimension/length profile* (DLP, [4]) and the ordered DLP. This difference is also the dimension of the state space of the conventional minimal trellis at level i . This relation between the mutual information and the state complexity profile of block codes has been shown in [8] and [9]. Consequently, for linear block codes, inequality (2) implies that the total number of superstates in a tail-biting trellis for a code cannot be smaller than the total number of states in the conventional trellis for the code at the same trellis level. However, we note that the bound of inequality (1) implies that the total number of states of the tail-biting trellis may be smaller than the respective number of states of the conventional trellis. For linear codes, the bound also implies that the maximum value of the state complexity profile of a tail-biting trellis T for C cannot be smaller than $\lceil \frac{1}{2} s_{\max}(T_0) \rceil$, where $s_{\max}(T_0)$ is the maximum entry of the state complexity profile of the unique minimal conventional trellis T_0 for the code, under the same symbol permutation. Similarly, under any coordinate permutation the state complexity of a tail-biting trellis is not smaller than $\lceil \frac{1}{2} s(C) \rceil$. The latter bound will be referred to as the *square root bound*. It is noteworthy to comment that in [3], the square root lower bound refers to a looser bound, $\lceil \frac{1}{2} s_{n/2}(T_0) \rceil$.

Example 1 (Cont.): The minimal (conventional) trellis for the (6,3) code generated by

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

has the following state complexity profile $\{0, 1, 2, 2, 1, 0\}$. The entries of this profile take on the values of $I(X_{i-}; X_{i+})$ where i varies over the index set

$I = \{0, 1, \dots, 6\}$. The state complexity profile of the tail-biting trellis of Fig. 1 meets our bound at indices 2–4. At index 1, condition (b) is violated: there are two edges with the same label (both 0 and 1) from the initial states to the states at level 1, i.e., the tail-biting trellis is *improper*. At index 5, condition (c) is violated: each state at level 5 is connected to only one end state, and thus there are only two superstates (state pairs), $H(S_5 S_0) = H(S_5) = H(S_0) = 1$.

The next theorem bounds the total number of branches between any two levels i and j , $j > i$. We lower bound the total number of such branches provided that they are comprised in valid paths that represent codewords (paths that begin and terminate at the same trellis state).

Theorem 2: The branch complexity between the indices i and j , $j > i$, of any tail-biting trellis T for an (n, M) block code C is bounded by

$$b_{i,j}(T) \geq I(X_{j-}; X_{i+}) - s_0(T), \quad 0 \leq i < j \leq n. \quad (4)$$

Proof: The proof is very similar to that of Theorem 1. Given a tail-biting trellis T for the code, we define a joint $X_{j-} X_{i+} Z_{i,j} S_0$ ensemble, where $Z_{i,j}$ is an ensemble that takes on the values of the set $B_{i,j}$. The sample space, Ψ , of $X_{j-} X_{i+} Z_{i,j} S_0$ is the set of quadruples $\{(x, y, b, v_0) : (x, y) \in \tilde{C}, b \in B_{i,j}, v_0 \in V_0\}$, where $\tilde{C} \triangleq \{[P_{j-}(\mathbf{c}), P_{i+}(\mathbf{c})] : \mathbf{c} \in C\}$, and $B_{i,j}$ is the set of branches in the tail-biting trellis between the levels i and j . Nonzero probabilities are assigned only to quadruples (x, y, b, v_0) that describe codewords of C through the tail-biting diagram. Using the development of [9], it can be shown that

$$\begin{aligned} H(Z_{i,j} S_0) &\geq \log_q M - H(X_{j+} | X_{j-}) - H(X_{i-} | X_{i+}) \\ &= I(X_{j-}; X_{i+}), \end{aligned} \quad (5)$$

and again,

$$H(Z_{i,j} S_0) \leq b_{i,j}(T) + s_0(T). \quad (6)$$

The theorem follows from the above inequalities. \square

The conditions under which equality holds in (4) are similar to those stated for the state complexity bound. Hence the use of $|V_0|$ starting states in a tail-biting trellis for a linear block code cannot reduce the branch count of a conventional trellis by a factor larger than $|V_0|$.

As shown in our previous work [9], the set \tilde{C} is a useful representation of the codewords of C for the purpose of bounding the total number of branches between any given two levels. Clearly, $[P_{j-}(\mathbf{c}), P_{i+}(\mathbf{c})]$, the concatenation of $P_{j-}(\mathbf{c})$ and $P_{i+}(\mathbf{c})$ does not provide a codeword of C since these two portions of \mathbf{c} have $(j-i)$ common symbols of a codeword. Each branch of a trellis can be identified with a complete rectangle of a Cartesian array describing the relation \tilde{C} , i.e., an array in which each row is associated with a different value of

$P_{j-}(\mathbf{c})$ for some $\mathbf{c} \in C$, and each column is associated with a different value of $P_{i+}(\mathbf{c})$ for some $\mathbf{c} \in C$. Thus, using this description, the branch set between any two levels also represents a covering of \tilde{C} by product-form subsets, and the problem of minimizing the total number of branches between any given two trellis levels is equivalent to the problem of finding a minimal covering by complete rectangles of the Cartesian array for \tilde{C} .

Definition 3: A *superbranch* \hat{b} is a pair $[b, v]$, where v is an initial state connected to b , a branch of the tail-biting trellis, via a valid path through the diagram.

Again, in Theorem 2, the sample space of the joint ensemble $Z_{i,j}S_0$ is the set of superbranches (between levels i and j). Inequality (5) indicates that the log-cardinality of this set cannot be smaller than $I(X_{j-}; X_{i+})$. The latter term is evidently the branch complexity of a conventional trellis for a linear code. We also recall that in our proof we counted only the branches (between levels i and j) that lie on a valid path that starts and ends in the same trellis state, and hence it depicts a codeword. Obviously, the total number of branches in this trellis section is not smaller than the devised bound.

The next theorem bounds the state complexity of a tail-biting representation for the first-order Reed-Muller codes $RM(1, m)$. It implies that actually a very minor complexity reduction as compared to the conventional trellis may be achieved by a tail-biting trellis. The theorem pertains to the standard bit-order of the code.

Let $\mathbf{v} = (v_1, v_2, \dots, v_m)$ be an m -tuple of binary variables. We denote by $c(\mathbf{v})$ a Boolean function of these m binary variables, and we generate a related vector of length 2^m , $\mathbf{c} = (c_1, c_2, \dots, c_{2^m})$. For $1 \leq i \leq 2^m$, c_i is the value of the function $c(\mathbf{v})$, where \mathbf{v} is the binary expansion of $i - 1$,

$$i - 1 = \sum_{j=1}^m b_{i,j}2^{j-1}, \quad c_i = c(b_{i,1}, b_{i,2}, \dots, b_{i,m}).$$

The first-order Reed-Muller code $RM(1, m)$ is the binary span of the codewords associated with the $m + 1$ Boolean functions $\{1, v_1, v_2, \dots, v_m\}$.

Theorem 3: The state complexity of a linear tail-biting trellis T for the first-order Reed-Muller codes, when arranged in the standard bit-order satisfies.

$$s_{\max}(T) \geq m - 1 \tag{7}$$

Proof: The state complexity of a linear tail-biting trellis for the code is determined by the span of the $m + 1$ (independent) codewords chosen to generate the diagram. We denote these codewords (generators) by $\{\mathbf{g}_1, \mathbf{g}_2, \dots, \mathbf{g}_{m+1}\}$. The Boolean functions that correspond to these generators are

$$g_i(\mathbf{v}) = b_{i,0}1 + \sum_{j=1}^m b_{i,j}v_j,$$

where $\{b_{i,0}, b_{i,1}, \dots, b_{i,m}\}$ is a set of $(m + 1)$ binary variables. A generator \mathbf{g}_i with $b_{i,j} = 1$ cannot have more than 2^j consecutive zeros, provided that $j \neq m - 1$. Consequently, the span of this generator is at least $2^m - 2^j$. We denote $d_i \triangleq \min\{j : b_{i,j} = 1\}$. Since the generators are independent, we assume, without loss of generality, that the generator set is ordered such that $d_i \leq m + 2 - i, i > 1$.

Thus, there are only three independent codewords, $\{0^{2^{m-1}}1^{2^{m-1}}, 1^{2^{m-1}}0^{2^{m-1}}, 0^{2^{m-2}}1^{2^{m-1}}0^{2^{m-2}}\}$, with the minimum possible span, 2^{m-1} . The fourth generator must have $b_{4,j} = 1$ for at least one value of j such that $j \leq m - 2$, and hence its span is at least $2^m - 2^{m-2}$. Similarly, the span of the fifth codeword (for $m \geq 4$) is not smaller than $2^m - 2^{m-3}$, and so forth. Consequently, the sum of lengths of the active intervals of all generators S_{tot} must satisfy.

$$\begin{aligned} S_{tot} &\geq (m + 1) \cdot 2^m - 3 \cdot 2^{m-1} - \sum_{j=1}^{m-2} 2^j - (m + 1) \\ &= (m - 1)(2^m - 1). \end{aligned}$$

Finally, this relation implies that there is at least one index at which the state complexity is not smaller than $\lceil (m - 1)(1 - 2^{-m}) \rceil = m - 1$. \square

The state complexity of the conventional trellis for the first-order Reed-Muller codes $RM(1, m)$ is $m \lceil 2 \rceil$. Thus, Theorem 3 shows a significant improvement on the square root bound $\lceil m/2 \rceil$. It implies that a tail-biting trellis for these codes may achieve just a very meager reduction of the vertex count relative to that of the conventional trellis. The next theorem lower bounds the state complexity of a tail-biting trellis for cyclic codes.

Theorem 4: Let C be an (n, k) cyclic code. The state complexity of a linear tail-biting trellis T for C is bounded by

$$s(T) \geq k - \left\lfloor \frac{k^2}{n} \right\rfloor = n - k - \left\lfloor \frac{(n - k)^2}{n} \right\rfloor. \tag{8}$$

Proof: The code C is generated by a polynomial of degree $(n - k)$, and hence the span of each of the codewords of C is at least $(n - k + 1)$. The sum of the lengths of the active intervals of any k codewords is at least $k(n - k)$. Again, this leads to the bound of (8). \square

Applying Theorem 4 to the cyclic bit-order of the first-order Reed-Muller codes $RM(1, m)$, it is easily proved that the bound on the state complexity under this bit-order does not improve on Theorem 3. The conventional trellis T_0 for cyclic codes has the worst possible state complexity profile, namely: $s_i(T_0) = \min\{i, n - i, k, n - k\}$ [16]. The state complexity of these codes is $\min\{k, n - k\}$. The difference between the state complexity of the conventional trellis of cyclic codes and our bound on the state complexity of a tail-biting trellis

ing of one level of a tail-biting trellis is equal to the total number of superedges at this level.

Let us check the calculation required to accomplish a decoding step of one trellis level. The vertex set and the superstate set of the trellis at level i is denoted V_i and \hat{V}_i , respectively. Similarly, we denote by E_i and \hat{E}_i , the set of edges and respectively, superedges, branching to vertices at level i . The algorithm proceeds by storing a cost for each superstate at the level of computation and the cost of edges at the succeeding levels. Let (v_{i-1}, α, v_i) and U be an edge in the examined trellis section and the subset of initial states connected to this edge through a valid path. The cost of this edge should be added to the set of superstates $\{[v_{i-1}, u] : u \in U\}$. Thus, the log-cardinality of the total number of additions required for the i th stage is $b_{i-1,i} + \rho_i$, i.e., at least the same number of additions required to carry out the decoding algorithm on the conventional trellis at the same level. Theorems 1 and 2 imply that the total number of vertex costs, $|\hat{V}_i| = q^{r_i} |V_i|$, and the number of additions, $|\hat{E}_i| = q^{\rho_i} |E_i|$, are not smaller than the respective complexity measures of the rival (conventional) trellis. We also define,

$$\hat{V} \triangleq \bigcup_{i=1}^{n-1} \hat{V}_i, \quad \hat{E} \triangleq \bigcup_{i=1}^n \hat{E}_i. \quad (9)$$

The total number of comparisons required for Viterbi decoding of each step of a tail-biting trellis is $|\hat{E}_i| - |\hat{V}_i|$ for $1 \leq i \leq n-1$, and $|\hat{E}_i| - 1$ comparisons are required at the final level, and hence $|\hat{E}| - |\hat{V}| - 1$ comparisons altogether. This is the number of merges (expansions) of the diagram. Our theorems do not ensure that this number of operations is not smaller than the corresponding computational complexity of the conventional trellis, and we shall prove this relation in a slightly different way. The decoding of a tail-biting trellis T is equivalent to the decoding of a standard n -section tree \mathcal{T} . This tree includes a single initial state. The vertex set of the new graph \mathcal{T} at level i consists of the superstates of the original tail-biting trellis at the same level. The edge set of the graph is precisely the set of superedges of the tail-biting trellis, and a superedge $\hat{e} = [e, u] = [(v_{i-1}, \alpha, v_i), u]$ branches from the superstate $[v_{i-1}, u]$ and incidents to the superstate $[v_i, u]$. At the last level all the edges enter the same single final state. The resulting graph is a conventional *supertrellis* diagram for the code which corresponds to the tail-biting trellis. This supertrellis need not be the minimal trellis for the code. This equivalent representation indicates that the total number of comparisons required for the Viterbi decoding cannot be smaller than the corresponding computational complexity of the conventional minimal trellis due to the fact that the minimal trellis also minimizes the *expansion index* of the trellis, $|E| - |V| + 1$ ([10], [13]). We thus conclude the following theorem.

Theorem 5: The arithmetic complexity of the Viterbi decoding of a tail-biting trellis is given by $2|\hat{E}| - |\hat{V}| - 1$ ($|\hat{E}|$ additions and $|\hat{E}| - |\hat{V}| - 1$ comparisons). This number accounts for the total number of additions and comparisons required by the algorithm. For linear block codes, this complexity is not smaller than the decoding complexity using the conventional trellis for the code.

Example 1 (Cont.): For the tail-biting trellis of Fig. 1 we have $|\hat{E}| = 26$ and $|\hat{V}| = 18$. The number of operations (additions/comparisons) required to carry out the six decoding steps on this trellis is $\{(4/0), (4/0), (8/4), (4/0), (4/2), (2/1)\}$. Except for the first level, this computational complexity breaks even with that of the conventional trellis that requires just two additions at the first step. We point out, however, that in general, the difference may be much larger.

The comparison carried out in this section applies to any coordinate ordering of the code. Consequently, the computational complexity of a tail-biting trellis under an *efficient* permutation is not smaller than the decoding complexity of the standard trellis under the same symbol ordering. The decoding complexity of a conventional trellis under an efficient coordinate permutation is *a fortiori* smaller than that of a tail-biting trellis under any permutation. Furthermore, it is well known that the decoding complexity may be reduced by sectionalization [4], i.e., by dividing the index set into sections of lengths greater than one. Yet, Theorem 2 guarantees that when we apply the same sectionalization to both the conventional trellis and the tail-biting trellis, the former will be advantageous for Viterbi decoding.

Eventually, we mention that a different, but equivalent, algorithm for maximum-likelihood decoding on a tail-biting trellis involves $|V_0|$ independent Viterbi decoders for each of the subtrellises comprising all the paths that start and end in the same trellis state [3]. That is, all the superedges and superstates associated with the same initial state generate a conventional trellis. Hence a tail-biting trellis can be treated as a set of $|V_0|$ distinct conventional trellises without any cross connections. The total number of additions and comparisons required to carry out the Viterbi decoding on these $|V_0|$ trellises at level i is $|\hat{E}_i|$ and $|\hat{E}_i| - |\hat{V}_i|$, respectively. Finally, the best path among the $|V_0|$ best paths of each trellis should be chosen. This final choice is accomplished by $|V_0| - 1$ comparisons. Therefore, this strategy involves exactly the same number of additions and comparisons which are required in our scheme. The equivalence of the two decoding schemes is ascribed to the fact that the superstates and superedges actually realize the decomposition of the graph into distinct subtrellises associated with different initial states. Clearly, in our analysis, no calculation involves superstates or superedges which are identified with different initial

state. These results suggest that the computational complexity of maximum-likelihood decoding methods through a tail-biting trellis is not smaller than that of the conventional trellis.

Acknowledgment

The authors wish to thank the anonymous reviewers for their helpful comments.

References

- [1] J.B. Anderson and S.M. Hladik, "Tailbiting MAP decoders," *IEEE J. Select. Areas Commun.*, vol.16, pp.297–302, Feb. 1998.
- [2] Y. Berger and Y. Be'ery, "Bounds on the trellis size of linear block codes," *IEEE Trans. Inf. Theory*, vol.39, pp.203–209, Jan. 1993.
- [3] A.R. Calderbank, G.D. Forney, Jr., and A. Vardy, "Minimal tail-biting trellises: The Golay code and more," *IEEE Trans. Inf. Theory*, vol.45, pp.1435–1455, July 1999.
- [4] G.D. Forney, Jr., "Dimension/length profiles and trellis complexity of linear block codes," *IEEE Trans. Inf. Theory*, vol.40, pp.1741–1752, Nov. 1994.
- [5] F.R. Kschischang and V. Sorokine, "On the trellis structure of block codes," *IEEE Trans. Inf. Theory*, vol.41, pp.1924–1937, Nov. 1995.
- [6] R. Kötter and A. Vardy, "Construction of minimal tail-biting trellises," *Proc. IEEE Information Theory Workshop*, pp.72–74, Killarney, Ireland, June 1998.
- [7] H.H. Ma and J.K. Wolf, "On tail biting convolutional codes," *IEEE Trans. Commun.*, vol.COM-34, pp.104–111, Feb. 1986.
- [8] R.J. McEliece, "On the BCJR trellis for linear block codes," *IEEE Trans. Inf. Theory*, vol.42, pp.1072–1092, July 1996.
- [9] I. Reuven and Y. Be'ery, "Entropy/length profile, bounds on the minimal covering of bipartite graphs, and trellis complexity of nonlinear codes," *IEEE Trans. Inf. Theory*, vol.44, pp.580–598, March 1998.
- [10] V.R. Sidorenko, "The Euler characteristic of the minimal code trellis is maximum," *Problems of Inform. Transm.*, vol.33, no.1, pp.72–77, March 1997.
- [11] V. Sidorenko, G. Markarian, and B. Honary, "Minimal trellis design for linear codes based on the Shannon product," *IEEE Trans. Inf. Theory*, vol.42, pp.2048–2053, Nov. 1996.
- [12] G. Solomon and H.C.A. van Tilborg, "A connection between block and convolutional codes," *SIAM J. Appl. Math.*, vol.37, pp.358–369, Oct. 1979.
- [13] A. Vardy and F.R. Kschischang, "Proof of a conjecture of McEliece regarding the expansion index of the minimal trellis," *IEEE Trans. Inf. Theory*, vol.42, pp.2027–2034, Nov. 1996.
- [14] Y. Weiss, "Belief propagation and revision in networks with loops," *M.I.T. A.I. Memo*, no.1616, Nov. 1997.
- [15] N. Wiberg, "Codes and decoding on general graphs," Ph.D. dissertation, Dept. Electrical Engineering, University of Linköping, Sweden, April 1996.
- [16] J.K. Wolf, "Efficient maximum likelihood decoding of linear block codes using a trellis," *IEEE Trans. Inf. Theory*, vol.IT-24, pp.76–80, July 1978.



Ilan Reuven was born in Ramat Gan, Israel, on June 12, 1968. He received the B.Sc. and M.Sc. degrees from the Tel Aviv University, Tel Aviv, Israel, in 1990 and 1995, respectively, both in electrical and electronic engineering. He is currently working towards the Ph.D. degree in the same institute. His current research interests include coding theory.



Yair Be'ery was born in Tel Aviv, Israel, on July 6, 1956. He received the B.Sc., M.Sc., and Ph.D. degrees, all in electrical engineering, from Tel Aviv University, Israel, in 1979, 1979, and 1985, respectively. From 1979 to 1985 he was a Senior Research Engineer in the Research Laboratories, Israeli Ministry of Defense. Since 1985 he has been with the Department of Electrical Engineering-Systems, Tel Aviv University, where he is an Associate Professor of Electrical Engineering. From 1979 to 1986 he held various positions at DSP Group including Vice President of Advanced Technology. His research interests include error control coding, combined coding and modulation, VLSI architectures for systolic arrays, and DSP cores. Dr. Be'ery is a recipient of the 1984 Eliyahu Golomb Award from the Israeli Ministry of Defense, the 1986 Rothschild Fellowship for Post-Doctoral studies at Rensselaer Polytechnic Institute, Troy, NY, and of the 1992 Electronic Industry Award in Israel.